Myanmar
Heritage

Java Programming ရေးသားခြင်းနှင့် ပတ်သက်၍ အခြေခံအကျဆုံး လမ်းညွှန်စာအုပ်

ကိုယ်တိုင်လေ့လာ Java သင်ခန်းစာ

- Java Basics
- Programming Basics
- Java and C++
- Applets Programming
- Graphics using Appets
- Object-Oriented Programming

ကွန်ပျူတာဂျာနယ် စာတည်းအဖွဲ့

ကိုယ်တိုင်လေ့လာ

JAVA

သင်ခန်းစာ

အေးအေးလိုင်
မြန်မာဘာသာပြန်သည်

Myanmar Heritage

Myanmar
Heritage

ပုံနှိပ်ခြင်း
ပထမအကြိမ်၊ ၂၀၁၁ခုနှစ် ဩဂုတ်လ၊
စောင်ရေ ၅၀၀၊

❑

ထုတ်ဝေသူ
ဒေါ်ခင်မာဆိုး(ဝဒုဂုဒ)၊ ပန်းဝေဝေစာပေ
၃၆၀၊ ဝိုလ်အောင်ကျော်လမ်း၊ ကျောက်တံတားမြို့၊ ရန်ကုန်တိုင်းဒေသကြီး။

❑

ပုံနှိပ်သူ
ဦးမောင်လွင်၊ နေလင်းပုံနှိပ်တိုက်
အမှတ် ၅၆၊
တောင်ဒဂုံစက်မှုဇုန် ၂၊ ရန်ကုန်တိုင်းဒေသကြီး။

မာတိကာ

အခန်း (၁)

Introduction to JAVA

ကျွန်ုပ်တို့မှာ C, C++ programming languages များ ရှိခဲ့ရာမှ ယခု Java language ရောက်ရှိလာပြီဖြစ်သည်။ ယခု JAVA ကို လေ့လာကြည့်ကြပါမည်။ Language များသည် တစ်ဆင့်ပြီးတစ်ဆင့် တိုးတက်ဖြစ်ပေါ်လာနေရာ JAVA ၏ character အများစုသည် C နှင့် C++ language နှစ်ခုမှ ဆင်းသက်လာသည်။ ၎င်းပြင် JAVA ၏ syntax များသည် C မှ Syntax များဖြစ်ပြီး Object-oriented features အများစုသည်လည်း C++ မှ ဖြစ်သည်။ Java ၏ တည်ဆောက်ပုံသည် လွန်ခဲ့သော နှစ်ပေါင်း (၃၀) အတွင်း ပေါ်ထွက်လာသော Programming language များထဲတွင် အတည်ငြိမ်ဆုံးနှင့် အလိုက်သင့် ပြောင်းလဲနိုင်သော language တစ်ခု ဖြစ်သည်။

JAVA သမိုင်းအကျဉ်း

၁၉၉၁ ခုနှစ်တွင် Sun Microsystems မှ ပညာရှင်များဖြစ်ကြသော Jame Gosling, Patrick Naughton, Chris Warth, Ed Frank နှင့် Mike Sheridan တို့က JAVA ကို စတင်တွေ့ရှိခဲ့ကြသည်။ (၁၈ လအကြာတွင် ၎င်းကို ပြုပြင် မွမ်းမံ၍ ပထမဦးဆုံး version ကို ဖန်တီးခဲ့သည်။ ၎င်းကို Oak ဟု ခေါ်ခဲ့၍ ၁၉၉၅ ခုနှစ်တွင် JAVA language အဖြစ် အမည်ပြောင်းလဲခဲ့သည်။ ၁၉၉၁ ခုနှစ်မှ ၁၉၉၅ ခုနှစ်အတွင်း Oak မှ JAVA သို့ ပြောင်းလဲရာတွင် Bill Joy, Arthur Van Hoff, Jonathan Payne, Frank Yellin နှင့် Tim Lindholm တို့သည် အဓိကဆောင်ရွက်ခဲ့ကြသော ပုဂ္ဂိုလ်များ ဖြစ်ကြသည်။

JAVA ထွက်ပေါ်လာပြီးနောက် internet တွင် platform အဖြစ် သုံးစွဲရန် မည်သူ တစ်ဦးတစ်ယောက်ကမျှ မမျှော်မှန်းခဲ့ကြပါ။ အမှန်အားဖြင့် JAVA ဖြစ်ပေါ်တိုးတက်လာရန် အဓိကအချက်မှာ platform-independent language ဖြစ်၍ ဖြစ်သည်။ ၎င်းသည် microwave oven နှင့် remote control များကဲ့သို့ လျှပ်စစ်ပစ္စည်းများတွင် ထည့်သွင်းတပ်ဆင်သည့် software များ ပြုလုပ်ရာတွင် သုံးနိုင်သည်။ ထို့ပြင် JAVA သည် မတူညီသော CPU များကို controller များအဖြစ် အသုံးပြုနိုင်သည်။ C နှင့် C++ အပါအဝင် အခြား language များသည် compile, design ပြုလုပ်ရန် ပစ္စည်းအမျိုးအစားကန့်သတ်ချက်ရှိသည်။ CPU အမျိုးအစားတစ်ခုတွင် C++ program compile လုပ်ရန် ဖြစ်နိုင်သော်လည်း full C++ compiler ဖြစ်ရန် လိုအပ်ချက်များ ရှိနေဦးမှာ ဖြစ်သည်။

ၵင်းအတွက် ဢေးကြီးခြင်း၊ အချိန်ကြာမြင့်ခြင်း ပြဿနာများလည်း ရှိနိုင်သေးသည်။ ၵင်း ပြဿနာများကို ဖြေရှင်းရန်နှင့် လွယ်လွယ်ကူကူ အလုပ်လုပ်နိုင်ပြီး Platform Independent language ဖြစ်ရမည်၊ မည်သည့် operating system နှင့်မဆို အလုပ်လုပ်နိုင်စွမ်းရှိပြီး CPU အမျိုးမျိုးတွင်လည်း ၵင်းတို့၏ proceed code များ ပါဝင်ရမည်၊ မတူညီသော အခြေအနေများတွင်လည်း အလုပ်လုပ်နိုင်စွမ်းရှိမည့် language အဖြစ် JAVA ကို ဖန်တီးခဲ့ကြသည်။ JAVA ဖြစ်ပေါ်လာမှု၏ ဒုတိယအဆင့်အနေနှင့် ပိုအရေးကြီးသော အပိုင်းမှာ JAVA သည် World Wide Web တွင် အရေးပါသောအခန်းမှ ပါဝင်လာခြင်း ပင် ဖြစ်သည်။ JAVA ၏ အသုံးဝင်မှုကို programmer များ ရုတ်တရက် မသိရှိခဲ့ကြပေ။

World Wide Web ၏ အရေးပါလာမှုနှင့်အတူ Web သည် လွယ်ကူသင့်တော်သော ပရိဂရမ် လိုအပ်မှု ဖြစ်ပေါ်ခဲ့သည်။ JAVA သည် computer language design တွင် ရှေ့မှ ဦးဆောင် ဖြစ်လာသည့်အကြောင်းမှာ ဤသို့ဖြစ်သည်။ Programmer အများစုသည် စောစောပိုင်းအချိန်များက ၵင်းတို့ဆန္ဒရှိသော programming language များအား အသက်မွေးဝမ်းကျောင်းအတွက် ရှာဖွေလေ့လာရင်း အခက်အခဲများ၊ ပြဿနာများ ပိုမို များပြားလာပြီး တခြားသို့ လမ်းကြောင်း ပြောင်းသွားတတ်ကြသည်။

ကွန်ပျူတာလောကတွင် Intel, Macintosh နှင့် Unix ဟူ၍ အပိုင်းကြီး (၃)ပိုင်း ခြားထားသောကြောင့် programmer အများစုသည် မိမိတို့ ကျွမ်းကျင်ရာတွင် ကျင်လည် ဆောင်ရွက်ကြရင်း လိုအပ်ချက်များ ဖြစ်ပေါ်လာနေကြသည်။ မည်သို့ပင်ဖြစ်စေ internet Web ပေါ်ထွက်လာမှုနှင့် အတူ ၵင်းအတွက် ဆောင်ရွက်ရန် ပြဿနာများလည်း ထွက်ပေါ်လာသည်။ အချုပ်ဆိုရလျှင် internet ကို မည်သည့် ကွန်ပျူတာ အမျိုးအစား၊ Operating system အမျိုးမျိုး၊ CPU အမျိုးမျိုးတို့တွင် အသုံးပြု၍ရနိုင်သော်လည်း user များသည် တူညီသော program နှင့်သာ ၵင်း၏ platform သို့ ဝင်ရောက်နိုင်ပါသည်။

၁၉၉၃ ခုနှစ်အတွင်းက JAVA Design Team အဖွဲ့ဝင်များကို internet အတွက် code များ တီထွင်နေချိန်တွင် မမြင်နိုင်သော ထိန်းချုပ်မှုများအတွက် code များကိုလည်း ရှာဖွေတွေ့ရှိခဲ့ပြီး ပြဿနာများကို အလွယ်တကူ ဖြေရှင်းနိုင်ခဲ့ကြသည်။ Internet Programming ကို အသုံးချလိုသူများသည် JAVA ကို အနီးကပ် လေ့လာ သဘောပေါက်လာကြသည်။ Internet သည် JAVA ၏ ကြီးမားသော အောင်မြင်မှုကို ဖြစ်ပေါ်လာစေသည်။

JAVA သည် C နှင့် C++ မှ ဆင်းသက်လာသောကြောင့် အတွေ့အကြုံရှိပြီး ဖြစ်သော C နှင့် C++ programmer များအတွက် ဆက်လက် လေ့လာသင့်ပါသည်။ JAVA သည် C နှင့် C++ ကိုလည်း ပိုမိုအောင်မြင်အောင် အကူအညီပေးနိုင်ပါသည်။

ပထမဦးစွာ JAVA ကို design ပြုလုပ်ခြင်း၊ စမ်းသပ်ခြင်း၊ ပိုမိုကောင်းမွန်အောင် ပြင်ဆင်ခြင်း စသည်တို့ ပြုလုပ်သောကြောင့် JAVA Language သည် ၵင်းကို ဖန်တီးသော သူများ၏ လိုအပ်ချက်များနှင့် အတွေ့အကြုံများပေါ်တွင် မူတည်သည်။ ထို့ကြောင့် JAVA သည် programmer တစ်ယောက်၏ language လည်း ဖြစ်သည်။ ဒုတိယအဆင့်အနေဖြင့် JAVA သည် ယုတ္တိဗေဒဆန်သည်။ တတိယမှာ JAVA သည် programmer များကို

full control ပေးသည်။ သင်၏ program ကောင်းလျှင် ဆောင်ရွက်မှု ကောင်းမည်ဖြစ်ပြီး သင့် program မကောင်းလျှင် ဆောင်ရွက်မှု ကောင်းမည် မဟုတ်ပါ။

JAVA language သည် သင်ရိုးညွှန်းတမ်းနှင့် သင်ကြား၍ရသော language တစ်ခု မဟုတ်ပါ။ Professional programmer များအတွက် language ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် JAVA နှင့် C++ တွင် တူညီသော အချက်အလက်များ ပါဝင်နေမှုကြောင့် ဖြစ်သည်။ JAVA သည် internet version of C++ နှင့် တူညီသည်။ မည်သို့ပင်ဖြစ်စေ ကြီးမားသော ကွဲလွဲမှုတစ်ခု ရှိသည်။ ၵင်းမှာ JAVA တွင် လက်တွေ့ပိုင်း ဆိုင်ရာနှင့် တွေးခေါ်မှုဆိုင်ရာ ခြားနားချက်သည် သိသာ ထင်ရှားခြင်းပင် ဖြစ်သည်။ JAVA သည် C++ ၏ အရှိန်အဝါ ရှိနေသော်လည်း C++ ၏ version တိုးမြှင့်မှု မဟုတ်ပါ။ JAVA သည် C++ မှ အချို့အရာများကို တိုးမြှင့်သင့်သည့်အရာကို တိုးမြှင့်ပြီး လျှော့သင့်တာ လျှော့ချခြင်း ဖြစ်သည်။ သင်သည် C++ programmer တစ်ယောက်ဖြစ်လျှင် JAVA ကို မိမိဘာသာ လေ့လာသင့်သည်။

Internet ပေါ်တွင် ဖြန့်ခွဲအသုံးချမှုအတွက် ဦးတည် ဆောင်ရွက်ရန် program platform လိုအပ်ချက်အရ JAVA လျင်မြန်စွာ ဖြစ်ပေါ်လာသည်။ JAVA သည် C++ ဖြင့် အသုံးပြုသော Object-Oriented Paradigm ကို ပို၍ ကောင်းမွန်အောင် ဆောင်ရွက်ပေးသည်။ JAVA သည် ယခုမှ သီးခြား ဖြစ်ပေါ်လာသော language တစ်ခုမဟုတ်ပါ။ လွန်ခဲ့သော နှစ်ပေါင်းများစွာက ဖြစ်ပေါ်ခဲ့သော ကြီးမားသော ဆောင်ရွက်ချက်တစ်ခုဖြစ်သည်။ ဤအချက်တစ်ခုတည်းက computer language history တွင် JAVA ဖြစ်လာဖို့ လုံလောက်သွားပါပြီ။

JAVA as Internet Language

ရှေ့တွင် ဖော်ပြခဲ့သည့်အတိုင်း JAVA programming သည် internet programming ၏ အဓိကအကြောင်းအချက်ပင် ဖြစ်သည်။ Internet သည် JAVA ကို programming များ၏ ရှေ့ဆုံးသို့ ရောက်အောင် တွန်းပို့သကဲ့သို့ JAVA သည် internet တွင် ပြည့်ဝသော အကျိုးရလဒ်ကို ဖြစ်ပေါ်စေသည်။ JAVA သည် Object များကို ကျယ်ပြန့်အောင် ချဲ့ထွင်ပြီး cyberspace ထဲမှာ လွတ်လပ်စွာ ရွှေ့သွားနိုင်သည်။ Network တစ်ခုတွင် အလွန်အရေးပါသော object အမျိုးအစား နှစ်မျိုးမှာ server နှင့် PC ကြားတွင် အသက်ဝင်စေသော passive information နှင့် dynamic, active program တို့ပင် ဖြစ်သည်။ ဥပမာ–သင် e-mail ဖွင့်ချိန်တွင် passive data များကို မြင်ရသည်။

သင် program တစ်ခုဖွင့်ကြည့်လျှင် program code သည် ဆောင်ရွက်ခဲ့စဉ်ကအတိုင်း ပြောင်းလဲမှုမရှိသော data များ ဖြစ်သည်။ ဒုတိယ object type မှာ computer ကို ဆောင်ရွက်စေသော Program နှင့် တစ်ဆင့်ပို့နိုင်သော program တို့သည် server နှင့် client computer တို့ကို ကိုယ်စားပြုဆောင်ရွက်နိုင်စေသည်။ ဥပမာ – Program တစ်ခု

သည် server တွင် ရှိသော data များကို ကောင်းမွန်စွာ ကြည့်ရှုနိုင်သဖြင့် server တွင် အသုံးပြုသော networked program များသည် လုံခြုံမှုနှင့် လွယ်ကူမှုရှိစေရန်မှာ ပြဿနာတစ်ခု ဖြစ်နေသည်။ JAVA address သည် ထိုကိစ္စရပ်များကို applet မှ တစ်ဆင့် ပြုလုပ်ဆောင်ရွက်နိုင်သည်။

Keywords of JAVA

JAVA language သည်လည်း တခြား ဆော့ဖ်ဝဲများကဲ့သို့ ပထမဦးဆုံး keywords များ နားလည်ရန်အရေးကြီးပြီး၊ ပြီးမှ ၎င်းအားအသုံးချရန် ကြိုးစားရမည် ဖြစ်သည်။ JAVA တီထွင်မှုတွင် လွယ်ကူမှု၊ လုံခြုံမှုသည် အခြေခံအင်အားအဖြစ် လိုအပ်သော်လည်း language ၏ final form ကို ပုံဖော်ရန် အရေးကြီးပါသည်။ JAVA keywords များကို စုပေါင်း၍ ဖော်ပြပေးပါမည်။

Simple

Computer programming သည် နှစ်ပေါင်းများစွာကတည်းက တဖြည်းဖြည်း တိုးတက်ဖြစ်ပေါ်လာသောကြောင့် အလုပ်နည်းနိုင်သမျှနည်းအောင် ဆောင်ရွက်လိုသော programmer များသည် ရှုထောင့်အမျိုးမျိုးမှ ကြည့်ရှုလာကြသည်။ ၁၉၇၀ ပြည့်နှစ်လောက်က ကွန်ပျူတာသည် auto-coder အဖြစ် FORTRAN language ကို သုံးခဲ့ကြသည်။ နောက်ပိုင်း တဖြည်းဖြည်းတိုးတက်လာပြီး computer language များသည် code ရေးရန် လွယ်ကူလာရုံသာမက program အသုံးပြုရာတွင်လည်း အခက်အခဲ နည်းပါးလာသည်။ JAVA သည်လည်း အသုံးပြုရန် လွယ်ကူသော language တစ်ခု ဖြစ်လာသည်။

သင်သည် programming experience ရှိလျှင် JAVA ကို လေ့လာရန် မခက်ပါ။ သင်သည် object oriented programming ၏ အခြေခံ concept ကို နားလည်လျှင် JAVA သည် ပို၍ လွယ်ကူစေသည်။ C++ မှ မဖြေရှင်းနိုင်သော ကိစ္စအချို့ကို JAVA က ဖြေရှင်းပေးနိုင်သည်။ C, C++ နှင့် JAVA တူညီနေသောကြောင့် လေ့လာရန် အားမစိုက်ရဘဲ လွယ်ကူစေသည်။ JAVA သည် အလုပ်တစ်ခုကို အရေအတွက် အနည်းငယ်နှင့်ပင် ရှင်းလင်းစွာ တည်ဆောက်နိုင်သည်။

Object-Oriented

JAVA သည် တခြား language များကဲ့သို့ ကျစ်လျစ်သော source code ရေးရန် design ပြုလုပ်ထားခြင်းမဟုတ်။ ဘာမှမရှိသော ဟင်းလင်းပြင်တွင် လွတ်လပ်စွာ ဖန်တီးစေသည်။ Design ပြုလုပ်ပြီး ထွက်ပေါ်လာသော ရလဒ်သည် ရှင်းသည်။ အသုံးပြုရလွယ်ကူပြီး object ကိုလည်း ချဉ်းကပ်စေသည်။ လွန်ခဲ့သော ဆယ်စုနှစ်များက ဖြစ်ပေါ်လာသော Object-Software တစ်စိတ်တစ်ဒေသမှ တစ်ဆင့် ဖြစ်ပေါ်လာခြင်း ဖြစ်သည်။ JAVA

မှ Object ပုံစံသည် ရှင်းပြီး ချဲ့ထွင်ရန် လွယ်ကူသည်။ Integers များကဲ့သို့ type များသည် High-performance non-objects အဖြစ် သိမ်းထားနိုင်သည်။

Security

JAVA သည် internet တွင် ပို၍ သိသာထင်ရှားစေခြင်းမှာ အရေးကြီးသော အချက်တစ်ခု ဖြစ်သည်။ Normal program တစ်ခုအား load ဆွဲတင်သည့်အခါတိုင်း virus ဝင်ရောက်နိုင်သည့်အန္တရာယ် ရှိနေသည်။ JAVA တွင် user အများစုသည် program များအား ထပ်ခါထပ်ခါ ဆွဲမတင်ပါ။ User များသည် ၎င်းတို့၏ system ကို virus ဝင်ရောက်မှာနှင့် program ကို ထိခိုက်စေသော အခြားအချက်များကိုလည်း ကာကွယ်ပေးသည်။ Program သည် ကိုယ်ပိုင်အချက်အလက်များ ဥပမာ credit card numbers, bank account balances နှင့် passwords အား computer's local file system ၏ contents များနှင့် ရှာဖွေနိုင်သည်။

JAVA-Compatable web browser ကို သုံးသည့်အခါ သင်သည် virus နှင့် အခြားထိခိုက်စေသော အချက်များအား စိုးရိမ်စရာမလိုဘဲ JAVA applets အား စိတ်ချစွာ ဆွဲယူ အသုံးပြုနိုင်သည်။ JAVA သည် JAVA execution environment ကို JAVA program တစ်ခုတွင် ထည့်သွင်းထားရုံဖြင့် တားဆီးကာကွယ်ထားနိုင်သော်လည်း အခြား computer အစိတ်အပိုင်းများကိုမူ မတားဆီးနိုင်ပါ။ JAVA ၏ အရေးကြီးဆုံး အချက်တစ်ချက်ကတော့ applet ကို ယုံကြည်စိတ်ချစွာ အခက်အခဲမရှိ ဆွဲတင်နိုင်ခြင်းပင် ဖြစ်သည်။

Portable

ကမ္ဘာပေါ်တွင် ကွန်ပျူတာအမျိုးမျိုး၊ operating system အမျိုးမျိုးတို့ ရှိကြပြီး အများစုသည် internet ကို ဆက်သွယ်ကြသည်။ Internet ကို ဆက်သွယ်သော platform အားလုံးကို အင်အားအကြီးဆုံးဆွဲတင်မည့် ပရိုဂရမ်များအတွက် လွယ်ကူ လုံခြုံမှုရှိသော code ပြုလုပ်ရန် လိုအပ်သည်။

Robust

ရှေ့တွင် ဖော်ပြခဲ့သည့်အတိုင်း JAVA သည် platforms များစွာရှိသည့်အနက် လွတ်လွတ်လပ်လပ် အလုပ်လုပ်သော platform တစ်ခုဖြစ်သည်။ ပရိုဂရမ်သည် systems အမျိုးမျိုးတွင် စိတ်ချစွာ ဆောင်ရွက်နိုင်ရန် လိုအပ်ချက်များကို ပြည့်စုံစွာ ပြုလုပ်ထားရမည်။ JAVA ၏ design သည် အရည်အချင်းရှိသော program ပြုလုပ်ရန် အမြင့်ဆုံး ဦးစားပေးထားသည်။ နောက်တစ်ခုက JAVA သည် ပရိုဂရမ် တီထွင်ရင်း တွေ့လာသော အမှားကို ရှာဖွေပြီး ၎င်းအမှားများကို JAVA key များနှင့် ကာကွယ်ပေးသည်။ JAVA သည် တိကျစွာ ရိုက်နှိပ်ရသော language တစ်ခုဖြစ်ပြီး compile time မှာပင် သင်၏

Code ကို စစ်ဆေးသည်။ Run-time မှာလည်း စစ်ဆေးသည်။ အခြေအနေအမျိုးမျိုးအတွက် ကြိုတင် ခန့်မှန်းနိုင်သော နည်းလမ်းအား ရေးသားရန် သိရှိခြင်းသည် JAVA ၏ အဓိက အားသာချက် ဖြစ်သည်။

JAVA သည် မည်ကဲ့သို့ စွမ်းအားလွန် ဆောင်ရွက်သည် ဆိုသည်ကို အချက်နှစ်ချက်နှင့် သိသာစေနိုင်သည်။ ၎င်းမှာ Memory Management Mistakes နှင့် Mishandled exceptional Conditions (that is, run-time errors) ဖြစ်သည်။ Memory management သည် Traditional Programming Environment မှာ ခက်ခဲရှုပ်ထွေးစေသည့် အလုပ်တစ်ခု ဖြစ်သည်။ ဥပမာ C, C++ မှ Programmer တစ်ယောက်သည် dynamic memory အား လုံးကို လွတ်လပ်စွာ နေရာယူ ဆောင်ရွက်သဖြင့် တစ်ခါတစ်ရံတွင် နေရာလွဲမှား၍ တခြား memory တွင် နေရာများဆောင်ရွက်သည့် ပြဿနာရှိသည်။ JAVA သည် ဤပြဿနာ အားလုံးနီးပါးအား memory allocation နှင့် deallocation လုပ်ခြင်းဖြင့် ဖြေရှင်းပေးသည်။ JAVA သည် Deallocation အား အလိုအလျောက် ဆောင်ရွက်စေတာကတော့ unused objects များအတွက် Garbage Collection ပြုလုပ်ပေးခြင်း ဖြစ်သည်။

Multithreaded

အချင်းချင်း အပြန်အလှန် ချိတ်ဆက် ဆောင်ရွက်သော ပရိုဂရမ်များ တိထွင်၍ ကမ္ဘာ တစ်ဝန်းလုံးရှိ လိုအပ်ချက်များကို တွေ့ဆုံရရှိရန် JAVA ကို တီထွင်ခဲ့ကြသည်။ JAVA သည် အလုပ်များစွာကို တစ်ပြိုင်တည်း ဆောင်ရွက်နိုင်သော Program ရေးရန် ခွင့်ပြုသော Multi-trheaded Programming ကို အသုံးပြုနိုင်သည်။ JAVA runtime system သည် အပြန်အလှန် ဆက်သွယ်သော system ကို ချောမွေ့စွာ လည်ပတ်စေနိုင်သော လုပ်ငန်းအများအပြားကို တစ်ပြိုင်တည်း ရှင်းလင်းပြီး အားလုံးနှင့်ဆိုင်သော အဖြေကို ရရှိစေသည်။ JAVA သည် Multitasking Subsystem များတွင် ခွင့်မပြုသော Multi-threading ကို လွယ်ကူစွာ ချဉ်းကပ်နိုင်သည်။

Architecture Neutral

Programmers များ အဓိက ရင်ဆိုင်နေရသော ပြဿနာတစ်ခုသည် ယနေ့ ပရိုဂရမ် ရေးလျှင် ၎င်းအား same machine ပေါ်တွင် နောက်တစ်နေ့မှ run နိုင်မည့် အာမခံချက် မရှိခြင်းပင် ဖြစ်သည်။ 70s နှင့် 90s များက Program များသည် Y2K ပြဿနာဖြစ်ပေါ်သော ၂၀၀၀ ပြည့်နှစ်တွင် run လို့ မရသည်ကို သတိရပါ။ Operating System, Processor တို့ကို Upgrade ပြုလုပ်ခြင်း၊ Main System Resources များ ပြောင်းလဲခြင်းတို့ကို ပြုလုပ်ခြင်းဖြင့် Program function တစ်ခုကို ဖြစ်စေခဲ့သည်။ JAVA designer များသည် ဤအခြေအနေများကို ပြေလည်စေရန် JAVA Virtual Machine နှင့် JAVA language တို့ကို နိုင်နိုင်မာမာ ရွေးချယ်ဆုံးဖြတ်ခဲ့ကြသည်။ ၎င်းတို့၏ ရည်မှန်းချက်ကတော့ "while once; run anywhere, anytime, forever" ပင် ဖြစ်သည်။ ၎င်းရည်မှန်းချက်သည် အထိုက်အလျောက် ပြီးမြောက်အောင်မြင်ခဲ့ပြီ ဖြစ်သည်။

Interpreted and High Performance

ရှေ့တွင် ဖော်ပြခဲ့တဲ့အတိုင်း JAVA သည် JAVA byte code ဟု ခေါ်သော ကြားခံကိုယ်စားပြု စကားလုံးတစ်ခုအဖြစ် ပြောင်းလဲ compile ပြုလုပ်ခြင်းဖြင့် cross platform programs များကို တည်ဆောက်ပေးနိုင်သည်။ ဤ code သည် JAVA virtual machine တစ်ခုကို အသုံးပြုသော any system ပေါ်တွင် ဘာသာပြန်ပေးနိုင်သည်။ တစ်ချိန်က cross platform solutions များပေါ်မှ ကြိုးပမ်းမှုအများစုသည် ဆောင်ရွက်မှု ကုန်ကျစရိတ် များပြားစေခဲ့သည်။ အခြား Interpreted system များ (ဥပမာ– BASIC, TCL နှင့် PERL) တို့သည် ဆောင်ရွက်ရန် လိုအပ်ငွေ များပြားစေခဲ့သည်။ JAVA ကို မည်သို့ပင်ဖြစ်စေ အလွန်နိမ့်သော CPU များပေါ်တွင်ပင် ကောင်းစွာ ဆောင်ရွက်နိုင်ရန် design ပြုလုပ်ထားသည်။ ရှေ့တွင် ရှင်းပြခဲ့သည့်အတိုင်း JAVA ကို JAVA byte code များအား Just-in-time-compiler တစ်ခုကို အသုံးပြုပြီး very high performance အတွက် native machine code အဖြစ်သို့ တိုက်ရိုက် လွယ်ကူစွာ ပြောင်းလဲနိုင်ရန် တည်ဆောက်ထားသည်။

Distributed

JAVA သည် Internet သို့ ဖြန့်ခွဲရန်အတွက် တည်ဆောက်ထားသည့် အကြောင်းမှာ ၎င်းသည် TCP/IP Protocols ကို လက်ကိုင်ပြုထားသည်။ URL တစ်ခု အသုံးပြုသော resource တစ်ခုကို ချဉ်းကပ်ခြင်းသည် file တစ်ခုအား ချဉ်းကပ်ခြင်းနှင့် များစွာခြားနားမှုမရှိ။ JAVA (Oak) ၏ Origrinal Version တွင် intra-address-space messaging အတွက် features များ ပါဝင်သည်။ ၎င်းသည် မတူညီသော ကွန်ပျူတာနှစ်မျိုးပေါ်တွင် Procedure များအား အလိုအလျောက် အဆင်ပြေစွာ ဆောင်ရွက်ရန် objects များကို ခွင့်ပြုထားသည်။ JAVA သည် Remote Method Invocation (RMI) ဟု ခေါ်သော package တစ်ခုတွင် interfaces များကို မကြာခဏ ပြန်လည်သတိရတတ်သည်။ ဤ feature သည် client/server programming ကို အကျဉ်းမျှ ချဉ်းကပ်သော အဆင့်တစ်ခုကို ဖြစ်စေသည်။

Dynamic

JAVA Programming သည် Run Time မှာ Objects များကို မှန်ကန်စွာ ဖြေရှင်းနိုင်ရန် မှန်ကန်သည့် run-time type informations များကို ယူဆောင်ပေးသည်။ ဤအချက်သည် သင့်လျော်မှုနှင့် လုံခြုံမှုတစ်ခုအဖြစ် dynamically link code ကို

ဖြစ်စေသည်။ Bytecode ၏ အမှားအယွင်း ချို့ယွင်းချက် အနည်းငယ်အား running system ပေါ်တွင် ခေတ်နှင့်အညီ မြန်မြန်ဆန်ဆန် ပြုပြင်ပေးခြင်းသည် applets environment ၏ robustness ကို အရေးတကြီး ဖြစ်စေသည်။

More to come

ရှေ့တွင် ရှင်းပြခဲ့တဲ့အတိုင်း JAVA ၏ တိုးတက်မှုသည် Internet ပေါ်တွင် အသုံးဝင်မှု တစ်ခုအဖြစ်နှင့် ယခင်စနစ်များအား တော်လှန်ခဲ့ခြင်း မဟုတ်ပါ။ အစပိုင်းတွင် သိသာစွာ တိုးတက်မှု မရှိသော်လည်း နောက်ပိုင်းတွင် Internet ပေါ်မှာ အမည်တစ်ခုအဖြစ် ထင်ရှားသိသာစေခဲ့သည်။ အခြား Software System များသည် အများအားဖြင့် platform တစ်ခုအဖြစ် settle ဖြစ်ပြီးမှ တဖြည်းဖြည်း တိုးတက်လာခြင်းဖြစ်သည်။ JAVA မှာမူ တမုဟုတ်ချင်း လျှင်မြန်စွာ ပြောင်းလဲ တိုးတက်လာခြင်း ဖြစ်သည်။

JAVA 1.0 အား ဖယ်ရှားပြီး မကြာမီ JAVA 1.1 ကို JAVA designer များက တီထွင်ခဲ့ကြသည်။ JAVA 1.1 တွင် ထပ်တိုးလာသော features များသည် ရှိပြီးသား features များကို ပို၍ ကောင်းမွန်စေခဲ့သည်။ JAVA 1.1 တွင် 1.0 library မှ features များကို ပြန်လည်ပြုပြင်ထားပြီး applets များဖြင့် ဆောင်ရွက်သော events များအား အဓိပ္ပာယ်ဖွင့်ဆိုထားသော new library elements များကို ထပ်ဖြည့်ထားသည်။ JAVA 1.0 မှ သာမန် features များကိုလည်း ဖယ်ရှားပစ်သည်။ တို့ကြောင့် JAVA 1.1 သည် attribute များအား ထပ်ဖြည့်ခြင်း၊ ဖယ်ရှားခြင်း နှစ်ခုစလုံးအား ဆောင်ရွက်သည်။

ဆက်လက်တိုးတက်လာသော JAVA 2 သည်လည်း ထို features နှစ်ခုလုံးကို ပြုလုပ်သည်။ Language များ ပြောင်းလဲမှု ဖြစ်စဉ်အားလုံးတွင် JAVA ၏ ပြောင်းလဲမှုသည် ပို၍ သိသာ ထင်ရှားသည်။ အကြောင်းမှာ older browsers များသည် new features များကို သုံးသော code ပြုလုပ်ရန် မဖြစ်နိုင်ပေ။ ပြောင်းလဲမှုများ ဘယ်လိုပင်ရှိရှိ၊ နားလည်မှုရှိဖို့သာ လိုပါသည်။ Original 1.0 မှ JAVA ၏ ဖြစ်ပေါ်ပြောင်းလဲမှုကို အကျဉ်းမျှ ကြည့်ကြပါစို့။

Features Added by 1.1

ဤ version သည် JAVA ကို အရေးကြီးသည့် features အချို့ ပေါင်းထည့်ပေးသည်။ Addition အများစုသည် JAVA Library ထဲတွင် ပါဝင်လာသည်။ New Language features အနည်းငယ်လည်း ပါဝင်သည်။ ၎င်းတို့မှာ-

* **JAVA Beans** - ၎င်းတို့သည် JAVA နှင့် ရေးထားသော Software Components များ ဖြစ်သည်။
* **Serialization** - ၎င်းသည် Object တစ်ခုကို save ပြုလုပ်ရန်နှင့် restore ပြန်လုပ်ရန် ဖြစ်သည်။

* **Remote Method Invocation (RMI)** - ၎င်းသည် မတူညီသော စက်မှ အခြား JAVA Object ၏ methods များကို အသုံးပြု ကိုးကားထားသော JAVA Object တစ်ခုကို ခွင့်ပြုသည်။ ဤအချက်သည် distributed application တည် ဆောက်ရာတွင် အရေးကြီးသော လိုအပ်ချက်တစ်ခု ဖြစ်သည်။
* **JAVA Database Connectivity (JDBC)** - Vectors အများအပြားထဲမှ SQL database ကို ဆောင်ရွက်ရန် Program ကို ခွင့်ပြုသည်။
* **The JAVA Native Interface (JNI)**- ၎င်းသည် other languages နှင့် ရေးထားသော code library နှင့်အတူ interface ပြုလုပ်ရန် သင့် program ကို နည်းသစ်ပေးသည်။
* **Reflection** - ၎င်းသည် run time မှာ JAVA object ၏ fields, constructors များနှင့် methods များကို ရွေးချယ် ဆုံးဖြတ်သော process ဖြစ်သည်။
* **Security** - Security features အမျိုးမျိုးမှာ digital signatures များ၊ message digests များ၊ access control lists နှင့် key generation တို့ ဖြစ်သည်။
* **Support for 16 bit** - Built-in သည် Unicode characters များကို အသုံးပြုဆောင်ရွက်သော 16-bit character stream အတွက် အသုံးပြုသည်။
* **Event handling** - Graphical User Interface (GUI) Components များနှင့် တွဲ၍ ဖြစ်ပေါ်လာသော အဖြစ်အပျက်များမှ နည်းလမ်းတစ်ခုအဖြစ် event handling ၏ ခိုင်မာသော ပြောင်းလဲမှု ရှိသည်။
* **Inner Classes** - ၎င်းမှာ အခြား class တစ်ခုကို အဓိပ္ပာယ် သက်ရောက်စေသော class တစ်ခုကို ပြုလုပ်သည်။

Features Deprecated by 1.1

ဖော်ပြခဲ့မည့်အတိုင်း JAVA 1.1 သည် ရှေ့မှ Library elements များကို ဖယ်ရှားပစ်သည်။ ဥပမာ - original data class အများစုကို ဖယ်ရှားပစ်သည်။ ထိုဖယ်ရှားလိုက်သော features များသည် လုံးဝမျောက်သွားခြင်းမဟုတ်ဘဲ နေရာရွှေ့သွားခြင်း ဖြစ်သည်။ ယေဘုယျအားဖြင့်တော့ ဖယ်ရှားလိုက်သော 1.0 features များဟာ ဆက်ခံလာသော အမွေ code အဖြစ် JAVA မှာ အသုံးဝင်နေဆဲ ဖြစ်သည်။ သို့သော် New Application များတွင်တော့ မသုံးသင့်ပါ။

Features Added by 2

1.1 ပေါ်မှ ထပ်မံတိုးချဲ့တည်ဆောက်မှုဖြစ်သော JAVA 2 တွင် အရေးကြီးသော New features များစွာ ထပ်ပေါင်းဖြည့်သည်။ ၎င်းတို့မှာ-

* **Saving -** ၎င်းသည် JAVA မှ user interface components အစုတစ်ခု ဖြစ်သည်။ သင်သည် သတ်မှတ်ထားသော operating system တစ်ခု သို့မဟုတ် တူညီသော အခြား operating system များပေါ်တွင် စဉ်းစားတွေးခေါ်ကြည့်နိုင်သည်။ သင်ကိုယ်ပိုင် အတွေးအမြင်များဖြင့် design ပြုလုပ်နိုင်သည်။

* **Collection -** Collection များသည် groups of object များ ဖြစ်သည်။ JAVA သည် အမျိုးမျိုးသော Collection အမျိုးအစားများ (ဥပမာ– Linked list, dynamic arrays, hash tables များကို အသုံးပြုရန် အထောက်အကူပြုသည်။ Collection များသည် တူညီသော Programming Porblem များကို ဖြေရှင်းရန် နည်းလမ်းသစ်တစ်ခု ပေးသည်။

* **Security -** သာမန်လုံခြုံမှုအဆင့်ရှိ စက်များသည် JAVA Programming အတွက် သုံးရန် အသင့်ဖြစ်သည်။ Policy file များကတော့ ခွင့်ပြုချက်ဖြင့်သာ သုံးနိုင်သည်။

* **Digital Certificates –** User တစ်ဦးအား မည်သူမည်ဝါ ဖြစ်သည်ကို ဖော်ပြသည့် အစိတ်အပိုင်း ဖြစ်သည်။ ၎င်းတို့ကို electronic passport များအဖြစ် စဉ်းစားသည်။ JAVA program များသည် လုံခြုံမှုအတွက် certificates များကို အသုံးပြုနိုင်သည်။

* **Security tools -** Security tools အမျိုးမျိုးတို့သည် သင့်ကို cryptographic keys များနှင့် digital certificates များကို ဖန်တီးသိမ်းဆည်းထားရန် အသုံးဝင်ပြီး JAVA Archive (JAR) fields များကို သတ်မှတ်ပေးသည်။ JAR file တစ်ခု၏ Signiture ကို စဉ်းစားသည်။

* **Accessibility Library -** ၎င်းသည် အခက်အခဲ အတားအဆီး အမျိုးမျိုးအား ကွန်ပျူတာနှင့် ဖြေရှင်းဆောင်ရွက်ရန် features များကို ပေးသည်။

* **2D library -** JAVA 2D library သည် shapes များ၊ images များနှင့် text အတွက် Advanced features များကို ပေးသည်။

* **Drag and drop -** Drag and Drop Capabilities များသည် keyboard မှ Japanese, Chinese နှင့် Korean Characters များကို ရိုက်နိုင်သည့် Character တစ်ခုကို ကိုယ်စားပြုရန် key-strokes အစဉ်ကို အသုံးပြု၍ ဖြစ်စေသည်။

* **Support for audio files -** WAV, AIFF, AV, MIDI နှင့် RMF audio file များနှင့် play back ပြုလုပ်နိုင်သည်။

* **Support for CORBA -** Common Object Request Broken Architecture (CORBA) သည် Object Request Broken (ORB) တစ်ခုနှင့် Interface Definition Language (IDL) တစ်ခုတို့ကို ဆိုလိုသည်။ JAVA 2 တွင် ORB တစ်ခုနှင့် idltojava compiler တစ်ခုတို့ ပါဝင်သည်။ နောက်ပိုင်းမှာ IDL specification တစ်ခုမှ code ပြုလုပ်သည်။

* **New Compiler -** နေရာများစွာမှာ တိုးတက်မှုများ ဖြစ်ပေါ်လာနေသည်။ Just-In-Time (JIT) computer သည် JDK တွင် ပါဝင်သည်။

* **JAVA Virtual Machine -** ဖော်ပြချက်အများစုတွင် applets များကို ကျွမ်းကျွမ်းကျင်ကျင် ဆောင်ရွက်ရန် သုံးသော JAVA Virtual Machine တစ်ခု ပါဝင်သည်။ သို့သော် JVMS Browser သည် နောက်ဆုံးပေါ် JAVA features များကို မဆောင်ရွက်နိုင်။ JAVA Plug-In သည် ဤပြဿနာကို ဖြေရှင်းပေးသည်။ JAVA Browser ထက် ပိုကောင်းသော JAVA Runtime Environment (JRE) တစ်ခုကို သုံးရန် Brwoser တစ်ခုကို ညွှန်ပြသည်။ JRE သည် JDK ၏ Subset တစ်ခုဖြစ်သည်။ ၎င်းတွင် development environment တစ်ခုတွင်သုံးသော tools နှင့် class များ မပါဝင်ပါ။

* **Advanced Tools -** Tools အမျိုးမျိုး (ဥပမာ – JAVAE, JAVA နှင့် JAVADOC) တို့သည် တိုးမြှင့်လာသည့် JVM အတွက် Debugger နှင့် Profiles intercer တို့သည် အသုံးပြုရန် အသင့်ဖြစ်စေသည်။

Features Deprecated by 2

1.0 နှင့် 1.1 ကြားမှ deprecated feature များ ကျယ်ကျယ်ပြန့်ပြန့်မဖြစ်သော်လည်း JAVA 1.1 features များ JAVA 2 မှ ဖယ်ရှားပစ်လိုက်သည်။ ဥပမာ Threads class ၏ methods များဖြစ်သော suspend ( ), resume ( ) နှင့် stop ( ) တို့ကို newcode တွင် မသုံးသင့်ပါ။

အခန်း (၂)
JAVA နှင့် C ++

ရှေ့ပိုင်းတွင် JAVA ကို C နှင့် C ++ မှ မည်သို့ အဆင့်မြင့် ခွဲခြား ဖြစ်ပေါ်လာသည်ဆိုခြင်းကို ဖော်ပြပြီးပါပြီ။ ယခု Chapter တွင်မူ C ++ နှင့် JAVA မှ တူညီချက်၊ မတူညီချက်များကို ရှုထောင့်မျိုးစုံမှ ကြည့်ကြပါသည်။

JAVA ALIAS C++

ပထမဦးစွာ C ++ နှင့် JAVA ကို နှိုင်းယှဉ်ကြည့်ကြပါမည်။ အောက်ပါအတိုင်း အမျိုးအစား (၃) မျိုး ခွဲခြားကြည့်ရှုနိုင်သည်။

၁။ C++ တွင် မရှိသော JAVA features များ
၂။ JAVA တွင် အသုံးမပြုသော C++ features များ
၃။ ၎င်းတို့နှစ်ခု၏ တူညီသော Features များထဲမှ ကွဲလွဲချက်များ

၁။ C++ တွင် မရှိသော JAVA Features များ

C++ တွင် မရှိသော JAVA features အများအပြားရှိသည်။ ၎င်းတို့အထဲမှ အရေးကြီးဆုံး ၃ ခုမှာ Multithreading, Packages နှင့် Interfaces တို့ ဖြစ်သည်။ သို့သော် JAVA Programming Environment ကို ပိုမိုကောင်းမွန် ပြည့်စုံစေသော အခြား Features အမျိုးမျိုး ရှိပါသေးသည်။ ၎င်းတို့မှာ–

Multithreading– နှစ်ခု သို့မဟုတ် နှစ်ခုထက်မကရှိသော တူညီသော ပရိုဂရမ်များအား အဆင်ပြေစွာ လည်ပတ်စေခြင်း ဖြစ်သည်။ C++ Program တစ်ခုကို Multithreading ပြုလုပ်လိုလျှင် သင်သည် operating system functions များကို အသုံးပြု၍ Manually ပြုလုပ်ရမှာ ဖြစ်သည်။ Method နှစ်ခုလုံးသည် နှစ်ခု သို့မဟုတ် နှစ်ခုထက်ပိုသော threads များကို အဆင်ပြေစွာ execute လုပ်ရန် ခွင့်ပြုထားသော်လည်း JAVA ၏ ယူဉ်းကပ်မှုသည် ပို၍ လွယ်ကူရှင်းလင်းသည်။

Library functions– JAVA Package တစ်ခုကို တိုက်ရိုက်ဖော်ပြသော Features များသည် C++ တွင်မရှိ။ အနီးစပ်ဆုံးတူညီမှုကတော့ Header File တစ်ခုကိုသုံးသော Library Function အုပ်စုတစ်ခုသာ ဖြစ်သည်။ C++မှာ Library တစ်ခုတည်ဆောက်အသုံးပြုပုံနှင့် JAVA မှာ Package တစ်ခု တည်ဆောက်အသုံးပြုပုံမှာ လုံးဝခြားနားသည်။

Abstract class – JAVA interface သည် C++ မှ abstract class များနှင့် အချို့အစိတ်အပိုင်းများတူညီသည်။ C++ abstract class သို့မဟုတ် JAVA Interface ၏ သာဓကတစ်ခု ပြုလုပ်ရန် မဖြစ်နိုင်ပါ။ နှစ်ခုလုံးသည် တူညီသော inferface တစ်ခုဖော်ပြရန်သာ သုံးသည်။ အဓိက ခြားနားချက်ကတော့ conceptကို ပို၍ ရှင်းလင်းစွာ ကိုယ်စားပြုဖော်ပြသော interface တစ်ခုပင် ဖြစ်သည်။

Memory Allocation – JAVA မှာ Memory Allocation အား ပြေပြစ်စွာ ချုပ်ကပ်မှုရှိသည်။ C++ သည် အသုံးပြုသော Keywords အသစ်ကို မဖျက်နိုင်ပါ။ JAVA သည် objectတစ်ခု၏ နောက်ဆုံး Referenceကို ဖျက်သည့်အခါ ၎င်း object သည် သူကိုယ်တိုင် အလိုအလျောက် အချိန်တစ်ချိန်တွင် ပျက်သွားပြီး Garbage Collection ဖြစ်ပေါ်သည်။

API Library–JAVA သည် C++ Standard Libraryကို ဖယ်ရှား၍ ၎င်းကိုယ်ပိုင် API Classes အစု တစ်ခုအား အစားထိုးသည်။ ပုံသဏ္ဌာန်နှင့် လက်တွေ့ဆန်သော တူညီမှု ရှိသော်လည်း names နှင့် parameters များတွင် သိသာ ထင်ရှားသော ခြားနားမှုများရှိသည်။ JAVA API Library တွင်ပါဝင်သော အရာအားလုံး သည် object-oriented ဖြစ်ပြီး C++ library တွင်မူ အစိတ်အပိုင်း တစ်ခုမျှသာ object-oriented ဖြစ်သည်။ Library Routines များတွင် ခြားနားမှု ရှိသည်။

Break & Continue– Break and Continue statementsများသည် Javaအား ထင်ရှားသော မှတ်တိုင်တစ်ခုအဖြစ် သတ်မှတ်လက်ခံလာစေပါသည်။

Character Type–JAVAမှ Character Type သည် 16 bit-wide Unicode characters များဖြစ်သည်။ ၎င်းတို့သည် C++ type နှင့် တူညီသည်။

>>> operator – JAVAမှာ ညာဘက်သို့ ပြသော >>> operator ထပ်တိုးသည်။

Documentation Comment – တစ်ကြောင်းနှင့် တစ်ကြောင်းထက်ပို သော မှတ်ချက်များရှိသည်။ Javaမှ မှတ်ချက်ပုံစံမှာ အစတွင် /** တစ်ခုနှင့် အဆုံးတွင် */ တစ်ခုဖြင့် ဖော်ပြသည်။

String–JAVAတွင် Built-in String Typeတစ်ခု ပါဝင်သည်။ String ပုံစံသည် C++မှ Standard String Class typeနှင့်တူညီသည်။ သင်၏ပရိုဂရမ်တွင် C++ class declarationပါဝင်လိုလျှင် C++stringကိုသာ အသုံးပြုရန်သင့်သည်။ ၎င်းသည် built- in Type မဟုတ်ပါ။

၂။ JAVA တွင် အသုံးမပြုသော C++ features များ

Java သည် C++ features များကို အသုံးမပြုပါ။ အချို့ အခြေအနေများတွင် သီးသန့်ဖြစ်သော C++ features များသည် Java နယ်ပယ်နှင့် ဆက်စပ်မှု မရှိ၍ Designerများသည် C++တွင် ပါဝင်သော featuresများ ထပ်ခါထပ်ခါ ပါဝင်နေမှုကို

C နှင့် C++နှစ်ခုစလုံးတွင် အသုံးပြုသည်။ Standard library functions အများစုသည် C++ တွင် သုံးသည်။ C based function များသည် argument ကို အသုံးပြုရန် argumentတစ်ခု၏ addressလိုအပ်သည်။ Functionအတွင်းတွင် argumentသည် ၎င်း၏ pointerကို အသုံးပြု၍ ရောက်ရှိသည်။

To provide more efficient means of implementing certain constructs especially array indexing

Array ကဲ့သို့ တည်ဆောက်မှုများတွင် ပို၍ကောင်းမွန်အောင် စွမ်းဆောင်နိုင်သည်။ ဥပမာ–၎င်းသည် array index ထက်ပို၍ pointer သုံးသော array ကို ပို၍အစဉ်လိုက်ရွေ့လျား ဆောင်ရွက်နိုင်သည်။ Moden Compilersများကို ပို၍စွမ်းရည် မြင့်စွာ သုံးထားလျှင် pointerများသည် အရည်အသွေးမြင့်စွာ ဆောင်ရွက်နိုင်သည်။ ထို့ကြောင့် arraysများကို ချုဉ်းကပ်ရန် pointersများကို သုံးခြင်းသည် C++ codeတွင် ပို၍ စွမ်းဆောင်ရည်ကောင်းသည်။

To support memory allocation

C++ တွင် memory နေရာယူမှုသည် address တစ်ခု (ဥပမာ–pointer ကဲ့သို့) ဖြင့် နေရာယူသည်။ ၎င်း addressကို pointer variable အဖြစ် သတ်မှတ်သည်။ JAVA တွင် Object တစ်ခုသည် ထို objectနေရာယူမည့် reference အဖြစ် နေရာယူသည်။ ထို referenceသည် ကျစ်လျစ် သည်။ JAVA reference variable သည် new operatorအဖြစ် နေရာယူသော Objectကို တိကျစွာ ညွှန်ပြသော်လည်း C++ Pointers များအတိုင်း မဆောင်ရွက်နိုင်ပါ။ ၎င်းတို့သည် JAVA runtime Context ၏အပြင်ဘက် memoryကို မညွှန်ပြနိုင်ပါ။

To provide access to any arbitray machine address possibly to call a ROM routine or to read/writ directly to memory

JAVAသည် Pointerကဲ့သို့ အချို့ ဆောင်ရွက်ချက်များကို ခွင့်မပြုပါ။ သင်သည် applicationsများကိုရေးလျှင် ၎င်း system resource ကို ဆောင်ရွက်သော native code routinesကို ပြန်လည်ရရှိသော JAVA'S native capabilities များကို သုံးနိုင်သည်။

Pointer သုံးထားသော C++ code အား JAVA သို့ ပြောင်းလဲသည့် ဥပမာ နှစ်ခုအား နမူနာအဖြစ် ကြည့်ပါ။

Converting Pointer Pararmeters

Cartesian coordinatesတစ်စုံကို သိမ်းထားသော Coord Object ၏ လက္ခဏာကို ပြောင်းလဲသော C++ Programတစ်ခုကို ကြည့်ကြစို့။ ၎င်းတွင် functionဖြစ်သော

reverse sign ()သည် ပြောင်းလဲမည့် coord objetကို pointerတစ်ခုဖြင့် ကြေညာသည်။ သင်သိပြီးဖြစ်သည့်အတိုင်း C++ ၏ *, &, –> စသည့် pointer operators များကို Operation အတွက် သုံးသည်။

```
//Reverse the signs of a coordinate C++ version
#include <iostream>
using namespace std;
class Coord {
public:
   int x;
   int y;
};
// Reverse the sign of the coordinates
void reverseSign (Coord *ob) {
   ob->x = -ob->x;
   ob->y = -ob->y;
}

int main ( )
{
   Coord ob;

   ob.x = 10;
   ob.y = 20;

   cout << "Original values for ob: ";
   cout << ob.x << ", " << ob.y << "

   reverseSing ($ob);

   cout << "Sign reversed values for ob: ";
   cout << ob.x << ", "<< ob.y << "\n";

   return o;
}
```

ဒင်း Programအား Java versionသို့ ပြောင်းနိုင်သည်။ ပြောင်းလဲမှု အများစုတွင် C++pointer operatorsများကို အသုံးမပြုဘဲ ဖယ်ထားသည်။ JAVAသည် Operators များကို referencer ကြေညာသည်နှင့် ရှင်းလင်းသော porameters အဖြစ်သို့ ပြောင်းသွားသည်။

```
// Reverse the signs of a coordinate Java version
class Coord {
   int x;
   int y;
};
```



```
Class DropPointers {
   // Revrese the sign of the coordinates.
   static void reverseCoord (Coord ob) {
      ob.x = -ob.x;
      ob.y = -ob.y;
   }
   public static void main (String args [ ] ) {
      Coord ob = new Coord ( ) ;

      ob.x = 10;
      ob.y = 20;

   System.out.printIn("Original values for ob: "+
      ob.x + ", " + ob.y) ;
   reverseCoord(ob);
   System.out.printIn("Signreversed values for ob:
      " + ob.x + " , " + ob.y) ;
   }
}
```

Program ၂ ခုလုံးမှ အဖြေသည် တူညီသည်။

```
Orignial values for ob:  10,  20
Sign reversed values for ob:  -10,   -20
```

Converting Arrays Based Pointers

Pointerအခြေခံသော array ဆောင်ရွက်မှုသည် အနည်းငယ် ခက်ခဲသည်။ အကြောင်းမှာ ရိုးရိုး C++ coding style သည် ပို၍ ရှုပ်ထွေး သိပ်သည်းသော ဖော်ပြချက်များကို ပေးသည်။ ဥပမာ– aray တစ်ခုမှ အကြောင်းအရာကို အခြားတစ်ခုသို့ ကူးယူသော short Program တစ်ခုတွင် array ၏ အဆုံးမှာ အမှတ်အသား "0" ကို သုံးသည်။

```
// Copy an array in C++ using pointers.
#include <iostream>
using namespace std;

int main ( )
{
   int nums [ ] = {10, 12, 24, 45, 23, 19, 44,
                   88, 99, 65, 76, 12, 89, 0};
   int copy[20];

   int *p1, *p2; // integer pointers

   // copy array
   p1 = nums; // p1 points to start of nums array
   p2 = copy;
```

```
    while (*p1) *p2++ = *p1++;
    *p2 = 0 ; // terminate copy with zero

    // Dispay contents of each array.
    cout << "Here is the original array : \n";
    p1 = nums;
    while (*p1) cout << *p1++ << "  ";
    cout << endl;
    cout << "Here is the copy: \n";
    p1 = copy;
    while(*p1) cout << *p1++ << " ";
    cout << endl;

    return 0;
}
```

ဒင်းသည် C++ code အတွက် အလွန်ရိုးသော်လည်း while (*P1) *P2 ++ = * P1 ++; ကို ကြည့်လျှင် ဒင်း၏ ဆောင်ရွက်ချက်ကို နားလည်အောင် စဉ်းစားချိန် လိုအပ်သည်ကို တွေ့ရမည်။ JAVA၏ အကျိုးကျေးဇူးကတော့ ဤကဲ့သို့ မဆောင်ရွက်ပါ။ JAVA Version Program ကို ကြည့်လျှင် ရည်ရွယ်ချက်နှင့် ရလဒ်ကို မြင်နိုင်ပါသည်။

```
// Array copy without pointers using Java.
class CopyArray
   public static void main (String args[ ]) {
    int nums [ ] = {10, 12, 24, 45, 23, 19, 44
                    88, 99, 65, 75, 12, 89, 0}
    int copy[ ] = new int (14};
    int i;

    // copy array
    for(I = 0; nums[i] ! = 0; i++)
      copy [i] = nums [i];
    nums [i] = 0; // terminate copy with zero

    // Display contents of each array.
    System.out.printIn ("Here is the original array;");
    for (I = 0; numsli] ! = 0; i++)
      System.out.print (nums {i] + " ");
      System.out.printIn();

      System.out.printIn("Here is the copy:");
      for(I = 0; nums [i] ! = 0; i++)
            System.out.print (copy[i] + " ");
      System.out.printIn();
   }
}
```


Program နှစ်ခုလုံးမှ အောက်ပါ result ကို ရရှိပါသည်။

```
Here is the original array:
10 12 24 45 23 19 44 88 99 65 76 12 89
Here is the copy:
10 12 24 45 23 19 44 88 99 65 76 12 89
```

Converting C++ Abstract Classes into JAVA Interfaces

JAVA ၏ အထူးခြားဆုံးတစ်ခုမှာ interface ဖြစ်သည်။ ဒင်းသည် အသေးစိတ် ဖော်ပြခြင်း မဟုတ်ဘဲ ယေဘုယျ ပုံစံကိုသာ ဖော်ပြသည်။ အဆင့်တစ်ခုစီမှ interface တစ်ခုသည် ဒင်း interface ဖြင့် ဖော်ပြသော နည်းစနစ်များအတိုင်း ဆောင်ရွက်သည်။ ထို့ကြောင့် JAVA မှာ interface တစ်ခုသည် class တစ်ခု၏ ယေဘုယျပုံစံဟု အဓိပ္ပာယ် ဖွင့်ဆိုနိုင်သည်။ ထို့အပြင် interface သည် JAVA မှ poly morphism အတွက် အထောက်အပံ့ပေးနိုင်သည်။

C++ မှ Abstract classes များသည် JAVA မှ abstract class များနှင့် တူသည်။ ဒင်းတို့တွင် အသေးစိတ် ဖော်ပြချက်များ ပြည့်စုံစွာ မပါဝင်ပါ။ C++ မှ abstract class တစ်ခုတွင် အနည်းဆုံး virtual function တစ်ခုပါဝင်သည်။ Virtual function အစစ်တစ်ခုသည် ဖော်ပြချက်များ မပါဝင်ပါ။ ရှေ့ပြေး function တစ်ခုကိုသာ အဓိပ္ပာယ်ဖွင့်ဆိုသည်။ ထို့ကြောင့် C++မှ virtual function အစစ်တစ်ခုသည် JAVA မှ abstract method တစ်ခုကဲ့သို့ အရေးပါသည်။

C++ တွင် abstract classes များသည် JAVA မှ interface များနှင့်တူညီသော function တစ်ခုကို ပြုလုပ်သည်။ C++ abstract classes အားလုံးသည် JAVA Interfaces များအဖြစ်သို့ မပြောင်းနိုင်သော်လည်း အများစုမှာ ပြောင်းနိုင်သည်။ Integer list တစ်ခု၏ ပုံစံကို ဖော်ပြသော Intlist ဟုခေါ်သော abstract class တစ်ခုကို အသုံးပြုထားသော C++ program တစ်ပုဒ်ကို ကြည့်ကြစို့။ ဤ class အား integer list တစ်ခုကို ဖော်ပြရန် array တစ်ခုအသုံးပြုထားသော intarray ဖြင့် တည်ဆောက် ထားသည်။

```
// A C++-style abstract class and its implementation.
#include <iostream>
#include <cstdlib>
using namespace std;

// An abstract class that defines the form of an
   integer list.
class IntList {
public:
   virtual int getNext() = 0; //pure virtual functions.
   virtual void putOnList (int i) = 0;
};
```

```
// Create an implementation of an integer list.
class IntArray : public IntList {
    int storage [100] ;
    int putIndex, getIndex;
publc:
    IntArray () {
    putIndex = 0;
    getIndex = 0;
}
// Retrurn next integer in list.
int getNext () {
    if (getIndex >= 100) {
        cout << "List Underflow";
        exit (1) ;
    }
    getIndex++;
    return storage [getIndex-1];
}

// Put an integer on the list
void putOnList (int i) {
    if (putIndex < 100) {
        storage [putIndex] = i;
        putIndex++;
    }
    else {
        cout << "List Overflow";
        exit (1) ;
    }
}

int main ()
{
    IntArray nums;
    int i;

    for (i = 0; i<10; i++) nums.putOnList(i);

    for(i = 0; i<10; i++)
        cout << nums.getNext() << endl;

    return 0;
}
```

ဤပရိဂရမ်မှာ abstract class Intlist သည် integer list တစ်ခုကိုသာ အဓိပ္ပာယ်ဖွင့်ဆိုသည်။ ၎င်းတွင် virtual functionစစ်စစ်တစ်ခုတည်းသာပွင့်ပြီး အခြား

data များကို မဖော်ပြပါ။ ဤအခြေအနေများအတွက် program အား JAVA သို့ ပြောင်းလိုက်လျှင် interface တစ်ခုအဖြစ် ပြောင်းလဲနိုင်သည်။

```
// Here, IntList is made into an interface which
InArray implements.
// Define interface for an integer list.
interface IntListIF {
    int getNext ();
    void putOnList (int i);
}

// Create an implementation of an integer list.
class IntArray implements IntListIF{
    private int storage [ ];
    private int putIndex, getIndex;

    IntArray () {
        storage = new int [1001];
        putIndex = 0;
        getIndex = 0;
}

// Create an implementation of an integer list.
public int getNext () {
    if (getIndex >= 100) {
        System.out.printIn("List Underflow");
        System.exit(1);
    }
    getIndex++;
    return storage[getIndex - 1];
}

//Put an integer on the list.
public void putOnList (int i) {
    if (putIndex < 100) {
        storage [putIndex] = i;
        putIndex++;
    }
    else {
        System.out.printIn("List Overflow");
        System.exit(1);
    }
}

class ListDemo {
    public static void main (String args[ ]) {
```

```
    IntArray nums = new IntArray () ;
    int i;

  for (I = 0; I < 10; i++) nums.putOnList(i);

  for(I = 0; I < 10; i++1
    System.out.printIn(nums.getNext ());
  }
}
```

သင်မြင်သည့်အတိုင်း: C++ abstract class မှ Intlist နှင့် JAVA Interface IntlistIF တို့ကြားတွင် တစ်ခုနှင့်တစ်ခု ဆက်သွယ်မှုသည် နီးစပ်သည်။ ၎င်းတွင် Intlist မှ IntlistIF ပြောင်းလျှင် ပိုသင့်လျော်သည်။ အကြောင်းမှာ Intlist တွင် virtual functions တစ်မျိုးတည်းသာ ပါဝင်သောကြောင့် ဖြစ်သည်။ ၎င်းမှာ အဓိကအချက် ဖြစ်သည်။ C++ code မှ JAVA သို့ ပြောင်းလိုလျှင် only pure virtual functions ပါဝင်သော abstract classes များပါဝင်သော example များကို ရှာဖွေပါ။ သို့သော် functions သို့မဟုတ် data အရေအတွက် အနည်းငယ်သာ ပါဝင်သော abstract C++ classes များ အတွက်တော့ မလိုပါ။

C++ကို interface construct တစ်ခုသို့ ပြောင်းလဲနိုင်ခြင်း မရှိလျှင် C++ programmers များအနေဖြင့် အသေအချာ စဉ်းစားရန် လိုပါမည်။ တစ်ခါတစ်ရံမှာ ခိုင်မာသော member တစ်ခုသည် abstract class တစ်ခုတွင် ပါဝင်နေတတ်သည်။

ဥပမာ အောက်ပါ abstract C++ class ကိုကြည့်ကြစို့။

```
// An abstract C++ class.
class SomeClass {
   bool isOK;
public:
   virtual int f1 () = 0;
   virtual void f2 (int i) = 0;
   virutal double f3 () = 0;
   virtual int f4 (int a, char ch) = 0
};
```

JAVA interface တစ်ခုသို့ ပြောင်းလဲစေနိုင်သော တစ်ခုတည်းသောအချက်မှာ isok ပါဝင်မှုပင် ဖြစ်သည်။ class နှင့် ဆက်စပ်သော အချို့နေရာများကို ညွှန်ပြရန် isok ကို အသုံးပြုသည်ဟု မှတ်ယူရသည်။ မည်သို့ပင်ဖြစ်စေ သင်သည် ၎င်းကို တွေးကြည့်လျှင် isok အတွက် variable တစ်ခုအဖြစ် ဖော်ပြရန် အမှန်တကယ် reason မရှိပါ။ အမှန်မှာ သင်သည် status ကိုပြသော isok ( ) ဟုခေါ်သည့် method တစ်ခုဖော်ပြနိုင်သည်။ isok ( ) ကို အခြားဖော်ပြချက် class များဖြင့် အခြားနည်းလမ်းများသုံး၍ ဖော်ပြမည်။ ထို့ကြောင့် သင်သည် ရှေ့မှ C++ abstract class ကို အောက်ပါအတိုင်း: JAVA interface သို့ ပြောင်းနိုင်သည်။


```
interface SomeClass {
   int f1 () ;
   void f2 (int i) ;
   double f3 () ;
   int f4 (int a, char ch);
   boolean isoK();
}
```

C++ မှ abstract classes အများစုအား JAVA သို့ Code ပြောင်းလဲ၍ interfaces ပြောင်းနိုင်သည်။ ထိုသို့ ပြောင်းရာတွင် Class hierachy ၏ structure ကို ဖော်ပြသော code ကို ရှာဖွေထားရမည်။

Converting Default Argoments

JAVA တွင် အသုံးမပြုသော်လည်း C++ ၏ features အဖြစ် ကျယ်ပြန့်စွာ အသုံးပြုသော အချက်တစ်ခုမှာ default function arguments ဖြစ်သည်။ ဥပမာအားဖြင့် အောက်ပါ C++ program တွင် ဖော်ပြထားသော area ( ) function သည် argument နှစ်ခုပါဝင်သော ထောင့်မှန်စတုဂံ တစ်ခု၏ ဧရိယာ သို့မဟုတ် argument တစ်ခုပါဝင်သော စတုရန်းတစ်ခု၏ ဧရိယာကို တွက်ချက်သည်။

```
// C++ program that uses default arguments.
#include <iostream>
using namespace std;

/* Compute area of a rectangle. For a square, pass only one argument.
*/
double area (double 1, double w = 0) {
   if (w == 0) return 1 * 1;
   else return 1 * w;
}

int main ()
{
   cout << "Area of 2.2 by 3.4 rectangle: ";
   cout << aea (2.2, 3.4) << endl;
   cout << "Area of 3.0 by 3.0 by 3.0 square: ";
   cout << area (3.0) << endl;
   return 0;
}
```

သင်တွေ့ရှိသည့်အတိုင်း: argument တစ်ခုတည်းသာ ပါဝင်သော area ဖြစ်သောအခါ second defaults သည် 0 ဖြစ်သည်။ Function တွင် ထောင့်မှန်စတုဂံ၏ အလျားနှင့်အနံ နှစ်ခုစလုံးအတွက် first argument ကိုသာ သုံးသည်။ default arguments များသည် မရှိမဖြစ်လိုအပ်သည်။ အဓိကမှာ default arguments များသည် parameters များစွာမှ

ဆွဲယူသော action of shorthand form တစ်ခုသာ ဖြစ်သည်။ ထို့ကြောင့် တစ်ခုသို့မဟုတ် တစ်ခုထက် ပိုသော default arguments များပါဝင်သော C++ function တစ်ခုအား JAVA သို့ ပြောင်းရန် case တစ်ခုစီကို handle လုပ်နိုင်သော overloaded methods များကို ပြုလုပ်ရမည်။ ဥပမာ argument နှစ်ခုပါသော area ( ) version တစ်ခုနှင့် argument တစ်ခုသာပါသော အခြား version တစ်ခု လိုအပ်သည်။ ရှေ့ program အား Java သို့ ပြင်ရေးသည့်အခါ အောက်ပါအတိုင်း ဖြစ်မည်။

```
// Java version of area program.
class Area {
   // Compute area of a rectangle.
   static double area (double 1, double w) {
      if (w==0) return 1 * 1;
      else return 1 * w;
   }

   // Overload area ( ) for a square.
   static double area (double 1) {
      return 1 * 1;
   }
   public static void main (String args [ ]) {
      System.out.printIn("Area of 2.2 by 3.4 rectangle:
        " + area (2.2, 3.4) );
      System.out.printIn ("Area of 3.0 by 3.0 square:
         area (3.0) );
   }
}
```

Reference Parameters of C++ And JAVA

Pointer parameter တစ်ခုမှ JAVA မှာ reference parameter တစ်ခု ဖြစ်လာပုံကို သင်မြင်ပြီးပြီ။ C++ သည်လည်း reference parameters များကို သုံးသည်။ C++ code မှာတွေ့ရသော pointer parameters အများစုသည် C မှ ဖြစ်ပေါ်လာခြင်း ဖြစ်သည်။ New C++ code အားလုံးသည် ၎င်းကိုယ်တိုင် argument အား ချဉ်းကပ်ရန် function တစ်ခု လိုအပ်သောအခါ reference parameters များကို သုံးလိမ့်မည်။ JAVA နှင့် C++ နှစ်ခုစလုံးသည် reference parameters များကို သုံးသည်။ ဤကဲ့သို့ ပုံစံမျိုးသည် အမြဲတမ်းဖြစ်လေ့ရှိသော case မဟုတ်ပါ။ ပိုမိုနားလည်ရန် အောက်ပါ C++ program အား regerence parameters သုံးသော Coord objects နှစ်ခု၏ contents များအဖြစ် ပြောင်းလဲပုံကို ကြည့်ကြစို့။

```
// Swap coordinates - C++ version.
#include <iostream>
class Coord {
public:
   int x;
   int y;
};

// Swap contents of two Coord objects.
void swap (Coord &a, Coord &b) {
   Coord temp;

   // Swap contents of objects
   temp = a;
   a = b;
   b = temp;
}

int main ( )
{
   Coord ob1, ob2;

   ob1.x = 10;
   ob1.y = 20;
   ob2.x = 88;
   ob2.y = 99;

   cout << "Original values:\n";
   cout << "ob1: "<< ob1.x << ", " << ob1.y << "/n";
   cout << "ob2: "<< ob2.x << ", " << ob2.y << "/n";
   cout "\n";
   swap (ob1, ob2);

   cout << "Swapped values : \n";
   cout << "ob1: "<< ob1.x << ", " << ob1.y << "\n";
   cout << "ob2: "<< ob2.x << ", " << ob2.y << "\n";

   return 0;
}
```

၎င်း program မှ အောက်ပါ အဖြေကို ရရှိသည်။ သင်မြင်သည့်အတိုင်း ob1 နှင့် ob2 contents များ ပြောင်းသွားသည်။

```
Original vales:
ob1 : 10, 20
ob2 : 88, 99
Swapped values:
ob1 : 88, 99
ob2 : 10, 20
```

JAVA မှာ objects အားလုံးသည် object reference variable တစ်ခုမှတစ်ဆင့် ချဉ်းကပ်သည်။ ထို့ကြောင့် object တစ်ခုသည် method တစ်ခုသို့ ရောက်ရှိသည့်အခါ ၎င်း၏ reference တစ်ခုတည်းကိုသာ ရောက်ရှိသည်။ ဤအချက်သည် objects အားလုံး JAVA method တစ်ခုသို့ reference ဖြင့် အလိုအလျောက် ရောက်ရှိသည်ကို ဆိုလိုသည်။ အမှန်ဖြစ်ပျက်မှုများကို နက်နက်နဲနဲမစဉ်းစားဘဲ တစ်စုံတစ်ယောက်သည် ရှေ့မှ program ၏ conversion အား အောက်ပါအတိုင်း မှားယွင်းစွာ ကြိုးစားလိမ့်မည်။

```
// Swap program incorrectly converted to Java.
class Coord {
   int x;
   int y;
};
Class SwapDemo {
   static void swap (Coord a, Coora b) {
     Coord temp = new Coord ( ) ;
   // this won't swap contents of a and b!
   temp = a;
   a = b. ,
   b = temp;
}
public static void main (String args [ ]) {
   Coord ob1 = new Coord ( );
   Coord ob2 = new Coord ( );

   ob1.x = 10;
   ob2.x = 20;

   ob2.x = 88;
   ob2.y = 99;

   system:out.prifitIn ("Original values:");
   System.out;printIn ("ob1: "+
      ob1.x + ", " + ob1.y);
   System.out.printIn("ob2: " +
      ob2.x + ", " + ob2.y + "\n");

swap (ob1, ob2);

System.out.printIn ("Swapped values:");
   System.out.printIn ("ob1: " +
      ob1. x + ", " + ob1.y);
   System.out.printIn ("ob2: " +
      ob2.x + ", " + ob2.y + "\n");
   }
```



```
}
```

ဤမှားယွင်းနေသော program မှ အောက်ပါအဖြေကို ရရှိပါမည်။

```
Original values:
ob1: 10, 20
ob2 : 88, 99
```

သင်တွေ့ရှိရသည့်အတိုင်း main ( )မှ ob1 နှင့် ob2 ၏ တန်ဖိုးများသည် ပြောင်းလဲမသွားပါ။ ပထမပိုင်းတွင် အနည်းငယ် မှန်သည်ဟု ထင်ရသော်လည်း object reference တစ်ခုသည် Method တစ်ခုသို့ မရောက်ပါ။ Java သည် call-by-value ကို အသုံးပြု၍ Arguments အားလုံးကို Method များသို့ ရောက်ရှိစေသည်။ Argument တစ်ခုကို copy ပြုလုပ်ရာတွင် method ၏ အတွင်း copy သည် method ကို ခေါ်ရန် သုံးသော arguments ပေါ်မှာ မသက်ရောက်နိုင်။ ဤအခြေအနေသည် object reference case ကို အနည်းငယ် မထင်မရှား ဖြစ်စေသည်။ ၎င်းကို ပြင်ရန် swap ( )ကို objects များ၏ contents များကို ပြောင်းလဲစေသောအချက်အား ပြန်ပြင်ရေးရန် ဖြစ်သည်။

```
// COrrected version of swap ().
static void swap (Coord a, Coord b) {
   Coord temp = new Coord () ;
   // swap contents of objects
   temp.x = a.x;
   temp.y = a.y;
   a.x = b.x ;
   a.y = b.y;
   b.x = temp.x;
   b.y = temp.y;
}
```

အဖြေအမှန်ကို ရရှိနိုင်ပါပြီ။

Converting Multiple-Inheritance Hierachies of C++

C++ သည် class တစ်ခုမှ တစ်ချိန်တည်းမှာ နှစ်ခု သို့မဟုတ် နှစ်ခုထက်ပိုသော classes များကို ရရှိဖြစ်ပေါ်စေသည်။ JAVA မှာ မဆောင်ရွက်နိုင်။ C++ သည် များစွာသော အမွေကို ပေးနိုင်ပြီး JAVA မှ မပေးနိုင်။ C++ application မှ JAVA သို့ ကူးပြောင်းမှုများကို သင်သိရှိပြီး ဖြစ်သည်။ အခြေအနေတိုင်း တွင် ခြားနားမှုများရှိပြီး ယေဘုယျနှစ်မျိုး ရှိနိုင်သည်။ ပထမတစ်ခုသည် case အတော်များများတွင် multiple inheritance များသည် C++ program တွင် အလုပ်လုပ်သော်လည်း အမှန်တကယ် မလိုအပ်ပါ။ ၎င်း case မျိုးအား single-inheritance hierachy တစ်ခုသို့ class structure ပြောင်းနိုင်သည်။ ဥပမာ– House ဟု ခေါ်သော class တစ်ခုကို တည်ဆောက်ထားသော C++ class hierachy တစ်ခုကို ကြည့်ကြစို့။

```
class Foundation {
   // ...
} ;
class Walls {
};

class Rooms {
   // ...
class House : public Foundation, Walls, Rooms {
   // ...
};
```

House မှာ Foundation, Walls, Room ဟူ၍ အမျိုးမျိုး ခွဲဖြာထွက်နေသည်။ ၎င်းမှာ မလိုအပ်ဘဲ များနေသည်။ JAVA အတွက် တူသော class အစုတစ်ခု တည်ဆောက်ကြပါမည်။

```
class Foundation {
   // ...
} ;
class Walls extends Foundation {
   // ...
} ;
class Rooms extend Walls {
} ;
class House extends Rooms {
   // ...
} ;
```

ပထမ class တစ်ခုအား class တစ်ခုစီ တဖြည်းဖြည်း ချဲ့ထွင်၍ နောက်ဆုံး House ကို ရရှိသည်။ တစ်ခါတရံမှာ multiple inheritanse hierachy တစ်ခုသည် နောက်ဆုံး object တွင် multiply inherited classes များ၏ objects များ ပါဝင်နေခြင်းဖြင့် ပို၍ ပြောင်းလဲရန် လွယ်ကူသည်။ ဥပမာ JAVA တွင် House ကို အခြားနည်းဖြင့် တည်ဆောက်ကြစို့။

```
class Foundation {
   // ...
} ;
class Walls {
   // ...
} ;
class Rooms {
   // ...
} ;
/* Now, House includes Foundation, Walls, and Rooms as object
members.
*/
```



```
class House {
   Foundation f;
   Walls w;
   Rooms r;
   // ...
}
```

ဤနေရာမှာ foundation, Walls နှင့် Rooms များသည် House မှ ခွဲထွက်ခြင်းထက် House ၏ အစိတ်အပိုင်း objects များ ဖြစ်သည်။ အခြားသတိပြုရမည့် အချက်တစ်ခုမှာ တစ်ခါတစ်ရံ C++ program တစ်ခုမှ ရိုးရိုး multiple - inheritance hierachy တစ်ခုပါဝင်ပါမည်။ အကြောင်းမှာ အားနည်းသော initial design ကြောင့် ဖြစ်ပါသည်။ သင့် JAVA သို့ ကူးပြောင်းသောအခါမှ ဤချို့ယွင်းချက် အနည်းငယ်ကို ပြင်ဆင်ရန် ဖြစ်သည်။

Destructors Versus Finalization

C++ မှ JAVA သို့ ပြောင်းသောအခါ ပိုမိုအရေးကြီးသော အချက်တစ်ခုကတော့ C++ destructor တစ်ခုနှင့် JAVA finalize() method တစ်ခုကြား၊ တွင် ခြားနားချက်ကို ရှင်းဆိုင်ကြရမည်။ ၎င်းတို့၏ ဆောင်ရွက်မှုများစွာတွင် တူညီကြသော်လည်း အမှန် ဆောင်ရွက်ရာတွင် ထူးခြားသော ကွဲလွဲချက်များ ရှိနေသည်။ C++ destructor နှင့် JAVA finalize() method ၏ ရည်ရွယ်ချက်နှင့် အကျိုးကျေးဇူးတို့အား ပြန်လည် လေ့လာကြည့်ကြစို့။ C++ မှာ object တစ်ခု scope ၏ အပြင်ဘက်ရောက်သွားသည်နှင့် ပျက်စီးသွားသည်။ ၎င်းပျက်စီးမှု၏ အဓိကမှာ destructor function ကြောင့် ဖြစ်သည်။ ဤသည်မှာ hard-and-fast rule တစ်ခုပင် ဖြစ်ပါသည်။ အခြားမရှိပါ။ ဤ rule ၏ အစိတ်အပိုင်း တစ်ခုစီကို အနီးစပ်ဆုံးကြည့်ကြစို့။

Scope ၏ အပြင်ဘက်ရောက်သွားသောအခါ object တိုင်း ပျက်သွားသည်။ ထို့ကြောင့် သင်သည် function အတွင်းမှာသာ local object တစ်ခုကို ကြေညာသင့်သည်။ ထို Function ကို ပြန်ခေါ်သောအခါ local object သည် အလိုအလျောက် ပျက်သွားသည်။ Function parameters များနှင့် functions များမှလာသော objects များအတွက် အတူတူပဲ ဖြစ်သည်။

မဖျက်ဆီးခင် object ၏ destructor ကို ခေါ်သည်။ ဤအချက်သည် အခြား program statements များ မဆောင်ရွက်မီ ရုတ်တရက် ဖြစ်တတ်သည်။ ထို့ကြောင့် C++ destructor တစ်ခုသည် deterministic fashion တစ်ခုတွင် အမြဲ ဆောင်ရွက်သည်။ သင်သည် destructor တစ်ခု၏ မည်သည့်အချိန် မည်သည့်နေရာတွင် ဆောင်ရွက်သည် ဟုတာကို သိနေရပါမည်။

C++ constructor ၏ deterministic behaviour နှင့် finalization ၏ ဖြစ်နိုင်သော မူတောင့်အမျိုးမျိုးသည် နေရာ အတော်များများမှာ တိုက်ဆိုင်ကြပြီး အခြားနေရာများမှာလည်း

တဲ့ပြဿနာမျိုးများ ရှိသည်။ ဥပမာ- အောက်ပါ C++ program ကို ကြည့်ပါ။

```
// This C++ program can call f () indefinitely.
#include <iostream>
#include <cstdlib>
using namespace std;

const int MAX = 5;
int count = 0;

class X {
public:
   // constructor
   X () {
   if (count <MAX) {
      count++;
   }
   else {
      cout << "Error - can't construct";
      exit (1) ;
   }
}
   // destructor
   -X() {
      count-;
   }
};

void f ()
{
   x ob; // allocate an object
   // destruct on way out
}

int main ()
}
   int i

   for(i = 0; i < (MAX*2); i++) {
      f()
      cout << "Current count is: " << count << endl;
   }
   return 0;
}
```

Output သည် အောက်ပါအတိုင်း ဖြစ်ပါမည်။

```
Current count is : 0
```



```
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
Current count is : 0
```

X အတွက် constructor နှင့် destructor ကို သေချာစွာကြည့်လျှင် canstructor inerement များသည် MAX ထက် နည်းသည့်အချိန်တွင် count တိုးသွားမည်။ destructor သည် count ကို လျော့စေသည်။ ထို့ကြောင့် x object တစ်ခုပြုလုပ်သည့်အခါ count သည် တိုးပြီး x object ဖျက်သည့်အခါ လျော့သည်။ သို့သော် MAX objects သည် တစ်ကြိမ်ထက်ပိုပြီး မဆောင်ရွက်နိုင်။ မည်သို့ပင် ဖြစ်စေ main (), f () မှာ MAX * 2 times များတွင် error တစ်ခု ဖြစ်စေသည်။ ဒါကြောင့် f () အတွင်းမှ cunt ကို တိုးစေမည့် x ၏ object တစ်ခုကို ပြုလုပ်ပါမည်။ ပြီးမှ function ကို ပြန်ခေါ်ရမည်။ အချို့ဖြစ်ရပ်များသည် object များ scope ၏ ပြင်ပသို့ ရုတ်တရက်ထွက်သွားပြီး destructor မှ counts များကို လျော့ချခြင်းကြောင့် ဖြစ်သည်။ ထို့ကြောင့် f () ကို ခေါ်သောအခါ count သည် မည်သည့်တန်ဖိုးမျှ မပြခြင်း ဖြစ်သည်။ ဤပုံစံမျိုးသည် အကန့်အသတ်မရှိ ဖြစ်စေနိုင်သည်။ မည်သို့ပင်ဖြစ်စေ JAVA သို့ ပြောင်းသောအခါ ဤအချက်များ မတွေ့ရတော့ပါ။

ရှေ့ Program အား JAVA ဖြင့် ရေးကြည့်ကြစို့။

```
// This Java program will fail after 5 calls to f().
class X {
   static final int MAX = 5;
   static int count = 0;
   // constructor
   X () {
      if(count<MAX) {
         count++;
}
   }
   else {
      System.out.printIn("Error-can't construct");
      System.exit(1);
   }
}
   //finalization
   protected void finalize() {
      count-;
   }
```

```
    static void f()
    {
       x ob = new X(); // allocate an object
       / destruct on way out
    }
    public static void main (String args[ ]) {
       int i
       for(i=0; i < (MAX*2); i++) {
         f() ;
          Systems.out.printIn(*Current count is: *+ count);
       }
    }
}
```

Function ကို ၅ ကြိမ်ခေါ်ပြီးနောက် အောက်ပါ output ကို ရရှိမည်။

```
Current count is : 1
Current count is : 2
Current count is : 3
Current count is : 4
Current count is : 5
Error -can't construct
```

ပရိဂရမ်၏ အားနည်းချက် ရှိရခြင်း အကြောင်းကတော့ garbage collection သည် f( ) returns အကြိမ်တိုင်းမှာ မဖြစ်ပွားနိုင်ပါ။ ထို့ကြောင့် finalize( )သည် မတူညီနိုင်ဘဲ count ၏ တန်ဖိုးသည် မလျော့ဘဲ ဖြစ်မည်။ Method ကို ၅ ကြိမ်ခေါ်ပြီးနောက် count သည် maximum value သို့ ရောက်ရှိပြီး program သည် fail ဖြစ်သွားသည်။ garbage collection ဖြစ်ပေါ်မှုများသည် သီးခြားရပ်တည်သော ဖော်ပြချက် မဟုတ်နိုင်သဖြင့် ၎င်းကို အလေးထားဆောင်ရွက်ရန် အရေးကြီးသည်။ ဤအခြေအနေမျိုးသည် C++ version မှ ရှေ့ program ကဲ့သို့ program များတွင်လည်း အချို့ platforms များတွင်လည်း Java တွင် ဖြစ်စေနိုင်သည်။ မည်သို့ပင်ဖြစ်စေ C++ မှာ ဆိုလျှင် သင်သည် destructor တစ်ခုကို မည်သည့်အချိန်၊ မည်သည့်နေရာမှာ ခေါ်ရမည်ဆိုတာ သိနိုင်ပြီး JAVA တွင်မူ finalize( )ကို မည်သည့်အချိန် မည်သည့်နေရာမှာ ဆောင်ရွက်မလဲဆိုတာ သင် မသိနိုင်ပါ။ ထို့ကြောင့် C++ မှ JAVA သို့ code ပြောင်းသောအခါ သင်သည် destructor တစ်ခု၏ မည်သည့်အကြောင်းအရာပေါ်တွင် တည်မှီပြီး ဆောင်ရွက်သည်ဆိုသည်ကို ဖြစ်ပေါ်မှု အကြိမ်များမှ သာဂေမျှားဖြင့် စောင့်ကြည့်ရမည်။ C++ discussion ပြီးဆုံး၍ ယခု Java ၏ အပြည့်အဝဆောင်ရွက်မှု discussion ကို ဆက်လက်လေ့လာကြစို့။

အခန်း (၃)
Java Programming

ယခုအခန်းတွင် Simple Java Programming အကြောင်းနှင့် Java Programs များရေးရန်အတွက် လေ့လာကြပါမည်။ ၎င်းတို့ကို compile ပြုလုပ်ရန်နှင့် run ရန်လည်း လေ့လာကြမယ်။

Writing The program

အားလုံးသိပြီးဖြစ်တဲ့အတိုင်း ရေကူးခြင်းကို တတ်လိုလျှင် အကောင်းဆုံးမှာ ရေကူးကန်ထဲသို့ ခုန်ချခြင်းပင် ဖြစ်သည်။ ထို့အတူပဲ ကျွန်ုပ်တို့ဟာ Java နှင့် Program တစ်ပုဒ် ရေးကြပါသည်။ ပြီးမှ အမျိုးမျိုးသော ပြောင်းလဲချက်များကို side by side လေ့လာပါမည်။ အခြား software များကဲ့သို့ Java မှာလည်း Java ဟုခေါ်သော ကိုယ်ပိုင် extension ရှိသည်။ ထို့ကြောင့် Java Program တစ်ခုစီမှာ Java extension တစ်ခုစီ ရှိပါလိမ့်မည်။

Test java program တစ်ပုဒ်ကို စတင်ရေးကြည့်ကြပါစို့။

```
Class test1
{
   public static void main (string argst[])
   {
   System.out.printIn("Sachin Tendulkar);
   }
}
```

ဤပရိဂရမ်မှ Sachin Tendulkar ဆိုသည့် စကားလုံးကို print လုပ်မည့် ပရိဂရမ် ဖြစ်သည်။ ၎င်းကတော့ Print ပြုလုပ်သည့် အမျိုးအစားဖြစ်သည်။ ပြီးမှ Computer Language အဖြစ်ပြောင်းပြီး မည်ကဲ့သို့ Compile လုပ်သည်ကို လေ့လာပါမည်။ Program တစ်ခုစီတွင် data နှင့် Processing ရှိသည်။ ရှေ့မှ Program ကမူ data Processing မရှိပါ။ Print out အတွက်သာ ဖြစ်သည်။

Processing မရှိသော်လည်း computer မှ ဆောင်ရွက်ရန် instruction ရှိသည်။ Processing Method definition မှာ { နှင့်စပြီး } နှင့်ဆုံးသည်။ ထို့အတူပဲ class definition သည်လည်း { နှင့်စပြီး } နှင့် ဆုံးသည်။

Reserved Words

Words အားလုံးကိုတော့ အသုံးမပြုနိုင်ပါ။ အချို့ words များသာ Java ၏ own use အဖြစ် အသုံးပြုနိုင်ပါသည်။ အောကတွင် Java၏ reserverd word များကို ဖော်ပြထားပါသည်။

Java သည် Upper Case, Lower Case အား ခွဲခြား ဆက်ဆံသည်။ Class အား class သို့မဟုတ် CLASS ဟု ဖော်ပြလျှင် ခွင့်မပြုပါ။ သင် Class သို့မဟုတ် CLASS ဟု ရေးလျှင် Complier က ၎င်းတို့ကို Ordinary Word အဖြစ်သာ လက်ခံဆောင်ရွက်ပြီး reserved word class အဖြစ် အသုံးပြုရန် သင်၏ ရည်ရွယ်ချက်ကို ဆောင်ရွက်ပေးမည် မဟုတ်ပါ။

အောက်တွင် reserve words များကို ပြည့်စုံစွာ ဖော်ပြထားသည်။

abstract	boolean	break	byte	case	cast
catch	char	class	const	continue	default
do	double	else	extends	final	finally
float	for	future	generic	goto	if
implements	import	inner	instanceof	int	interface
long	native	new	null	operator	outer
package	private	protected	public	rest	return
short	static	super	switch	synchronized	
this	throw	throws	transient	try	var
void	volatile	while.			

Identifiers

တစ်စုံတစ်ခုအား ဖော်ပြရန် အသုံးပြုသော words များ ရှိကြပါသည်။ ရှေ့ပရိုဂရမ်မှ class name တစ်ခုအဖြစ်သုံးသော tests စကားလုံးသည် computer programming ၏ ဝေါဟာရ ဖော်ပြချက်တစ်ခု ဖြစ်သည်။ Java မှ ဖော်ပြချက်တစ်ခုသည် letter တစ်ခု၊ underscore (-), dollar sign ($) တစ်ခုတို့နှင့် စတင်နိုင်သည်။ ကိန်းဂဏန်းများသည် ဖော်ပြချက်များ၏ အတွင်းတွင်သာ ပါဝင်နိုင်သည်။

@	#	%	"	&	*	:	+	-
=	<<	>>	;	,	I	!	'	"
'	0	]	0	a	~	|		

စသော symbols များသည် Java တွင် reserved ထားသော operators များဖြစ်သည်။ Cryptic symbols များကိုမူ ရှောင်သင့်ပါသည်။

Format of the Program

အခြား Programming Language များနှင့် ရင်းနှီးပြီးသားသူများအဖို့တွင် Program Format အကြောင်းကို သိပြီး ဖြစ်ပါလိမ့်မည်။ Java သည် Free Format Language ဖြစ်သည်။ Compila သည် Java Program တစ်ခုကို ဖတ်သည့်အခါ လိုင်းတစ်လိုင်းမှာ စကားလုံး အရေအတွက်ကို ဂရုမစိုက်ပါ။ Extra blank များကိုလည်း လျစ်လျူရှုသည်။ Tab character ကိုလည်း blank တစ်ခုအဖြစ်သာ မှတ်ယူသည်။ punctuation symbols များ ဖြစ်သည့် ; , { ,) တို့သည် words များ မဟုတ်ပါ။

```
class test1
{
public static void main (String args [ ])
   {System.out.printIn("Sachin Tendulkar"); }
class test1 {public static void main (string args [])
   {system.out.printIn("sachin Tendulkar");}}
class test1
   public static void
main(String
   args[ ))
{
   System.out.printIn("Sachin Tendulkar")
) )

      class test1 {
   public static void main(sring args[  ])
{
   System.out.printIn("Sachin Tendulkar")
     ; ) )
class
test1
{
public
static
void
main
(
string
args
[
]
)
{
System
.
out
.
```

```
printIn
(
"Sachin Tendulkar"
)
;
}
}
```

ဤပရိုဂရမ်တစ်ခုစီကို compile, run ပြုလုပ်သည့်အခါ result သည် Sachin Tendulkar တစ်မျိုးတည်းသာ ရရှိမည်။ အဓိကကတော့ သင့် program အား မည်သူမဆို ကြည့်ရုံနှင့် နားလည်နိုင်ရန်ပါ။ Double quotes ကြားမှာ text string ဟု ခေါ်သည်။ ၎င်းကို single word တစ်ခုအဖြစ်သာ မှတ်ယူသည်။ Sring တစ်ခုသည် လိုင်းများကွဲနေလျှင် error ဖြစ်မည်။

Adding Comments To The Program

ရှေ့မှ ပရိုဂရမ်အား Comments များ ထပ်ဖြည့်ကြည့်ကြစို့။

```
class test1
{
   public static void main(String args [ ])
   {
   // This is a single line comment.
   System.out.println("Sachin Tendulkar");
   /* Between these
      two indicators you
      can write anything.    ....*/
   }
}
```

Program ၏ resultant output တွင် ၎င်း comment line များကို ထည့်သွင်းစဉ်းစားမှု မပြုပါ။ // ၏ နောက်မှ စာသားများအား ထိုလိုင်းအဆုံးထိကို comment ဟုခေါ်ပြီး ၎င်းကို Compiler မှ လျစ်လျူရှုသည်။ ထို့ပြင် /* နှင့် */ ကြားမှ စာသားများသည်လည်း comment တစ်ခုဖြစ်သည်။ ပထမ Comment သည် short comment တစ်လိုင်းသာ ဖြစ်ပြီး နောက်တစ်ခုမှာ big discriptions ဖြစ်သည်။ Single line comment (//)သည် multiline comment (/*....*/)အတွင်းတွင် ပါဝင်နိုင်သည်။

```
/*  something // anything */
```

သို့သော် ပြောင်းပြန်အနေနှင့် /* */ သည် // အတွင်း ပါဝင်နေမှု

```
// anything  /* something  */ everyting
```

(သို့မဟုတ်)

```
// something  /*  anything
```



```
else  */
```

ကိုတော့ ခွင့်မပြုပါ။ Single-line comment တစ်ခုအား အခြား single-line comment တစ်ခုအတွင်း ပါဝင်နေမှုကိုလည်း ခွင့်ပြုပါသည်။

```
// anything  // everything
```

သို့သော် ၎င်းအား ဖော်ပြလေ့မရှိ၊ multi-line comment တစ်ခုသည် အခြား multi-line comment အတွင်း ပါဝင်မှုကို ခွင့်မပြုပါ။

```
/* anything
   /* everything
else
   */
something
   everything  */
```

Comments အားလုံးကတော့ program ဖတ်သူနားလည်အောင် ဖော်ပြပေးခြင်းပါပဲ။ Machine နှင့် compiler ကတော့ ၎င်းတို့ကို အသုံးမပြုပါ။ သို့သော် ရှေ့သင်ခန်းစာတွင် ဖော်ပြခဲ့သည့်အတိုင်း Java ကို ပို၍ ကျယ်ကျယ်ပြန့်ပြန့် တိုးတက်လာပြီး နှစ်အနည်းငယ်အတွင်းမှာ Java Program ကို အခက်အခဲမရှိ နားလည်စေရန် good comments များ ဖော်ပြထားသင့်ပါသည်။

Compilation The Program

Java Programs များကို ပထမ Compile လုပ်ပြီးမှသာ run သည်။ Compiled Program မှ result ရရှိပြီးမှ လိုအပ်လျှင် program အား ပြန်လည်ပြင်ဆင်ပြီး နောက်ထပ် Compile ပြုလုပ်နိုင်သည်။

Cycle of Edit- Compile- Run

Java ကို သင့်တော်သော directory တစ်ခုတွင် သင့် system ပေါ်သို့တင်သောအခါ java, javac, appletviewer, javadoc, javah, javap, jdb စသည့် အမည်ရှိသော file များသည် system commands များအဖြစ် store ပြုလုပ်သည်။ သင် test 1 java program ကို ရေးပြီးသည့်နောက် အောက်ပါ command ကို ရိုက်ပါ။

```
javac  test  1.  java
```

ပြီး enter နှိပ်ပါ။ System သည် (Java compiler အတွက် အတိုကောက်စာလုံး: javac) compilation ပြုလုပ်ပြီး test 1 class ဟုခေါ်သော တခြားဖိုင်တစ်ခုတည်ဆောက်ပါလိမ့်မည်။ ဤဖိုင်သည် java process တစ်ခုလုံးအတွင်းမှ အလယ်အဆင့်ဖြစ်သည်။ ပြီးလျှင် နောက်ထပ် command တစ်ခုရိုက်ရပါမည်။

```
java . test1
```

ဒီနေရာတွင် မှတ်ရန်မှာ command သည် javap သာ ဖြစ်ပြီး javac သို့မဟုတ် java တို့ မဟုတ်ပါ။ နောက်တစ်ခုမှာ test 1 သာရိုက်ပါ။ Test 1 class သို့မဟုတ် test 1 java စသည်ဖြင့် မရိုက်ရပါ။ Case တွင် သင် အမှားပြုလုပ်လျှင် machine မှ လာပြမယ်။ ၎င်းအား Monitor မှ instructions များအတိုင်း ဆောင်ရွက်ပါ။ Upper/lower cases များကိုလည်း ဂရုစိုက်ရပါမည်။ Javac ရိုက်ရမည့်နေရာမှာ Javac သို့မဟုတ် JAVAC ဟု ရိုက်မိပါက system အတွက် ကြီးမားသော အမှားတစ်ခု ဖြစ်သွားပါလိမ့်မည်။

test. java file မှ program ကို ကြည့်ကြစို့။

```
class test1
{
   public static void main(String args [ ])
   {
   System.out.printIn("Sachin Tendulkar"); }
   }
}
```

ထိုပရိုဂရမ်အား အောက်ပါ command ဖြင့် compile လုပ်နိုင်ပါသည်။

```
javac  test 1 . java
```

သင့် directory မှာ test 1 class ဟုခေါ်သော ဖိုင်တစ်ဖိုင် ဖြစ်သွားပါမည်။ program အား run ပြုလုပ်ရန် အောက်ပါ command ကို ရိုက်ပါမည်။

```
java  test1
```

Screen မှာ အောက်ပါ result လာပြပါမည်။

```
Sachin  Tendulkar
```

Sachin Tendulkar နေရာတွင် အခြားအမည်များ အစားထိုး၍ program အား ပြင်ရေးနိုင်ပါသည်။

```
   class test1
{
   public static void main(String args [ ] )
   {
   System.out.printIn("Rahul Dravid" );
   }
}
```

Javac test 1 java ကို သုံး၍ compile ပြုလုပ်၍ tests class အား java test 1 သုံး၍ run ခြင်းဖြင့် အောက်ပါအဖြေကို ရပါမည်။

```
Rahul Dravid
```

Class name အား test 2 သို့ ပြောင်းရေးပါမည်။ output တွင် ပြောင်းလဲမှုကို

မြင်ရန် Mohan ထပ်ထည့်ပါ။

```
   class test2
{
   public static void main(string args [ ])
   {
   System.out.printIn("Rahul Mohan Dravid");
   }
}
```

ဤပရိုဂရမ်အား compile, run ပြုလုပ်နိုင်ပါသည်။ ၎င်းသည် directory ထဲတွင် test 2 class အဖြစ် file သိမ်းသည်။ ဖိုင်အဟောင်း test class လည်း ရှိသည်။ Test 1 class အား run လျှင် old output ရရှိမည်ဖြစ်ပြီး test 2 class အား run ပါက Rahul နှင့် Dravid ကြားတွင် Mohan ပါဝင်လာမယ်။

```
Rahul  Mohan  Dravid
```

Debugging of the Program

သင်သည် ပရိုဂရမ်တစ်ပုဒ် မရေးနိုင်သေးလျှင် ရှေ့မှ program ကိုသာ run ကြည့်ပါ။ ပရိုဂရမ်အကောင်းဆုံးမဖြစ်ခင် ၂ ကြိမ်၊ ၃ ကြိမ် run ရန်လိုသည်။ ယခု သင်ပြုလုပ်နိုင်သော အမှားအမျိုးအစားများနှင့် သူတို့ကို ဘယ်လိုပြင်မလဲဆိုသည်ကို လေ့လာကြပါမည်။ ကျန်အမှားများကို ကွန်ပျူတာ ဝေါဟာရအရ bug ဟု ခေါ်ပြီး bug တစ်ခုအား ပြင်ဆင်ခြင်း သို့မဟုတ် ဖယ်ရှားခြင်းကို debugging ဟု ခေါ်သည်။ နည်းအမျိုးမျိုးဖြင့် bugs များကို ရှာဖွေတွေ့ရှိနိုင်သည်။ ယေဘုယျအားဖြင့် Java compiler သည် case အများစုတွင် bugs များကို ထောက်ပြသည်။ ရှေ့မှ program အား ပြန်ကြည့်ကြရအောင်။

```
class test1
   {
   public static void main(String args [ ] )
   {
   System.out.printIn
      ("Rahul Mohan Dravid";
   .}
}
```

ပထမလိုင်းတွင် typical mistake တစ်ခုဖြစ်သည်။ Program အား javac သုံး၍ compile ပြုလုပ်သောအခါ error တစ်ခု ရရှိမည်။

```
test. java:1:class or interface declaration  expected.
{
^
1  error
```

Error message မှာ file name, line number စသည်ဖြင့် ပြသည်။ ပြီး error ကို ထောက်ပြသည်။

အခြား မှားလေ့ရှိသော အမှားတစ်ခုမှာ statement အဆုံးတွင် semi colon (;) မေ့ခြင်းပင် ဖြစ်သည်။

```
class test1
{
   public static void main(String args[ ] )
   {
   System.out.printIn("Rahul Mohan Dravid")
   }
}
```

Error တစ်ခုပေးပါမည်။

```
test.1. java: 6: ';' expected.
   System. out.  println ("Rahul  Mohan  Dravid")
^
1  error
```

အခြားအမှားတစ်ခုကတော့ closing parenthesis (}) မေ့ကျန်ခြင်း ဖြစ်မည်။

```
class test1
{
   public static  void main(String args[ ])
   {
   System.out.printIn("Rahul Mohan Dravid")
}
```

Error message မှာ

```
test 1.java:4: '}' expected
}
^
1  error
```

အခြားမှားလေ့ရှိသော အမှားတစ်ခုမှာ string အဆုံးတွင် closing double quote မေ့ခြင်း ဖြစ်သည်။

```
   class test1
{
   public static void main(String args [ ])
   {
   System.out.printIn("Rahul Mohan Dravid)
   }
}
```


အောက်ပါ error message ပေးပါမည်။

```
test1.java:3: String not terminated at end of line.
   (
   System.out.printIn("Rahul Mohan Dravid) ;
   (
      ^
test1.java:4: ')' expected
}
^
1 errors
```

String မှာ closing မပါသည်နှင့် string ၏ အစိတ်အပိုင်းများ ဖြစ်သော လိုင်း ၄ မှ) နှင့် } တို့သည်လည်း error လာပြပါမည်။ ဤဥပမာအရ error တစ်ခုသည် program ၏ ကျန်သည့် နေရာများတွင်လည်း သက်ရောက်နိုင်သည်။

သင်ပြုလုပ်နိုင်တဲ့ errorများနှင့် ၎င်းတို့ကို ဘယ်လိုပြုပြင်ရမလဲဆိုသည်ကို ယခု သင် သိပါပြီ။

DATA VARIABLES

Programထဲမှာ ပေးနိုင်တဲ့ dataအကြောင်းကို ခဏထားရပါဦးမည်။ Dataသည် computer ထဲမှာ သိမ်းထားနိုင်သလို different arithmetic operations များ အသုံးပြု၍လည်း တွက်ချက်နိုင်သည်။ သိမ်းထားသော numbers များကို ကွန်ပျူတာ ပေါ်ဟာရတွင် variable ဟု ခေါ်သည်။ Variable သည် ကွန်ပျူတာမှာ ဆောင်ရွက်မှု မှာ variableတစ်ခုသည် valueတစ်ခုသိမ်းထားနိုင်သော computer's memoryတွင် နေရာတစ်ခုပဲ ဖြစ်သည်။ ဤ value သည် ၎င်းကို ထပ်ခါထပ်ခါ တွက်ချက်ခြင်းဖြင့် ပြောင်းနိုင်သည်။ Name တစ်ခုအား variable တစ်ခုပေးနိုင်ပြီး ဤ name နှင့် ရည်ညွှန်းအသုံးပြုနိုင်သည်။

Variable name တစ်ခု သတ်မှတ်ရွေးချယ်ရန် Java မှ ဆောင်ရွက်ပုံသည် identifiersရွေးချယ်သတ်မှတ်ခြင်းနှင့် တူညီသည်။ Variable nameကို ရွေးပြီးနောက် store ပြုလုပ်ရမည်။ Variable ၏ value typeကို ကြေညာရမည်။ ၎င်းကို type of variable ဟု ခေါ်သည်။ type ဟုလည်းခေါ်သည်။ သင်္ချာဘာသာရပ်တွင် numbers များ၏ types 2 မျိုး ရှိသည်။ Integer (such as 0, 1, 2, -1, -2, -3, etc) နှင့် Real numbers (2.33, -35. 8967680954 , [illegible] စသဖြင့်) တို့ ဖြစ်သည်။ Java မှာ ပထမအမျိုးအစားကို int (integer ၏ အတိုကောက်အမည်) ဟု လည်းကောင်း၊ ဒုတိယ အမျိုးအစားကို doubleဟု ကြေညာသည်။ စက်အတွင်းတွင် ဤနှစ်မျိုးစလုံးကို ခြားနား မှု သိမ်းသည်။ Doubleသည် intထက် spaceပိုယူသည်။ ထို့ကြောင့် number ကို ကြေညာလျှင် intနှင့် doubleကို တိကျစွာ ကြေညာရပါမည်။ xသည် int number တစ်ခုဖြစ်ပြီး yyသည် double တစ်ခုဖြစ်လျှင် ၎င်းတို့အား

```
int x;
double yy;
```

ဟု ကြေညာရပါမည်။

ASSIGNMENTS

သင်သည် int variable y အား value 5 assign လုပ်ချင် လျှင်

```
y = 5 ;
```

ဟု ရေးရပါမည်။

မှတ်ရန်မှာ 'equal to' sign (=) သည် သခ်ျာမှ ပုံစံနှင့် မတူပါ။ ဤ operator သည် right hand side (rhs) မှ value အား left hand side (lhs) တွင် ရောက်ရှိစေခြင်း ဖြစ်သည်။ side ပြောင်းလဲ၍

```
5 = y;
```

ဟု ရေးလျှင် compilerမှ နားမလည်ဘဲ ဤ command ကို ဖယ်ထုတ်ပစ်လိမ့်မည်။

```
y = 5 + 6;
```

ဟုလည်း ရေးနိုင်သည်။ ၎င်းသည် rhs တွင် တွက်ချက်ပြီး value 11 သည် lhs တွင် ရောက်ရှိမည်။ Variable value ကို screen မှ ပြနိုင်သော program ရေးလျှင်

```
class test1
   {
   public static void main (String args [ ] )
   {
   int y;
   y = 5 + 6;
   System.out.println("y is " + y) ;
   }
}
```

output မှာ y is 11 ဖြစ်ရမည်။

အပေါ်မှ println statement တွင် အမျိုးအစား-နှစ်မျိုးအား print ပြုလုပ်သည်။ string "y is" ပြီးနောက် string "11" ကို ပြသည်။ ၎င်းတွင် *strings နှစ်ခုကြား၌ '+ ' sign တစ်ခု string တစ်ခုတည်းပြုလုပ်ပြီး ၎င်းအား system.out. println Command ဖြင့် print ပြုလုပ်ပေးသည်။

Strings နှစ်ခုအား Java တွင် + sign ဖြင့် ပေါင်းနိုင်သည်။ သခ်ျာသဘောတွင် numbers နှစ်ခုပေါင်းရန်အတွက်လည်း symbol +ကို သုံးသည်။ ရွပ်ထွေးစွာ ဖော်ပြမှုမျိုး မပြုလုပ်သင့်ပါ။ + sign သေးမှ ဖော်ပြချက်များကို ကြည့်လျှင် numbers များ ၏ + နှင့် strings များ၏ + တို့အား ပေါင်းစပ်မှုကို ရှင်းလင်းပါစေ။ အောက်ပါ program တွင် ပို၍ ရှင်းသည်။


```
class test1
   {
   public static void main(String args [ ])
   {
   int y;
   y = 5 + 6;
   System.out.println("y is " + y + y);
      }
   }
```

output မှာ-

```
y is 1111
```

ဖြစ်ပါမည်။

Computer သည် argument 3 ခု အား တစ်ခုပြီးတစ်ခု Print ပြုလုပ်သည်။ y +y အားပေါင်းလျှင် 22 ဖြစ်စေလျှင် print line command ကို ပြန်ရေးရပါမည်။

```
System.out.println ("y is" + (y+y)); ၎င်းသည် 22 ဖြစ်ပြီး string
```

ဖြစ်လာပါမည်။ ၎င်းအား

```
System. out . println ( " y is " + "22");
```

ဖြင့် ဖော်ပြနိုင်ပြီး output မှာ

```
y is 22
```

ဖြစ်သည်။

Variable Values

Variable တစ်ခုတွင် store လုပ်ထားသော value သည် ၎င်းအား အမျိုးအစား number တစ်ခုအဖြစ်သတ်မှတ်ပြီး ၎င်းကို ရည်ညွှန်းသောအခါ အလိုအလျောက်ဖတ်သည်။ ဥပမာ သင်သည်

```
x = 7 ;
```

ကို assign ပြုလုပ်ထားလျှင်

```
y = 5 + x ;
```

ဟု ရေးပါ။

ပြီးလျှင် rhs မှ x တွင် ၎င်း၏တန်ဖိုး 7 အစားထိုးမည်။ y သည် value 12 ရရှိမည်။ ၎င်းပြင် x အား lhs နှင့် rhs နှစ်ဖက်စလုံးတွင် တူညီစွာ ထားရှိနိုင်သည်။

```
x = x + 3 ;
```

Lhs မှ x သည် သင် assig ပြုလုပ်ရမည့် value ဖြစ်ပြီး rhs မှ x သည် x ၏ current value ဖြစ်သည်။ အရေးကြီးတဲ့ တစ်ချက်ကတော့ current value ဖြစ်သည်။ assignments တော်တော်များများမှာ current value ၏ အမျိုးမျိုးပြောင်းလဲနိုင်ပြီး ၎င်းကို ထိန်းသိမ်းထားရန် လိုသည်။

ဥပမာ-

```
class test1
    {
    public static void main(String args [ ])
    {
    int y;
    y = 7;
    System.out.printIn("y now is " + y);
    y = y + y
    System.out.printIn ("y now is " + y);
    y = y + y
    System.out.printIn ("y now is " + y);
    y = y + y
    System.out.printIn ("y now is " + y);
    }
}
```

Values များ အမျိုးမျိုးရရှိပါမည်-

```
x now is 7
x now is 14
x now is 21
x now is 28
```

Swapping Values of Variables

Programming တွင် ပြောင်းလဲမှုကို ထိန်းသိမ်းထားသော variable ၏ Current value ကို သင်မေ့ပြီး variable နှစ်ခုအား တန်ဖိုးပြောင်းလဲခြင်းသည် အနည်းငယ် ရှုပ်ထွေးသည်။ x နှင့် y variable နှစ်ခုအား ပြောင်းလဲစေသော ပရိဂရမ်အား comments များနှင့် ရှင်းပြပါမည်။

```
class test1
    {
    public static void main(string args [])
    {
    int x, y; // You can combine
       // declarations of the same type.
    x = 3;          // Assignment
    y = 4;          // Assignment
    x = y;          // Swapping
    y = x;          // Swapping
    System.out.printIn("x is " + x);
    System.out.printIn("y is " + y);
    }
}
```


output သည်

x is 4

y is 4 ဖြစ်သည်။

y သည် 3 မဖြစ်ဘဲ x နှင့် y နှစ်ခုစလုံး 4 ဖြစ်နေသည်။ x = y ပြုလုပ်တဲ့အခါ x ၏ original value က ပျောက်သွားသည်။ ၎င်းကိုကာကွယ်ရန် program အား အောက်ပါအတိုင်း ပြင်ရေးသင့်သည်။

```
class test1
    {
    public static void main(String args [ ])
    {
    int x, y, temp;        // New variable temp
    x = 3;          // Assignment
    y = 4;          //Assignment
    temp = y;       //Swapping;
       // preserving original y value
    y = x;          // Swapping
    x = temp;
       // Swapping; using stored value
    System.out.printIn("x now is " + x);
    System.out.printIn("y now is " + y);
    }
}
```

output မှာ-

x now is 4

y now is 3 ဖြစ်သည်။

Decision Making

Storing နှင့် Printing အကြောင်း တင်ပြခဲ့ပြီးပါပြီ။ သို့သော် အားလုံးတော့ မဟုတ်သေးပါ။ Computer ကို Decisions ပြုလုပ်ရန်အတွက်လည်း သုံးနိုင်ပါသည်။ Java သည် 'if' နှင့် 'switch' statements များ တည်ဆောက်မှုနှစ်ခုကို ပြုလုပ်နိုင်သည်။

The If Statement

၎င်း၏အဓိပ္ပာယ်မှာ ဤဟာဖြစ်လျှင် ဤဟာပြုလုပ်မည်၊ မဖြစ်လျှင် ထိုဟာကို ပြုလုပ်မည်ဟူ၍ ဖြစ်သည်။ ဥပမာ- Ram သည် 60% marks ရလျှင် သူ၏အဖေက သူ့ကို နာရီတစ်လုံးပေးမည်၊ မရလျှင် ဖောင်တိန်တစ်ချောင်းပေးမည်။ Java မှ 'if' statement သည် English Language မှ 'if' စကားလုံးနှင့် တော်တော်များများတူသည်။

၎င်းသည် အခြေအနေပေါ်တွင် မှီနေသော true or false ဖြစ်စေသော condition တစ်ခု သို့မဟုတ် logical statement တစ်ခုဖြစ်သည်။ Current value ပေါ်မှာ အခြေခံ၍ actions အမျိုးမျိုး ပြုလုပ်သည်။ ရိုးရိုးပုံစံဖြင့် ရှင်းပြရလျှင်

```
if (condition)
   action;
```

Condition သည် True ဖြစ်လျှင် action တစ်ခုတည်းကိုသာ ပြုလုပ်မည်ဟု ဆိုလိုသည်။ ဥပမာ-

```
if  (marks <50))
system.out. println ("3rd  division");
```

int အမျိုးအစား variable marks ကို ကျောင်းသားတစ်ယောက်၏ ရမှတ်ကို သိမ်းရန် သုံးထားသည်။ ဤ statement သည် result ရရှိရန် တွက်ချက်သည့် အပိုင်းတစ်ခုဖြစ်သည့် Condition ၏ ရှေ့နောက်တွင် brackets များ လိုအပ်သည်။ Condition သည် အလွန်ရှည်သော statement ဖြစ်နိုင်သည်။

```
if    (    (sub1marks >= 50)  &&
       (sub2marks >= 50)  &&
       (sub3marks >= 50)  &&
       (sub4marks >= 50)  &&
       (sub5marks >= 50)  )
   System.out.println("Passed");
```

&& သည် and ဟု ဆိုလိုပြီး > = သည် greater than or equal to ဟု ဆိုလိုသည်။ brackets များကို opening- closing အစုံလိုက် ဂရုတစိုက် arrange သတ်မှတ်ပေးရပါမည်။ ထို့အတူ action part တွင်လည်း နှစ်ခု သို့မဟုတ် နှစ်ခုထက်ပိုတဲ့ statements များ ပါဝင်နိုင်သည်။

```
if (marks < 50)
   {
   System.out.println("Failed");
   TotalFailures = TotalFailures + I;
   }
```

Statements အားလုံးသည် curly brackets { and } အတွင်းမှာ တစ်ခုပြီးတစ်ခု 'if' statement အတွက် အကျုံးဝင် ပြုလုပ်သည်။ Statements များထဲတွင်ပင် အခြား 'if' statement ပါဝင်နိုင်သေးတယ်။

```
if  (marks >=50)
   {
   System.out.printIn("Passed");
   if (marks >= 60)
      System.out.println("in First Class");
   }
```

တစ်ကြိမ်ထက်မကလည်း ပါနိုင်ပါသည်။

```
if (marks >=50)
   {
   System.out.printIn("Passed");
   if (marks >=60)
      {
      System.out.printIn(" in First Class");
      if (marks >=70)
            System.out.printIn
                  ("With Distinction");
      }
   }
```

The Else Part

Java မှ 'if' construct ကို styles နှစ်မျိုးဖြင့် သုံးနိုင်သည်။

```
if (condition)
      action1; // condition evaluates to true.
or
   if (condition)
      action1; // condition evaluates to true.
else
      action2; // condition evaluates to false.
```

သင်ရေးလျှင်

```
   if (marks < 50)
   System.out.printIn ("Failed");
else
      System.out.printIn ("Passed");
```

၎င်း statement အား အခြားနည်း တစ်နည်းဖြင့် ရေးလျှင်လည်း အတူတူပင်ဖြစ်သည်။

```
if (marks >= 50)
   System.out.printIn ("Passed");
else
   System.out.printIn("Failed");
```

ယခုဆက်လက်ပြီး

```
   if (x > y)
```

```
    x = y;
y = 0;
```

y = 0 case မှာ အပေါ်မှ 'if' တွင် ဘာမှ အလုပ်မလုပ်ပါ။ (x> y) condition ကိုတော့ တွက်ချက်ပါလိမ့်မည်။ x > y ဖြစ်လျှင် y = 0 ဖြစ်စေချင်လျှင် နောက်ဆုံးလိုင်း ၂ လိုင်းအား curly brackets အတွင်း ထားသင့်ပါသည်။

```
if (x > y)
   {
   x = y;
   y = 0;
   }
```

အခြား example ပြပါဦးမည်။

```
if (condition1)
if (condition2)
action3;
else
action4;
```

ဥပဒေသ သတ်မှတ်ချက်အရ 'else' သည် အနီးဆုံး 'if' နှင့် သက်ဆိုင်သည်။ ဥပဒေသအရဆိုလျှင် condition 1 သည် မှားလျှင် ဘာမှ ဖြစ်မလာတော့ပါ။ မှန်လျှင် action 3 ဒါမှဟုတ် action 4 ကို condition 2 ပေါ်မူ၍ ပြုလုပ်မည်။ action 4 သည် action 3 နှင့် တခြားစီ ဖြစ်သည်။ ၎င်းတို့အနက် တစ်ခုကိုတော့ ပြုလုပ်ရမည်။

```
if (condition1)
{
if (condition2)
action3;
}
else
action4;
```

Condition 1 မှားလျှင် action 4 ကို ပြုလုပ်မည်။ Condition 1 မှန်လျှင် action 3 ကို condition 2 မှန်မှသာ ပြုလုပ်မည်။ ထိုကဲ့သို့ codes များကို ရေးသည့်အခါ အစအဆုံး အစွန်းများကို သတိထားရပါမည်။

```
if (condition1)
   if (condition2)
   action3;
   else
   action4;
```

အခြားတစ်နည်းမှာ

```
if (condition1)
   {
   if (condition2)
   action3;
   }
else
   action4;
```

The ?: Operator

Java သည် 'if ... else' statement အတွက် အတိုကောက် သတ်မှတ်ချက် ?: ကို သုံးသည်။ Statement အဖြစ်

```
condition ? action 1 : action 2;
```

၎င်းသည်

```
if (condition)
   action1;
else
   action2;
```

နှင့် အတူတူပင် ဖြစ်သည်။

?: operator သည် {နှင့် } brackets အတွင်းမှ ကြီးမားသော statements အစုထက် action 1 နှင့် action 2 တို့သည် single action ဖြစ်သည့်အခါ သုံးသည်။ ဤ operator သည် 'if' statement ဖြင့် single-line code တစ်ခုရေးရန် လိုသည့်အခါ ယေဘုယျ သုံးသည်။ single expression တစ်ခုအား ဖော်ပြထားသည့်အတိုင်း သုံးပါတယ်။ ဥပမာ အားဖြင့် x သည် a နှင့် b ၏ maximum ကို သိမ်းမယ်ဆိုလျှင်

```
if (a > b)
   x = a;
else
   x = b;
```

?: ကို သုံး၍ ရေးလျှင်

```
x = a>b ? a : b;
```

အဲဒီမှာ x သည် ?:operator အတွင်းသို့ မရောက်ပါ။ သို့ သော် ?: နှင့် တန်ဖိုး ကွက်ချက်၍ x ကို assign ပြုလုပ်သည်။ သင်သည် if...else ကို သုံး၍ အမြဲတမ်း ရေးသော်လည်း တစ်ခါတရံမှာ ?: operator သည် အလွန်အဆင်ပြေပါသည်။

The Switch Statement

Action တစ်ခု သို့မဟုတ် နှစ်ခုတွင်သာ 'if' statement သည် အသုံးဝင်သည်။ သို့သော် actions များလာတဲ့အခါမှာ အဆင်မပြေတော့ပါ။ သင်သည် တစ်လအတွင်း

မတူညီသော နေ့များအတွင်း မတူညီသော actions များကို ပြုလုပ်ရန် လိုအပ်လျှင် == (equal to) ကဲ့သို့ တစ်စုံတစ်ခုရေးခြင်းဖြင့် စစ်ရပါမည်။ (=) sign အား လက်မခံပါ။ Assignment တွင်

```
if (date ==1)
actions ;
else
if (date == 2)
action 2;
else
if  (date  = = 3)
action 3;
else
if (date == 4)
action 4 ;
else
..... •
```

ဤအခြေအနေအား switch statement ကို အောက်ပါအတိုင်း အသုံးပြု၍ အဆင်ပြေစေသည်။

```
switch (date)
{ // _____ switch begins here.
   case 1:  { action1; break; }
   case 2:  { action2; break; }
   case 3:  { action3; break; }
   case 4:  { action4; break; }
   ........ .......... ..
   case 31: { action3; break; }
   default:  {default-action; break; }
} // ................. switch ends here.
```

Computer တွင် .... case နေရာ၌ ဆက်ထည့်သွားပါ။ Date ၏ value 3 ဖြစ်မည်ဆိုလျှင် ဥပမာ computer သည် case 1 ကို သွားမည်။ ထို့နောက် {နှင့်} ကြားမှာရှိသော အလုပ်များကို တစ်ခုပြီးတစ်ခု ခုန်သွားပါမည်။ ပြီးနောက် case 2 ကို သွားပြီး ပထမပုံစံအတိုင်း ပြုလုပ်သည်။ ပြီးလျှင် case 3 ကိုသွာသည်။ အဲဒီမှာ date ၏ value ကို တွေ့ရှိမှန်သွားပြီ။ ဤတွင် { နှင့် } blackets အတွင်းမှ statement အားလုံးကို ပြုလုပ်မည်။ } မတိုင်မီ 'break;' statement ပါဝင်နေလျှင် ကွန်ပျူတာသည် switch statement အဆုံးထိသွားရင်းနှင့် case စစ်ပြီးမှန်သည်နှင့် ရုတ်တရက်ရပ်ပြီး switch statement မှ ထွက်သွားပါမည်။ } မတိုင်မီ နောက်ဆုံး state ment အဖြစ် break; ကို မဖော်ပြလျှင် ကွန်ပျူတာသည် ဆက်လက် checking ပြုလုပ်နေပြီး case 4 အစ ရှိသဖြင့် ဆက်သွားနေပါမည်။

Computer သည် default ဟုခေါ်သော last case ပြုလုပ်မယ့်နေ့တွေမှာ actions မရှိပါ။ Default case သည် မိမိစိတ်ဆန္ဒအရ သုံးခွင့်ရှိသော်လည်း အသုံးပြုလျှင် ပို၍ကောင်းပါတယ်။ အမှားနည်းပါတယ်။ အမှားဖြစ်စေနိုင်တဲ့ value တွေကိုလည်း ရှာဖွေပေးနိုင်တယ်။ ဥပမာ ရှေ့မှ case တွင် value အား 32(သို့) 33 သို့မရောက်ရှိနိုင်စေရန် default ကို ထည့်ပေးရတယ်။ Curly brackets {နှင့်} အတွင်းမှ case အားလုံးနှင့် သူတို့၏ actions များကို မမေ့ပါနှင့်။

REPETITIONS

ပေးထားသော အလုပ်တစ်ခုအား ထပ်ခါထပ်ခါပြုလုပ်ခြင်းသည် ပြုလုပ်ရသော လူပုဂ္ဂိုလ်အတွက် ငြီးငွေ့စရာ ကောင်းပါတယ်။ ဒါပေမဲ့ ကွန်ပျူတာများတွင်တော့ အသုံးဝင်တဲ့ ကြိုးမားသော စွမ်းရည်တစ်ခုပါပဲ။ Java မှ repetitions ဆောင်ရွက်ရန် constructs အမျိုးမျိုးရှိပါတယ်။

တစ်ခုစီသည် ကွန်ပျူတာအား မည်မျှ အချိန်ကြာကြာ ဆောင်ရွက်ရမည် ဘယ်အချိန်ရပ်မည်ကို ပုံစံအမျိုးမျိုးနှင့် ပြောတယ်။ Repetition ကို Computer science မှာ loop (သို့) Looping ဟုလည်း ခေါ်ပါတယ်။

The While Statement

'while' statement ၏ ပုံစံမှာ

```
while  (condition)
   action  1;
```

(သို့) actions များလျှင်

```
while  (condition)
   {
   action 1;
   action 2;
   action 3 ;
   ..........
   }
```

'if' case မှာကဲ့သို့ condition သည် variable ၏ current value ပေါ်တွင် မူ၍ TRUE/FALSE ကို တွက်ချက်သော statement တစ်ခုဖြစ်ပါတယ်။ action part သည် condition TRUE ဖြစ်နေသမျှ ထပ်ခါထပ်ခါ လည်ပတ်နေပါမယ်။ Variable အချို့သည် ဆောင်ရွက်နေရင်း value ပြောင်းသွားမယ်။ တစ်ချိန်ချိန်မှာ condition-FALSE ဖြစ်သွားတဲ့အခါ statement သည် ရပ်သွားမယ်။ 'while' loop အား ပိုပြီးနားလည်စေရန် အောက်ပါ ဥပမာနှင့် ရှင်းပြပါမယ်။

```
class  while  Test 1
```

```
{ Public static void main (string args[ ])
   {
   init i = 2;
   while (i < 15)
         {
         system.out.println(i);
         i = i * 2;
         )
   )
}
```

၎င်း၏ Output မှာ-

```
2
4
8
```

int i ကြေညာခြင်းနှင့် initialization statement i = 2 အား single statement int i = 2 တစ်ခုတည်းဖြင့် ပေါင်းထားပါတယ်။ Java မှာ ၎င်းကို ခွင့်ပြုတယ်။ ပိုပြီးလည်း အဆင်ပြေတယ်။ while ကို စတင်ဝင်ရောက်တာနဲ့ i တွင် 2 ရှိတယ်။ i < 15 condition တွင် 2 < 15 သည် မှန်တယ်။

ထို့ကြောင့် body statement ကို တွက်ချက်တယ်။ 2 . ကို print ပြလုပ်ပြီး i = i * 2 ကို ဆောင်ရွက်တယ်။ rhs မှ i သည် i ၏ current value ဖြစ်၍ 2 ဖြစ်တယ်။ * sign သည် မြှောက်ခြင်း (Java တွင်) ဖြစ်၍ rhs တွင် 2 * 2 =4 ရရှိပြီး ယခု i ၏ value မှာ 4 ဖြစ်သွားတယ်။

ပြီးနောက် Computer သည် while ကို ပြန်သွားတယ်။ procedure တစ်ခုလုံးကို ပြန်ဆောင်ရွက်တယ်။ i = 4 ဖြစ်လျှင် 4 < 15 သည် မှန်တယ်။ 4 ကို print ပြလုပ်ပြီး i = i *2 ကို တွက်ချက်၍ i သည် 8 ဖြစ်သွားတယ်။ while ကို ပြန်သွားပြီး 8 < 15 သည် မှန်တယ်။ 8 ကို print ပြလုပ်ပြီး i = i *2 ကို တွက်ရာ 16 ဖြစ်သွားတယ်။ while ကို ပြန်သွားရာ အဲဒီအခါ condition 16 < 15 သည် မှားသွားပြီ။ 'while' statement သည် ရပ်သွားပြီး အခြား procedure များ မပြုလုပ်တော့ပါ။ တန်ဖိုး ၃ ခု 2, 4, 8 အား print ထုတ်မယ်။ while state ment ၏ လိုအပ်ချက်တစ်ခုကတော့ condition မှာ ပါဝင်သော variable များသည် statement အတွင်းမှာ တန်ဖိုးပြောင်းလဲမယ်။

ပိုမိုအရေးကြီးတဲ့ လိုအပ်ချက်ကတော့ variables များသည် တစ်ချိန်ချိန်မှာ condition - FALSE ဖြစ်စေသည်အထိ တွက်ချက်မှုများနှင့် value ကို ပြောင်းသင့်တယ်။ Loop သည် မရပ်ဘဲ ဆက်သွားနေမည့် နမူနာ ပရိုဂရမ်မှာ-

```
class  whileTest 2
{ public static void main (String args [ ])
   {
   int i = 1;
   while (i ! = 6)
      {
```



```
      System.out.println(i);
      I = I + 2;
      )
   )
}
```

အဲဒီနေရာမှာ i! = သည် Java တွင် not equal to ဟု အဓိပ္ပာယ်ရပါတယ်။ out put မှာ-

```
1
3
5
7
9
11
13
. . . .
```

၎င်းသည် မရပ်ဘဲ သွားနေမယ်၊ ဘာကြောင့် ဖြစ်ရသလဲဆိုလျှင် i ပြောင်းလဲတိုင်းမှာ 6 နှင့်မတူသည့် "မ" တန်ဖိုးများ ဖြစ်နေတယ်။ အမှန် condition မှ i< 6 ဖြစ်ရမယ်။ i သည် 7 ဖြစ်လျှင် FALSE ဖြစ်သွားမယ်။

အရေးကြီးသော အချက်တစ်ခုကတော့ while statement တွင် condition ပြီးလျှင် semicolon (;) မပါရှိရပါ။

ဥပမာ

```
while  (condition);
   action 1;
```

or

```
while  (condition);
   {
   action 1;
   action 2;
   .............
   }
```

၎င်းသည် မှားတယ်။ semicolon သည် two statement မှာပဲ ပြုလုပ်ရတယ်။ ပထမဖြစ်စေတဲ့ ပြဿနာကတော့ action part ကို မဆောင်ရွက်ဘဲ condition ထဲမှာ variable သည် ပြောင်းလဲခြင်း မရှိတော့ပါ။

The DO Statement

'do' statement ပုံစံမှာ

```
do
   action 1;
while  (condition)
```

သို့မဟုတ် action များပါက

```
do
   {
   action 1;
   action 2;
   action 3;
   ..............
   }
while (condition)
```

Condition သည် variable ၏ current value အပေါ်တွင် TRUE(သို့မဟုတ်) FALSE ကိုတွက်ချက်တဲ့ statement တစ်ခုဖြစ်တယ်။ Action part သည် condition - TRUE ဖြစ်နေသမျှ ထပ်ခါထပ်ခါ ဆောင်ရွက်နေမှာ ဖြစ်တယ်။ ဆိုလိုသည်မှာ variables များသည် execution အတွင်းမှာ တန်ဖိုးပြောင်းလဲနေပြီး တစ်ချိန်ချိန်မှာ condition -FALSE ဖြစ်သွားပြီး statement ရပ်သွားမယ်။

' do ' Loop မှာ action part ကို အရင်ဆောင်ရွက်တယ်။ ပြီးမှ condition ကို check လုပ်တယ်။ ဆိုလိုသည်မှာ actions ကို အနည်းဆုံးတစ်ကြိမ် ပြုလုပ်တယ်။ ၎င်းသည် ' while ' statement နှင့် ' do ' statement ၏ အဓိက ခြားနားချက်ပဲ ဖြစ်တယ်။ အခြားတော့ မရှိပါဘူး။

The FOR Statement

'For' တည်ဆောက်မှုသည် တင်ပြချက်များတွင် အလွန်အဆင် ပြေတယ်။ အကြောင်းမှာ body မှာ action part သာထားရှိပြီး မတူညီသော အပိုင်း (၃) ပိုင်း (initializing variables, testing the condition, changing cariable values) ကို header part တွင် ထားနိုင်တယ်။ Java မှ ' for ' statement တစ်ခုကို ကြည့်ကြစို့။

```
for  (initialize ; condition; change)
```

(သို့မဟုတ်)

```
for  (initialize ; condition; change)
   {
   action 1;
   action 2;
   ..... .
   }
```

ရှေ့မှ 2 ၏ power ကို print ပြုလုပ် ပရိုဂရမ်ကို ရေးကြည့်လျှင်

```
class forTest1
{
   public static void main(string args[ ])
   {
   for (int i = 2; i < 15; i = i * 2) system.out.println(i);
```



```
         i = i - 1;
   }
}
```

မှတ်ရန်မှာ ' for' statement အတွင်းမှာ variable i အား ကြေညာသည့်အခါ initialization ပါ တစ်ခါတည်း ပြုလုပ်နိုင်ခြင်းဖြစ်သည်။ လူအများက ' for' statement သည် ' while ' သို့ ' do ' statement သုံးခြင်းထက် ပို၍အဆင်ပြေသည်ဟု ဆိုကြပါသည်။ ၎င်းသည် debugging အတွက်လည်း ကူညီသည်။ အကြောင်းမှာ ပြောင်းလဲမှုနှင့် ထိန်းချုပ်မှု အချက်အလက် အားလုံးဟာ ထိပ်မှ header line သောကြောင့်ဖြစ်သည်။ Condition မှ variables ပြောင်းလဲမှုများကို ထိပ်မှာသာ ဖော်ပြရသည်မဟုတ်ပါ။ Loop အတွင်းမှာလည်း ပြောင်းနိုင်ပါသေးသည်။

```
class forTest2
{
   public static void main(string args[ ])
   {
   for (int i = 2; i < 15; i = i * 2)
         {
         system.out.println(i);
         i = i - 1;
         }
   }
}
```

၎င်းမှ –

```
2
3
5
9
```

ရရှိသည်။ နောက်တစ်မျိုး ကြည့်နိုင်ပါသေးသည်။

```
class forTest3
{
   public static void main(string args[ ])
   {
   for (int i = 30; i > 5) // Last part empty.
         {
         system.out.println(i);
         i = i - 5;
         }
   }
}
```

Output မှာ

```
30
25
20
```

15

10 ဖြစ်သည်။ သို့မဟုတ်

```
class forTest4
{
   public static void main(string args[ ])
   {
   int i = 2;
   for ( ; i < 30; )
         // First and last part empty.
   {
   system.out.println(i);
   i = i * 2;
   }
}
```

Program ရေးခြင်းအပိုင်းတော့ ပြီးသွားပါပြီ။ နောက်လာမည့် အပိုင်းများတွင် Java ၏ advanced programming သည်ဘာလဲ၊ Java ၏ နောက်ထပ် commands များအကြောင်းကို ဘယ်နေရာမှာ လေ့လာမလဲ စသည်များကို ဖော်ပြသွားပါမည်။

အခန်း (၄)

APPLETS PROGRAMMING

ပြီးခဲ့သည့် သင်ခန်းစာများတွင် programming တွေနဲ့အတူ သင် ငြီးငွေ့နေပါလိမ့်မည်။ Java မှ ready made programs အချို့ကို သင်လိုချင်ပါလိမ့်မည်။ Java မှာ ready made programs တွေ ရှိပါသည်။ အဆိုပါ programs တွေကို Applets ဟု ခေါ်ပါသည်။ အများစုကို net ပေါ်မှာ အသုံးချလို့ရပြီး download, run လုပ်နိုင်ပါသည်။ နောက်ပြီး အဲဒီ applets တွေကို အသုံးပြုပြီး animation ပြုလုပ်နိုင်ပါသေးသည်။ သူတို့ကို သင် လိုအပ်သလို ပိုကောင်းအောင် ပြောင်းနိုင်ပါသည်။

ရှေ့မှာ သင်လေ့လာခဲ့ရသည့် Java programming က သာမန်ရိုးရိုး programming များသာ ဖြစ်သည်။ အဲဒီ program များသည် stand-alone programs များ ဖြစ်ပါသည်။ ၎င်းတို့သည် screen နောက်မှာသာ run ပြီး file တစ်ခုမှာ output ရေးရန် စီစဉ်ထားလျှင် ၎င်းတို့ run သော အကြောင်းအရာများကို screen ပေါ်တွင် မည်သို့မျှ မပြပါ။ အဆိုပါ ပုံစံမျိုး သင်ခန်းစာတွေက လွန်ခဲ့တဲ့ နှစ်ပေါင်း ၇၀၊ ၈၀ နှစ်များက ရှိခဲ့သော အဓိက ဖိုင်တစ်ခုအဖြစ် တည်ဆောက်ရဲ့ သက်သက်ပါပဲ။ နောက်ပိုင်းမှာတော့ အများအားဖြင့် program တွေဟာ Windows ကို အခြေခံလာသည်။ Visual လို programming language မျိုးဖြစ်သည်။ ယေဘုယျအားဖြင့်တော့ input/out put များသည် screen ပေါ်မှာ window ဖွင့်ပြီးမှ ပြုလုပ်နိုင်ပါသည်။ အဆိုပါ လိုအပ်ချက်များအားလုံး ပြေလည်စွာ ဆောင်ရွက်နိုင်ရန် Java သည် applets facility နှင့်အတူ အရေးပေါ် ပေါ်ပေါက်ခဲ့ပါသည်။

အဓိက,ကတော့ applet သည် problem ၂ ခုကို ဖြေရှင်းပေးသည်။ Program ကို run ရင်းနှင့် တစ်ခါတည်း show ပြသွားသည်။ ရှေ့မှ program များတွင် ပိုပြီး complicated ပြုလုပ်ပေးသည်။ Applet တစ်ခုသည် fonts, colours များနှင့် shape များ အသုံးပြု၍ မည်ကဲ့သို့ရေးနိုင်သည်ဆိုခြင်းကို လေ့လာကြည့်ကြစို့။

APPLET CREATION

test 1.Java အမည်နှင့် file တစ်ခု တည်ဆောက်လျှင်-

```
import java.applet.Applet;
import java.awt.Graphics;
public class test1 extends java.applet.Applet
   {
   public void paint (Graphics gg)
```

```
    {
    gg.drawString ("Sachin Tendulkar!", 5, 25);
    }
}
```

ပြီးခဲ့သည့် chapterမှ programတွင် ပြောင်းလဲသွားသောအချက်ကို သတိထားမိမှာပါ။ Program ၏ ထိပ်တွင်

```
import java.applet.Applet;

import java.awt.Graphics;
```

လိုင်း ၂ လိုင်း တိုးလာပါသည်။

Java သည် store ပြုလုပ်ထားသော import statement များကို ယူသုံးနိုင်တယ်။ Package တစ်ခုသည် related classes ၏ collection တစ်ခုမျှသာ ဖြစ်သည်။ ဤ caseတွင်မူ public classesဖြစ်သော java.applet. Appletနှင့် java.awt.Graphic များကို Java Kit ၏ part တစ်ခုအဖြစ် ယူသုံးထားသည်။ awt သည် abstract windowing မှာ toolkit ကို ဆိုလိုသည်။

နောက် ပြောင်းလဲချက်ကတော့ class definition ဖြစ်သည်။

```
public class test 1 extends Applet
```

extends keyworkသည် Applet class၏ sub classတစ်ခုအဖြစ် ညွှန်ပြတယ်။ Applet classသည် inputပြု လုပ်ခဲ့သော java.applet. Applet packageတွင် ဖွင့်ဆို ရှင်းပြထားပါသည်။ test 1 သည် Applet class ၏ subclass တစ်ခုဖြစ်ပြီး Applet class ၏ functionအားလုံးကို အလိုလို ရရှိသည်။ Appletပြုလုပ်နိုင်သော အရာတိုင်း ကို testကလည်း ပြုလုပ်နိုင်သည်။ Appletနှင့် applicationကြားမှာ နောက် ခြားနား ချက် တစ်ခုကတော့ သီသာထင်ရှားမှု နည်းစေခြင်းပင် ဖြစ်သည်။ Appletsတွင် Main method မရှိပါ၊ Main method က အမှန်တကယ် browser သို့မဟုတ် applet-viewerမှာပဲ ရှိသည်။ Appletsကိုယ်တိုင်မှာတော့ မရှိပါ။ Appletsများ သည် extra functionality များကို အံ့ထောက်အကူ ဖြစ်စေသော code modules များ ဖြစ်သည်။ သို့သော် သူတို့ကို Main program မှာ ဖော်ပြမထားပါ။ Applets ဖော်ပြရမယ့် နေရာမှသာ code စတင်ပါသည်။

Applet တစ်ခုသည် key press, applets visible area မှ ရွေ့လျားနေသော mouse pointerသို့မဟုတ် mouse clickတို့ကဲ့သို့ event seriesတစ်ခု ရရှိပြီးနောက် သတ်မှတ်ထားသော eventhandlerကို ဆောင်ရွက်သည်။ ရှေ့မှ programတွင် event handler တစ်ခုသာရှိပြီး နောက်ပိုင်းမှာ events အကြောင်း ပိုပြီးလေ့လာရပါမည်။ Applets အများစုသည် paint eventကို handleပြုလုပ်ရန် လိုအပ်သည်။ ဤ event သည် applet ၏ visible area အစိတ်အပိုင်းတစ်ခုအား uncover ပြုလုပ်ပြီး

ပြန်၍ရေးဆွဲရပါမည်။ Paint method သည် ggအား Graphics objectတစ်ခုအဖြစ် ရရှိတယ်။ Graphic class သည် importပြုလုပ်သော java.awt.Graphic packageမှာ အဓိပ္ပာယ်ဖွင့်ဆို ပါဝင်ပါသည်။ Paint method မှာတော့ gg's drawstring methodကို "Sachin Tendulkar!" ဟုခေါ်သော stringအား coordinates (5,25) တွင် ရေးဆွဲရန် သုံးပါသည်။ Applet ၏ upper left hand Cornes မှ 5 pixels across နှင့် 25 pixels down နေရာဖြစ်သည်။ ဤ drawing ဟာ applet screen ရဲ့ portion တစ်ခုအား covered ပြုလုပ်တိုင်း နေရာရွှေ့ပြီးမှ uncovered ပြုလုပ်၍ refreshed ပြုလုပ်ပါသည်။

ရှေ့မှ appletအား အောက်ပါ command ဖြင့် compile ပြုလုပ်ကြစို့။

```
Javac  test 1.  java
```

သင့် directory မှ test.class ဟုခေါ်သော fileတစ်ခု ရရှိသွားပြီ။ ဤ applet ကို ကြည့်ရန် Netscape 2.0 ကဲ့သို့ Java capable browser တစ်ခု သို့မဟုတ် HotJava တို့ လိုအပ်သည်။ သို့မဟုတ်ပါက Java compiler နှင့် အတူပါလာသော applet viewer ကို သုံးနိုင်သည်။ applet viewerသည် web browserတစ်ခု မဟုတ် ပါ။ Web pageတစ်ခုလုံးကို ကြည့်ရန် မဖြစ်နိုင်ပါ။ သို့သော် applet ကို မြင်ချင်သည့် ပုံစံမျိုးနှင့် testing လုပ်လုပ်၍ ရပါသည်။

ယခုအခါ test 1.class ၏ same directory မှ test 1.html file ကို တည်ဆောက်ပါမည်။ (HTML အကြောင်း နောက်ပိုင်းတွင် ဖတ်ရပါမည်)။ test 1. html ရဲ့ contents များမှာ အောက်ပါအတိုင်း ဖြစ်သည်။

```
< HTML>
   <HEAD>
      <TITLE>
           Draw String Using test1
      </TITLE>
   </HEAD>
   <BODY> <P>My Java applet say:
      <APPLET CODE="test1.class" WIDTH=400 HEIGHT=200>
      </APPLET>
   </BODY>
</HTML>
```

၎င်းမှာ <နှင့်>ကြားမှ charactersများသည် upper caseဖြစ်ရပါမည်။ အချို့ကို lower caseခွင့်ပြုသော်လည်း upper case onlyကတော့ အဆင်မပြေမှု အနည်းဆုံးပါပဲ။

command ပေးကြည့်ပါ။

```
appletviewer  test 1.html
```

သင်၏ terminal မှ applet run ထားခြင်းကို ကြည့်ပါ။ သင်သည် applet viewer ကို သုံးလျှင် (My Java applet says...) appletကို မမြင်ရပါ။ Sachin Tendulkar! ကို screen မှာ မြင်ရပါမည်။

FONTS

Word processing knowledge မှ font style ကို သင်သိခဲ့ပြီးပြီ။ Word processing မှာ fonts အများစုသည် အသုံးပြုနိုင်သော်လည်း Java တွင် အချို့ fonts များကို ကန့်သတ် တားမြစ်ထားပါသည်။ Printed text တစ်ခုအဖြစ် output ၏ font size သတ်မှတ်ရာတွင် previous program မှာတော့ extension တစ်ခုပဲရှိပါသည်။ ထိုအချိန်တွင် applet နှင့် out packages မှာ class တစ်ခုထက်ပိုပြီး သုံးရန်လိုအပ်လာပါသည်။ import ပြုလုပ်ရန် ၎င်းတို့အားလုံးကို wild-card character * ကို သုံးပြီး ညွှန်းရပါမည်။ Compiler သည် program မှ ဖော်ပြချက်များအား လိုအပ်သလို link ပြုလုပ်ပေးထားသည်။

```
import  java.awt. *;
import java.applet.*;
public class test1 extends Applet
   {
      public void paint (Graphics 99)
            {
            Fonts testfont = new Font
                  ("TimesRoman", Font.BOLD, 24) ;
            gg.setFont(testfont);
            gg.drawString("Sachin Tendulkar!", 5,25);
            }
   }
```

output မှာ

Sachin Tendulkar

FONT NAME

Java တွင် font တစ်ခုသည် object တစ်ခုသာဖြစ်သည်။ Class of fonts ကို Java တွင် Font ဟု ခေါ်ပါသည်။ ၎င်းသည် Java.awt package တွင်ပါဝင်သည်။ အပေါ်က example မှာ testfont ကို class font ရဲ့ object တစ်ခုအဖြစ် ကြေညာထားပါသည်။ ထိုလိုင်းမှာပင် new ကို သုံးပြီး object တည်ဆောက်ထားပါသည်။ New ပြီးနောက် constructor ကို signature တစ်ခုနှင့်အတူ font class မှ ရှင်းပြထားတယ်။

```
public  Font (String  name, int  style, int  size)
```

Java မှာ အသုံးပြုသော font names အစုကတော့

TimesRoman	Sample Text
Helvetica	Sample Text
Courier	Sample Text

သင့်ရဲ့ system ပေါ်မှာ မူတည်ပြီး fonts ပိုပြီး ထင်ရှားလာစေပါသည်။ သို့သော် အဆိုပါ font သုံးမျိုးကတော့ system တိုင်းမှာ သုံးကြလေ့ရှိသည်။

FONT STYLE

Second parameter မှာ font style ပဲဖြစ်သည်။ ယေဘုယျအားဖြင့် text များတွင် style ၄ မျိုးသုံးပါသည်။

PLAIN	Sample Text
BOLD	**Sample Text**
ITALIC	*Sample Text*
BOLD and ITALIC	***Sample Text***

Class Font တွင် ဤ words များကို အောက်ပါအတိုင်း ဖော်ပြပါသည်။

```
public  static  final  int  PLAIN = 0;
public  static  final  int  BOLD = 1;
public  static  final  int  ITALIC = 2;
```

Font တစ်ခု၏ net style ကတော့ အားလုံး၏ ပေါင်းလဒ်ပဲ ဖြစ်ပါသည်။ ထို့ကြောင့် text သည် BOLD နှင့် ITALIC နှစ်ခုစလုံးဖြစ်လျှင် style သည် 3 ဖြစ်မည်။ Second parameter နေရာမှာ Font.BOLD ဟုရေးရန်လိုသည်။ (BOLD တစ်ခုသာရေးလျှင်) text သည် BOLD နှင့် ITALIC နှစ်ခု ဖြစ်လျှင် Font.BOLD + Font.ITALIC ကို သုံးပါ။

FONT SIZE

Third parameter မှာ font ၏ size ဖြစ်ပါသည်။ ၎င်းကို point ဟု ခေါ်သော unit ဖြင့် ဖော်ပြသည်။

1 inch = 72 points

မည်သည့် positive integer ကိုမဆို ၎င်း၏ value အဖြစ်သုံးနိုင်တယ်။ New ကိုသုံးပြီး သင်၏ own font testfont ကို define ပြုလုပ်ပြီးနောက် text ကိုရေးဆွဲနေစဉ်မှာ Graphic ၅၅ ဖြင့် အသုံးပြုရန် font ကို နေရာချထားပေးဖို့ လိုသည်။ Graphic class ရဲ့ setFont method ကို အသုံးပြုမှာ ဖြစ်သည်။ အောက်ပါ ဥပမာနှင့် အဓိပ္ပာယ်ကို ရှင်းပြထားသည်။

```
import  java.awt.*;
import java.applet.*;
public class test1 extends Applet
   {public void paint(Graphics gg)
      {Font testlfont = new Font
```

```
        ("Helvetica", Font. PLAIN, 14),
    gg.setFont(test1font);
    gg.drawString("Sachin Tendulkar!",5,35);
    Font test2font = new Font
        ("TimesRoman", Font.Bold, 24);
    gg.setFont(test2font);
    gg.drawString("Sachin Tendulkar!",5,75);
    Font test3font = new Font
        ("Courier", Font.ITALIC, 36);
    gg.setFont(test3font);
    gg.drawString("Sachin Tendulkar!",5,155);
    Font test4font = new Font
        ("Helvetica", Font.Bold + Font. ITALIC, 20);
    gg.setColor(Color.orange);
    gg.drawString("Sachin Tendulkar!",5,255);
    }
}
```

COLOURS

အပေါ်က example မှာ colours အသုံးပြုပြီးပြီး ၎င်းကို ကြေညာရပါမည်။ Java မှာ Colours ကို မည်ကဲ့သို့ သုံးသလဲ ကြည့်ကြစို့။ Output window မှာ ရေးနေစဉ် text colour ကို Change နိုင်သည်။ Java တွင် colour သည် object တစ်ခု ဖြစ်ပြီး colours အားလုံး၏ class ကို color (not colour) ဟု သတ်မှတ်သည်။ Colour ကို RGB model ဖြင့် Red, Green, Blue components များနှင့် သတ်မှတ်ဖော်ပြပြီး range [0, 255] တွင် integers အဖြစ် colour တစ်ခုစီကို သိမ်းသည်။ သင် bits အကြောင်းကို သိလျှင် colour တစ်ခုစီသည် 8 bits အသုံးပြု၍ သိမ်း၍ totol colour value မှာ 24 bits ဖြစ်ပြီး int variable တစ်ခုဖြစ်သည်။

Red component in bits 16-23

Green component in bits 8-15

Blue component in bits 0-7 တို့သည် convention ဖြစ်တယ်။

Actual colour သည် best match given the colour space ပေါ်တွင် မူတည်သည်။ Colour ရွေးချယ်ရန် suitable constructors များသည် Java Package documentation မှ colour word spelling များကို အသုံးပြု ဖော်ပြသည်။

Public Color (Int r, Int g, Int b)

Range (0-255) တွင် ဖော်ပြထားသော red, green, blue values တို့နှင့် colour တစ်ခု တည်ဆောက်ပါ။

Public Color (float r, float g, float b)

Range (0.0–1.0) တွင် ဖော်ပြထားသော red, green, blue values တို့နှင့် color တစ်ခု တည်ဆောက်ပါ။

ပေးထားသော colour ၏ individual components ကို ထုတ်နုတ်၍လည်း သုံးသည်။ Component တစ်ခုစီကို control ပြုလုပ်နိုင်သည်။

Public Int getRed ( )

Red component ကို ရသည်။

Public Int getGreen ( )

Green component ကို ရသည်။

Public Int getBlue( )

Blue component ကို ရသည်။

Applications များတွင် RGB components ၃ မျိုး၏ proportion ကို affecting မပြုလုပ်ဘဲနှင့် colour ၏ intensity ကို ပြောင်းရန်လိုသည်။

Public Color brighter ( )

Color ၏ bright version တစ်ခုရရှိသည်။

Public Color darker ( )

Color ၏ darker version တစ်ခုကို ရရှိသည်။

သင်သည် regular RGB model အစား Hue, Saturation and Brightness (HSB) model ကို သုံး၍ အလုပ်လုပ်လျှင် Java ၏ Color class သည် conversion methods နှစ်မျိုးပေးသည်။

Public static Int HSB to RGB (float hue, float saturation, float brightness)

ပေးထားသည့် HSB color components များနှင့် ဆက်သွယ်သော default RGB Color Model ဖြင့် သတ်မှတ်ထားသည့် RGB value ရရှိသည်။

Public static float ( ) RGB to HSB (Int r, Int g, Int b, float ( ) hsbvals)

Red, Green, Blue components များနှင့် သတ်မှတ်သော color တစ်ခုကို ဆက်သွယ်ထားသည့် HSB values ကို ရရှိသည်။

အများဆုံး သုံးလေ့ရှိသော colours များကို Color class ၏ အစမှာ ၎င်းတို့၏ ဆက်သွယ်ထားသော RGB value များ ကို အသုံးပြု ဖော်ပြသည်။

Public final static Color black = new Color (0,0,0);

public final static Color blue = new Color (0,0,255);

public final static Color cyan = new Color (0,255, 255);

ဤ definitions များသည် အောက်ပါ example မှ ပြထားသော simple English names များကို အသုံးပြုရန် ခွင့်ပြုသည်။

```
import java.awt.*;
import java.applet.*;
   {
      public void paint(Graphics gg)
            {
            Font test4font new font ("Helvetica" Font.BOLD +
Font.ITALIC, 20);
            gg.setFont (Test4font);
            gg.setColor(Cplor.orange);
            gg.drawString("Sachin Tendulkari!", 5,225);
            }
   }
```

output သည် black and white ဖြင့် ပြသည်။

Sachin Tendulkar

GRAPHICS

Java မှာ graphics များနှင့် အလုပ်လုပ်သည့် အကောင်းဆုံး နည်းလမ်းတစ်ခု ရှိသည်။ ၎င်းသည် Graphics class မှ applet မှာ ပုံစံအမျိုးမျိုး ဆွဲရန် graphical context ကို သုံးသည်။ ၎င်းကို java.awt package တွင် အဓိပ္ပာယ်ဖွင့်ဆိုထားသည်။ သင် အခုလေးတင်ပဲ 'Sachin Tendulkar' string ကို ဆွဲသော enample ကို တွေ့ပြီးပြီ ဖြစ်သည်။ Graphics သည် component class ဖြင့်ပြုလုပ်ထားသော abstract class တစ်ခုပင် ဖြစ်သည်။ Applet class သည် Component class ၏ extension တစ်ခုဖြစ်သောကြောင့် သင်၏ ရေးဆွဲမှု (Graphics gg) method ကို applet တစ်ခုမှာ အဆင်ပြေစွာရေးနိုင်ခြင်း ဖြစ်သည်။ graphics class ဖြင့် ပြုလုပ်သော various methods list ကို အောက်တွင်ဖော်ပြထားသည်။ ၎င်းတို့၏ usage ကို small illustration ဖြင့် ပြထားသည်။

Java မှ coordinate system ၏ origin မှာ ဘယ်ဘက်ထောင့်ထိပ်တွင် ရှိသည်။ x axis သည် left မှ right သို့သွားသည်။ y axis သည် top မှ bottom သို့ သွားသည်။ ၎င်းသည် notebook's page မှ English text ရေးခြင်းနှင့် တူသည်။ အဆိုပါ system သည် geometry မှ coordinate system နှင့် opposite ဖြစ်သည်။ Geometric system မှ y axis သည် bottom မှ top ကို သွားသည်။

Public abstract void drawline
(Int x1,Int y1, Int x2, Int y2);

Coordinates (x1, y1) နှင့် (x2, y2) ကြားတွင် line တစ်ခုဆွဲသည်။ Line ကို ဖော်ပြထားသည့် coordinates ၏ ဘယ်ဘက်အောက်တွင် ဆွဲသည်။ Integer coordinates များသည် ဖော်ပြထားတဲ့အတိုင်း အတိအကျ match မဖြစ်နိုင်ပါ။

Public abstract voide fill Rect
(Int x, \ Int y, Int width, Int height);

Specified rectangel ကို current color နှင့် ဖြည့်သည်။

Public void drawRect (Int x, Int y, Intwidth, Int height)

Specified rectangle outline ကို current color နှင့် ဆွဲသည်။ Specified rectangle ၏ အတွင်းဘက် outline ကို ဆွဲလိုလျှင် drawRect (x,y, width-1, height -1) ကို သုံးပါ။

Public abstract void clearRect
(Int x, Int y, Int width, Int height);

Specified rectangle ကို current drawing surface ၏ current background color နှင့် ခြယ်ပြီး clear ပြုလုပ်သည်။ ရွေးချယ်ထားသော drawing surface သည် graphics context တည်ဆောက်မှုပေါ်တွင် မူတည်သည်။

Public abstract void draw Round Rect (Int x, Int y, Int width, int height, Int arcWidth, Int arcHeight);

Rectangle ၏ outlined rounded corner ကို current color နှင့် ဆွဲသည်။

Public abstract void fillRound Rect (Int x, Int y, Int width, Int height, Int arcWidth, Int arc Height);

Rounded rectangle တစ်ခုကို current color နှင့် ဖြည့်ဆွဲသည်။

Public void draw 3D Rect (Int x, Inty y, Int width, int height, Boolean raised)

Highlighted 3-D rectangle တစ်ခု ဆွဲသည်။

Public void fill 3Drect
(Int x, Int y, int width, int height, Boolean raised);

Highlighted 3-D rectangle ကို current color သုံးပြီး ဆေးခြယ်သည်။

Public abstract boolean draw Image
(Image Img, Int x, Int y, ImageObserver observer);

Specified coordinate (x,y) တွင် specified image တစ်ခုဆွဲသည်။ Image သည် incomplete ဖြစ်လျှင် image observer က notify ပြုလုပ်လိမ့်မည်။

Public abstract boolean draw Image (Image img, int x, int y, Int width, Int height, Image Observer observer);

Specified image ကို specified rectangle အတွင်းမှာ ဆွဲသည်။ လိုအပ်လျှင် image ကို စကေးသတ်မှတ်တယ်။ image သည် incomplete ဖြစ်လျှင် image observer က notify ပြုလုပ်လိမ့်မည်။

Pubic abstract boolean draw Image (Image img, intx, inty. Color bgcolor, ImageObserver observer); Specified image ကို specified coordinates (x, y) တွင် ပေးထားသော solid background Color နှင့်ဆွဲသည်။ Image သည် incomplete ဖြစ်လျှင် image observer သည် notify ပြုလုပ်လိမ့်မည်။

Public abstract boolean draw Image (Image Img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer);

Spcified image ကို specified rectangle အတွင်းမှာ ပေးထားသည့် solid background Color နှင့် ဆွဲသည်။ လိုအပ်လျှင် image ကို စကေး သတ်မှတ်ရမည်။ Image သည် incomplete ဖြစ်လျှင် image observer သည် notify ပြုလုပ် လိမ့်မည်။ Graphic context ကို dispose ပြုလုပ်ရမည်။ dispose ပြုလုပ်ပြီးနောက် Graphics context ကို မသုံးနိုင်တော့ပါ။

Public abstract void dispose ( );

Graphics context ကို တစ်ကြိမ် disposes off ပြီးပါက နောက်ထပ် ဖော်ပြရန် မလိုတော့ပါ။

graphics methods အသုံးပြုထားတဲ့ ဥပမာတစ်ခုကို ကြည့်ပါ။

```
import  java.applet .*;
import  java.awt .*;
public class test1 extends java.applet.Applet
   {
   public void paint(Graphics gg)
      {
      gg.setColor(Color.red);
      gg.drawString("Sachin Tendulkar!",500,500);
      gg.drawRect (30, 30, 40, 50)
      gg.fillRect(130, 130, 140, 150)
      gg.drawOval(30, 30, 40, 50);
      }
   }
```

သင့်၏စက်မှာ ၎င်းကိုကူးပြီး ကြိုးစားကြည့်ပါ။

WINDOWS ENVIRONMENT

Java သည် windows environment ကိုလည်း အထောက်အကူပေးသည်။ ထို environment program တွင် logic သည် most procedural code ဖြင့် ပြုလုပ်ထားသည့်အတွက်ကြောင့် program ၏ top မှ bottom သို့ မသွားပါ။ Operating system သည် events များကို collect ပြုလုပ်ပြီး program က ၎င်းတို့ကို responds ပြုလုပ်ပေးသည်ယ်။ Events များကမူ mouse clicks, key presser, Ethernet port ပေါ် ရောက်ရှိနေသော network data သို့မဟုတ် အခြားဖြစ်နိုင်သော အရာများဖြစ်သည်။ Operating system သည် event တစ်ခုစီကို ကြည့်သည်။ ထို့နောက် သင့်လျော်သော program's event queue မှာ event ကို နေရာချပေးရန်အတွက် မည်သည့် program ကို သုံးမည်ကို ဆုံးဖြတ်သည်။ Application program တိုင်းမှာ event loop တစ်ခုရှိသည်။ ၎င်းသည် အဆက်မပြတ် ဖြစ်စေသော while loop တစ်ခုသာ ဖြစ်သည်။ Loop ကို ဖြတ်သန်းစဉ် application သည် ၎င်း၏ event queue မှ next event ကို ရရှိပြီးသည်နှင့် တစ်ဆက်တည်း responds ပြုလုပ်သည်။ Java applet သည် environment မှ event များကို နေရာတကျ ဖြစ်အောင် responds ပြုလုပ်သည်။ runtime environment မည်သို့ပင်ဖြစ်စေ၊ browser သည် applet အတွက် event loop ကို ဖော်ဆောင်တဲ့အတွက် တိကျစွာ သီးသန့်တစ်ခုရေးရန် မလိုတော့ပါ။

EVENTS

Applet တစ်ခုရေးနေစဉ်မှာ appletsများသည် eventsများနှင့်သာဖြစ်ပေါ်စေသည်။ Event တစ်ခုဖြစ်ပေါ်သည့် အချိန် တစ်ခုစီမှာ သင့်ရဲ့ appletသည် ၎င်း၏ ဆောင်ရွက်မှုကို notify ပြုလုပ်တယ်။ Event occurance တစ်ခုသည် ဘာကို ပြုလုပ်သလဲဆိုခြင်းကို အောက်တွင် ရှင်းပြထားသည်။ Events များသည် အမျိုးမျိုးသောအရာများ applet, window, mouse နှင့် keyboard များမှ ဖြစ်ပေါ်လာသည်။

APPLET EVENTS

Initialization

သင်၏ applet အသက်ဝင်လာသည်နှင့် Initializationဖြစ် ပေါ်လာသည်။ ၎င်းက applet တစ်ခုခုကို ဖြစ်ပေါ်လာနိုင် first event ပဲ ဖြစ်ပါသည်။

Destory

၎င်းကတော့ Initialization နဲ့ opposite ပါပဲ။ Applet တစ်ခုခုမှာ last event ကြောင့် ဖြစ်လာတယ်။ Final clean up ပြုလုပ်လိုလျှင်တော့ ၎င်းကို သုံးမယ်။

Stop

Stop eventက userမှ appletပါဝင်သည့် pageကို ဆက်ကြည့်ရန် မလိုတော့ဘူး ဆိုရင် ညွှန်ပြသည်။ ၎င်းသည် userကို pageမှ ထွက်ခွာလိုလျှင် သို့မဟုတ် window အား minimize ပြုလုပ်လိုလျှင် သုံးပါသည်။

Start

Start event သည် user မှ applet ကို ဝင်ရောက်လိုသည့်အခါ သုံးသည်။ Page ကို ဝင်ရောက်ရန် window ကို maximize ပြုလုပ်ပြီးနောက် ဖြစ်စေသည်။ ၎င်းသည် initialization event ပြီးနောက် ဖြစ်ပေါ်စေသည်။

WINDOW EVENTS

Paint

Paint event သည် fully covered ပြုလုပ်ထားသည့် applet window ကို uncovered ပြုလုပ်တဲ့အခါ ဖြစ်ပေါ်လာသည်။ ၎င်းသည် start event ပြီးနောက် ဖြစ်လာသည်။ သင် paint method ကို မြင်ပြီးပြီ။ Drawing အားလုံးကို ပြုလုပ်သည့် method ဖြစ်သည်။ Paint method မှ applet screen မှာသာ ရေးနိုင်သည်။

MOUSE EVENTS

Mouse Up

Mouse Up event သည် သင်၏ applet မှ mouse button ကို released ပြုလုပ်သည့်အခါ ဖြစ်ပေါ်လာသည်။ Case အများစုမှာတော့ ၎င်းသည် mouse down ကိုကြည့်ပြီး ဖြစ်လာစေသော event ဖြစ်သည်။ Screen မှ button ၏ symbol တစ်ခုသည် mouse button ကို symbol ပေါ်မှာ နှိပ်သည့်အခါ typically highlighted ဖြစ်လာသည်။ သို့သော် ၎င်းသည် mouse button ကို release ပြုလုပ်စဉ်မှာ activate မဖြစ်ပါ။ ၎င်းသည် userအတွက် mouse buttonကို release မပြုလုပ်ဘဲနှင့် button symbol မှ cursor ကို ရွှေ့စေဖို့ စိတ်ကြိုက် အခွင့်အရေးပေးသည်။ Mouse up event သည် mouse ကို released ပြုလုပ်တဲ့နေရာမှာ point ၏ coordinates များကိုလည်း offer ပြုလုပ်ပေးပါသည်။

example,

```
public  boolean mouseUp(Event e, int x, int y) {
theList.addItem("mouseUp event at ("+x+","+y+")");
return false;
}
```

Mouse Down

Mouse Down Event သည် mouse button ကို သင့် applet မှာ press ပြုလုပ်သည့်အခါ ဖြစ်လာသည်။ Caseအများစုမှာ any action မဖြစ်ခင် ဖြစ်လာစေသည့် mouse up event အတွက် စောင့်ပြီး ပြုလုပ်ပေးသည်။ Mouse down event သည် mouse ထိစပ်စေသောနေရာမှာ point ၏ coordinates များကို ဖြစ်စေသည်။

example,

```
public  boolean mouseDown(Event e, int x, int y) {
theList.addItem("mouseDown event at (+x+","+y+")");
returb false
}
```

Mouse Drag

Mouse Drag event သည် user မှ mouse button ကို hold down ပြုလုပ်၍ mouse ကို ရွှေ့သည့်အခါ ဖြစ်လာသည်။ ၎င်း event သည် mouse ရှိသည့် နေရာမှာ eventဖြစ်စေသည့်အခါ point ၏ coordinates များကိုလည်း ဖြစ်စေသည်။ ယေဘုယျအားဖြင့်တော့ သင့်လက် ရွှေ့သည့်အခါများတွင် mouse drag event sequence တစ်ခုကို ရရှိပါသည်။

example,

```
public  boolean mouseDrag (Event e, int x, int y){
theList.addItem("mouseDrag event at ("+x+","+y+")');
return falst;
}
```

Mouse Move

Mouse move event သည် user တစ်ယောက်မှ mouse button အား holding down မပြုလုပ်ဘဲနှင့် mouse ကို ရွှေ့စေသည့်အခါ ဖြစ်လာသည်။ ၎င်းသည် mouse ရှိတဲ့နေရာမှာ mouse event ဖြစ်လို့သည့်အခါ point ၏ coordinates များကို ဖြစ်စေသည်။ ယေဘုယျအားဖြင့် သင့်လက်ရွှေ့တဲ့အခါ mouse move events sequence တစ်ခုကို ရရှိသည်။

example,

```
public  boolean mouseMove(Event e, int x, int y) {
theList.addItem("mouseMove event at ("+x+","+y+")");
return false;
}
```

Mouse Enter

သင့် applet window ကို တစ်နေရာမှ cursor ရောက်ရှိသည့်အခါ သင့် applet သည် mouse enter တစ်ခုကို ရရှိသည်။ Applet ကို ဝင်ရောက်လာစေသော cursor နေရာမှာ point ၏ coordiniates များကိုလည်း ရရှိသည်။ applet မှတစ်ဆင့် ဖြစ်လာသော cursor ကြောင့် mouse moved events sequence တစ်ခုဖြင့် type ပြုလုပ်သည်။

example,

```
public  boolean mouseEnter (Event e, int x, int y) {
theList.addItem(mouseEnter event at ("+x","+y+")");
return false;
}
```

Mouse Exit

သင့် applet မှ cursor ကို ဝင်ထွက် သွားသည့်အခါ mouse exit event တစ်ခုကို ရရှိသည်။ သင့် applet မှ cursor ထွက်သည့်နေရာမှာ point ၏ coordinates ကိုလည်း ရရှိသည်။

example,

```
public  boolean mousexit(Event e, int x, int y) {
theList.addItem("mouseExit at "x+","+y+")")
return false;
```



```
}
```

get focus

```
public void getFocus( ) {
theList.addItem("getFocus event");
}
```

gotFocus

```
public void gotFocus( ) {
theList.addItem("gotFocus event");}
```

lostFocus

```
public void lotFocus( ) {
theList.addItem("lostFocus event");
```

KEY BOARD EVENTS

Key Down

Key Down event တစ်ခုသည် သင့် applet သည် active ဖြစ်နေစဉ်တွင် user မှ key တစ်ခုကို press ပြုလုပ်သည့်အခါ ဖြစ်လာစေသည်။ Integer key code တစ်ခုသည် press ပြုလုပ်ရမည့် key အား ညွှန်ပြရန် အသုံးဝင်သည်။ General rule အနေဖြင့် actual character ရရှိရန် ၎င်းကို char တစ်ခုသို့ ပြောင်းရမည်။

example,

```
public  boolean keyDown(Event e, int x) {
theList.addItem("The"+(char)x+ "key was pressed.");
return false;
}
```

Key Up

Key up event တစ်ခုသည် user က pressed key ကို release ပြုလုပ်သည့်အခါ ဖြစ်လာစေသည်။ Integer key code တစ်ခုသည် press ပြုလုပ်သည့် key ကို ညွှန်ပြပြီး ပြန်ရရှိစေသည်။ General rule တစ်ခုအနေဖြင့် actual letter ရရှိရန် ၎င်းကို char သို့ ပြောင်းရမည်။

```
public  boolean keyUp(Event e, int x) {
thelist.addItem("The"+(char)x+"key was raised.");
return false;
}
```

EVENTS HANDLING

Java ရဲ့ event handling mechanism သည် အနည်းငယ် ရှိးပါသည်။ အခြား events များကဲ့သို့ပင် Java မှ events များသည် Java developer's kit (JDK) ၏

အစိတ်အပိုင်းအဖြစ် class ပုံစံများဖြစ်သည်။ ၎င်းသည် java.awt package တွင် ရှာသည်။ (awtသည် abstract windowing toolkitအတွက် ရပ်တည်သည်)။ event တစ်ခုခုဖြစ်တဲ့အခါ system သည် class Event objectတစ်ခုတည်ဆောက်ခြင်းဖြင့် program ကို အချက်ပေးသည်။ ပြီးလျှင် postEvent (Event e){...} ဟုခေါ်သော method တစ်ခု ရရှိစေသည်။ ဤအချက်သည် ၎င်းနှင့် ဆက်စပ်နေသော method ကို check ပြုလုပ်သည်။ ပြီး handle ပြုလုပ်ထားသည်။ ထို့ကြောင့် ၎င်းသည် အခြား method ဖြစ်သော handle Event (Event e) {...} ကို ရရှိစေသည်။ Handle Event methodတွင် different casesများအတွက် event handlersဟုခေါ်သော different methods များဖြစ်ပေါ်စေသည့် long switch statement တစ်ခု ရှိသည်။

ဤ event handlers အများစုသည် empty body ဖြစသည့် methods များဖြစ်ကြသည်။ ၎င်းအား real handlingတစ်ခုခု မပြုလုပ်ဘဲနှင့် event ကို မဖြစ်စေပါ။ ၎င်းမှာ Javaမှ ပြုလုပ်သော default actionဖြစ်သည်။ ယခု သင့် applet fileမှ class hierarchy ကိုကြည့်မည်။ သင့် appletတစ်ခုရေးသည့်အခါ Appletကို ချဲ့ထွင်သည်။ Appletသည် source codeရဲ /java/ applet subdirectoryတွင်ပါဝင်သော Panel file ကိုချဲ့ထွင်တယ်။ Panel သည် Container ကို ချဲ့ထွင်သည်။ Container သည် Componentကို ချဲ့ထွင်သည်။ ဤ Panel, Containerနှင့် Component classes များ အားလုံးသည် java. awt package တွင် ပါဝင်သည်။

awt packageမှာ various classesများမှ basic classတစ်ခုအဖြစ် Component ရှိသည်။ ထို့ကြောင့် methods အားလုံးနှင့် Component data, Container, Panel နှင့် Appletများသည် သင်ရေးထားသည့် appletတစ်ခုခုတွင် အသုံးဝင်သည်။ ဥပမာ– test1, test2နှင့် အခြားအမည်များဖြစ်မည်။ post Eventနှင့် handle Event method 2 ခုသည် awt package ၏ Component class မှာရှိသည်။ ဤဟာများသည် Container, Panel, Appletနှင့် သင့် test 1 classes များ၏ method အဖြစ် အသုံးပြုသည်။ ယေဘုယျအားဖြင့်တော့ default action သည် empty သို့မဟုတ် no action ပဲဖြစ်သည်။ သင်ဆန္ဒရှိလျှင် ဤ methods များမှ တစ်ခုခုကို override ပြုလုပ်နိုင်သည်။ Override ပြုလုပ်ရန် နည်းလမ်းကတော့သင့် class methodကို re-defineပြုလုပ်ရုံပဲဖြစ်သည်။ တစ်ခုတည်းသော Constraintမှာ ၎င်း၏ signature သည် original method ၏ signature နှင့် match ဖြစ်သင့်သည်။ အောက်တွင် အမျိုးမျိုးသော event handler methodsများ၏ default codeအပြည့် အစုံနှင့် တစ်ခုစီ၏ description ကို လေ့လာနိုင်ပါသည်။

APPLET METHODS

Public void Init ( ) { }

Applet ကို initializeပြုလုပ်သည်။ သင်သည် ၎င်းကို မည်သည့်အခါမျှ directly

ခေါ်ရန်မလို၊ appletတစ်ကြိမ် တည်ဆောက်လိုက်ရုံဖြင့် systemက automatically ခေါ်ပြီးပြီ ဖြစ်သည်။

Public void start ( ) { }

Appletကို startပြုလုပ်သည်။ ဤ methodကို dir-ectlyခေါ်စရာမလို။ Applet ရဲ့ documentရောက်ရှိလာသည်နှင့် သို့မဟုတ် ၎င်း၏ windowသည် visibleဖြစ်လာတာနဲ့ ခေါ်ပြီးသား ဖြစ်သည်။

Public void stop ( ) { }

Applet ကို stopပြုလုပ်ရန် ဖြစ်သည်။ Screenပေါ်တွင် applet's document ကို မရှိစေချင်တော့သည့်အခါမှာ ၎င်းကို ခေါ်သုံးသည်။ ၎င်းသည် destroy () ကို မခေါ်ခင်မှာသုံးသည်။ ဤ methodကို မည်သည့်အခါမျှ directly ခေါ်ရန်မလိုပါ။

Public void destroy ( ) { }

သုံးထားတဲ့ resourcesများကို cleans upပြုလုပ်သည်။ Applet activeဖြစ်လျှင် ၎င်းသည် ရပ်သွားသည်။

WINDOW METHODS

Public void paint (Graphics g) { }

Componentကို paintပြုလုပ်သည်။ သင် paint methodကို သိပြီးပြီ။ သင်၏ drawing အားလုံးကို ပြုလုပ်သည့်နေရာမှာ ၎င်း methodကို သုံးတယ်။ Paint method တွင် applet screenတစ်ခုမှာ ရေးရန် သုံးနိုင်သည်။

MOUSE METHODS

Public boolean MouseDown (Event evt, Int x, Int y) { return false;}

Mouse down လုပ်သည့်အခါ သုံးသည်။

Public boolean mouseDrag (Event evt, Int x, Int y) { return false;}

Mouse button ကို downပြုလုပ်ပြီး draggedပြုလုပ်လိုသည့်အခါ သုံးပါသည်။

Public boolean Mouse Up (Event evt, int x, int y) {return false;}

Mouse up ပြုလုပ်သည်။

Public boolean mousemove (Event evt, Int x, Int y) {return false;}

Mouse button ကို up ပြုလုပ်၍ move ပြုလုပ်မည်။

Public boolean mouse Enter (Event evt, Int x, Int y) {return false;}

Mouse အား component ကို ဝင်လာစေလိုသည့်အခါ သုံးသည်။

Public boolean mouse Exit (Event evt, Int x, int y) {return false;}

Mouse အား component မှ ထွက်စေလိုသည့်အခါ သုံးသည်။

KEY BOARD METHODS

Public boolean KeyDown (Event evt, Int Key) {return false;}

Character ကို press ပြုလုပ်စေသည်။

Public boolean KeyUp (Event evt, Int key) {return false;}

Character ကို release ပြုလုပ်စေသည်။

ဥပမာ

အောက်တွင်ဖော်ပြထားသည့် applet သည် programming မှ event အချို့ကို ဖြစ်စေရန် designပြုလုပ်ထားပြီး eventsအမျိုးမျိုးသည် တစ်ခုပြီးတစ်ခု ဖြစ်ပျက်ကြသည်။

Event တစ်ခုဖြစ်ပေါ်သည့်အခါ applet သည် usual command line မှ event ရဲ့ name ကို printing ပြုလုပ်ပြီး တုံ့ပြန်သည်။

```
import  java.applet.Applet;
import java.awt.*;
public class test1 extends Applet
   {
   public void init( )
```



```
    {System.out.println ("init event");}
 public void paint(Graphics g)
    {System.out.println ("paint event");}
 public void start( )
    {System.out.println ("start event");}
 public void destroy( )
   {System.out.println ("destroy event");}
public void update (Graphics g)
   {System.out.println ("update event");}
public boolean mouseUp
   (Event et int xi int y)
   {System.out,printiln ("mouseup event");
     return false;}
public boolean mouseDown
   (Event et int xt int y)
   {System.out.println ("mouseDown");
     return false;}
public boolean mouseDrag
   (Event et int xt int y)
   {System.out.println ("mouseDrag event");
     return false;}
public bollean mouseMove
   (Event et int xt int y)
   {System.out.println ("mouseMove event");
     return false;}
public boolear. mouseEnter          (Event et int xy int y)
   {System.out.println (mouseEnter event");
     return false;}
public boolean mouseExit
   (Event et int xt inl y)
   {System.out.println ("mouseExit event");
     return false;}
public boolean keyDown
   (Event et int x)
   {System.out.println ("keyDown event");
     return true;} }
```

ဒင်းနှင့် ဤ applet play ကို တစ်ကြိမ် compiled, loaded ပြုလုပ်ပြီးထားပြီ။ Applet window မှာ mouse ကို click နှိပ်ပါ။ Mouse ကို double click နှိပ်ပါ။ Textအချို့ကို ရိုက်ကြည့်ပါ။ Browser windowကို resizeပြုလုပ်ပါ။ ဒင်းကို cover ပြုလုပ်ပြီး uncoverပြုလုပ်ပါ။ ဒင်းကို ပြုလုပ်နေစဉ် standard outprut ကို ဂရုတစိုက်ကြည့်ပါ။

HYPER TEXT MARKUP LANGUAGE (HTML)

Hyper text သည် web page ပေါ်မှာပြသော matter ကို ကိုယ်စားပြုပါသည်။ ဒင်းသည် textသာမက graphics, a title for the page နှင့် textအတွင်းမှ markers အချို့ဖြစ်သော highlighted text သို့မဟုတ် icons များ ပါဝင်ပါသည်။ သင်သည်

marker တစ်ခုပေါ်မှာ click ပြုလုပ်သည့်အခါ HTML သည် တည်ဆောက်ထားသည့် reference တစ်ခုဖြစ်သော applet သို့မဟုတ် အခြား page ဆီကို ရောက်ရှိသွားသည်။ ပြီးလျှင် အဲဒီ page သို့မဟုတ် applet သည် သင့်ရှေ့သို့ pop up ပုံစံမျိုးရောက်လာသည်။ အချို့ text သို့မဟုတ် graphical image တစ်ခု သို့မဟုတ် အခြား web page (static) များသည် ဤပုံအတိုင်းဖြစ်သည်။ Java နှင့် ရေးထားတဲ့ applet တစ်ခုဖြစ်လျှင် ၎င်း၏ code သည် executed ပြုလုပ်ပြီးလျှင် results ကို ပြသည်။

HTML FILES

သင်သည် HTML files မှ <APPLET> tag နှင့်အတူ applet တစ်ခုကို ကိုယ်စားပြုရာတွင် မှတ်ရန် နှစ်ချက်ရှိသည်။ သင် applet ပါဝင်သော class name ကို ညွှန်ပြရန် CODE attribute ကို သုံးပါ။ Applet ၏ size ကို ညွှန်ပြရန် WIDTH နှင့် HEIGHT ကိုသုံးပါ။ Browser သည် page ပေါ်မှ applet အတွက် နေရာအကျယ်ကို သိရှိရန် values ကို သုံးပါသည်။ box တစ်ခုသည် 150 pixels wide နှင့် 25 pixels high နှင့် တည်ဆောက်သည်။

HTML file တစ်ခုမှာ extension . html ဖြစ်ပြီး ၎င်းသည် ယေဘုယျအားဖြင့် သင့် applet ရှိတဲ့ directory မှာပဲ သိမ်းသည်။ ၎င်းသည် အဓိကမကျသော်လည်း directories များ ခွဲခြားထားခြင်းသည် မလိုအပ်သည်များ ထုတ်ပေးခြင်းကို တားဆီးနိုင်ပါသည်။ HTML file တစ်ခုတွင် </HEAD> နှင့် <HEAD> ကဲ့သို့ various tag pairs များ ပါဝင်သည်။ ပထမတစ်ခုမှာ opening bracket ဖြစ်ပြီး ပါရှိသော အခြားတစ်ခုမှာ closing bracket ဖြစ်သည်။ တစ်စုံပြည့်မှ complete text block ဖြစ်သည်။ Blocks အမျိုးမျိုး ရှိသည်။

<HTML>...</HTML>	The main html block.
<HEAD>...</HEAD>	The heading block.
<TITLE>...</TITLE>	The tite block.
<BODY>...</BODY>	The main body block.
<APPLET>...</APPLET>	The applet block.
<IMG>...</IMG>	The image block.
<B>...</B>	The bold text block.
<I>...</I>	The italic text block.

HTML block တွင် entire text ရှိရမည်။ ၎င်းအတွင်းမှာ HEAD နှင့် BODY blocks နှစ်ခုရှိသည်။ HEAD block အတွင်းမှာ page အတွက် title နှင့် heading အတွက် အခြား attributes များပါမည်။ Title ရဲ့ text သည် TITLE block အတွင်းမှာ ရှိသည်။ BODY block မှာ IMG block မှ image များ၊ APPLET blocks မှ applets များ အစရှိသဖြင့် text matter အားလုံးပါဝင်သည်။ Bold or italic text မှာ B or I block တွင် ရှိရပါမည်။

Block နှင့်စသော APPLET သည် ရှည်လျားသည့် statement တစ်ခုဖြစ်နိုင်သည်။

ဥပမာ– <APPLET CODE = "test1. class" WIDTH = 150 HEIGHT = 25>

ဤ statement တွင် tags အမျိုးမျိုး ပါဝင်ပြီး ၎င်းတို့ကို values assigns လုပ်ထားသည်။ Tag CODE သည် "test1.class" value ဖြစ်ပြီး applet အတွက် code ရှာရန် နေရာကို ပြသည်။ WIDTH tag သည် 150 value ဖြစ်ပြီး HEIGHT tag သည် 25 value ဖြစ်သည်။ ၎င်းတို့သည် web page ပေါ်မှာ ပေါ်လာသော marker ၏ width နှင့် height ဖြစ်သည်။ သင်သည် applet's. class file ကို html file အဖြစ် same directory မှာ မသိမ်းခဲ့လျှင် ၎င်းကိုရှာရမည့် နေရာအတွက် system ကောပြရမည်။ CODE-BASE para meter ကိုသုံးပြီး ဖော်ပြတယ်။ File သည် သင်၏ directory မှာရှိလျှင် သင့် html file မှ applet tag line တွင် သွားပေါင်းနိုင်သည်။

PASSING PARAMETERS TO APPLETS

Opening နှင့် closing APPLET tag ကြားမှ area သည် applets များကို parameters သတ်မှတ်ရန်လည်း သုံးနိုင်သည်။ Applet ကို သတ်မှတ်သော paramcters များသည် PARAM ရဲ့ seperate tag တစ်ခုအောက်မှာ ဖြစ်သည်။ ၎င်းသည် <APPLET> နှင့် </APPLET> ကြားမှာ ဖြစ်ပေါ်သည်။ ၎င်းမှာ NAME နှင့် VALUE ဟုခေါ်သော ကိုယ်ပိုင် parameters နှစ်ခုရှိသည်။ NAME သည် get Parameter method အတွက် parameter ကို အမည်သတ်မှတ်သည်။ VALUE သည် String တစ်ခုအဖြစ် parameter ၏ value ဖြစ်သည်။ နှစ်ခုစလုံးသည် အခြား HTML tag parameters များကဲ့သို့ double quote marks ဖြင့် ဖော်ပြသည်။ ၎င်းကို demonstrate ပြုလုပ်ရန် သင်သည် Java for You ကို applet နှင့်ရေးဆွဲထားသော generic string တစ်ခုသို့ ပြောင်းရပါမည်။ ထိုသို့ပြုလုပ်ရန် ရေးဆွဲမည့် string ကို ဖော်ပြ သော applet parameters ကို သတ်မှတ်ဖို့ လိုပါသည်။

```
import  java.applet. Applet;
import java.awt.graphics;
public class test1 extends java.applet.Applet
  {
  String inpu-from-page;
  public void init( )
    {
    input-from-page =
      getParameter ("String");
    }
  public  void paint(Graphics gg)
  {
```

```
    gg.drawString(input-from-page, 50, 25);
    }
}
```

ယခု HTML file ဖြစ်သော test1.html ကို <APPLET> နှင့် </APPLET> ပါဝင်သော lines များကြားတွင် အောက်ဖော်ပြပါ လိုင်းတစ်လိုင်း အစားသွင်းခြင်းဖြင့် modify ပြုလုပ်ကြည့်ပါ။

<PARAM NAME = "string" VALUE = "Who are you?">

သင် "Who are you?" ရဲ့ မတူညီသော version တစ်ခုကို ထပ်ထည့်ရမယ်။ မှတ်ရန်မှာ ၎င်းသည် code ကို changing သို့မဟုတ် recompiling မပြုလုပ်ဘဲနှင့် applet ၏ output ကို ပြောင်းလို့ရပါသည်။ သင်ဟာ parameter တစ်ခုတည်းသုံးရပါမည်၊ တစ်ခုထက်ပိုသုံးရမယ်ဟု ကန့်သတ်ထားစရာ မလိုပါ။ သင်နှစ်သက်သလို applet တစ်ခုကို parameters များစွာ သတ်မှတ်ပေးနိုင်သည်။ get parameter method သည် straight forward ဖြစ်သည်။ သင်သည် ၎င်းကို သင်လိုချင်သည့် parameter name string တစ်ခုကို ပေးနိုင်သည်။ Parameter value string တစ်ခုကို ပြန်ရပါမည်။ Parameters အားလုံးကို strings များအဖြစ် သတ်မှတ်တာပါပဲ။ သင်ဟာ တစ်စုံ တစ်ခု ရချင်သည်။ သို့မဟုတ် integer တစ်ခုရချင်လျှင် ၎င်းကို String တစ်ခုအဖြစ် သတ်မှတ်ရမပါမည်။ ပြီးလျှင် ၎င်းကို သင်ဖြစ်ချင်သည့် type ကို ပြောင်းပေးရပါမည်။

နောက်လာမည့် Graphics using Applets applets ၏ အရေးကြီးတဲ့ အပိုင်းကို demonstrate ပြုလုပ်ရန် ပိုမိုကောင်းမွန်သည့် applets examples များကို ဖော်ပြပါမည်။

အခန်း (၅)
GRAPHICS USING APPLETS

Applets များသည် အမျိုးမျိုးသော objects များကို ဆွဲရန်အတွက် အလွန် အသုံးဝင်ပါသည်။ Applets အများစုသည် net ပေါ်တွင်လည်းကောင်း၊ သင့် applet directory တွင်လည်းကောင်း အသုံးပြုရန် အသင့်ဖြစ်သည်။ ၎င်းတို့အနက် အနည်းငယ်ကို ဖော်ပြပါမည်။

Drawing Rectangles

နှစ်သက်ရာ size သတ်မှတ်နေရာပေးထားသော rectangles များနှင့် screen ကို ပြည့်စေသော applet တစ်ခုကို ရေးကြစို့။ ဤ process တွင် ကျွန်ုပ်တို့ applet graphics ရဲ့ အခြေခံများကို လေ့လာရပါမယ်။ ကျွန်ုပ်တို့ bit တစ်ခု ပေါင်းထည့်သည့် တစ်ကြိမ်မှာ step တစ်ခု ရှိရမည်။ Programs များကို သုံးပြီး ၎င်းတို့ကို Mondrian 1 မှ ရှေ့သို့ တစ်ခုစီတိုးပြီး အမည်သတ်မှတ်မယ်။ First applet မှာတော့ screen မှ rectangle တစ်ခုသာ ဆွဲမယ်။ HTML file မှ ဖော်ပြထားသည့်အတိုင်း applet size ကို ယူမည်။ ပြီးလျှင် ၎င်းကို frame ပြုလုပ်ရန် applet ကို ပတ်၍ rectangle တစ်ခုကို ဆွဲမည်။ code ဖြင့် ဖော်ပြလျှင်

```
// Draw a rectangle

import java.applet.*;
import java.awt.*;

public class Mondrianl extends Applet {

int height,width;

public void init ( ){
Dimension d = size ( );
height = d.height;
width = d.width;
repaint ( );

}

public void paint(Graphics g) {
```

```
g.drawrect (0,0, height, width);
}
}
```

ဤ applet ကို compile ပြုလုပ်ပါ။ သင်၏ classes directory ဆီသို့ resulting classfile ရောက်ရှိသွားပြီး၎ င်းကို point လုပ်သော HTML file တစ်ခု create လုပ်ပါ။ Applet ၏ height ကို 300 pixels နှင့် width ကို 300 pixels မှာ ကောင်းစွာ နေရာယူသည်။ ၎ င်း file ကို သင့် browser ဆီသို့ (load) ဆွဲတင်တဲ့အခါ သင် ဘာမြင်ရမလဲ။ သင် ကောင်းစွာ မျှော်မှန်းနိုင်မည် မဟုတ်ပါ။ Rectangle တစ်ဝက်ကို သင် မြင်ရပါမည်။ အခြားတစ်ဝက်က ဘာဖြစ်သွားပါသနည်း။ ၎ င်းကို fencepost error တစ်ခုဟု ခေါ်တယ်။ Applet သည် 300 pixels tall နှင့် 300 pixels wide ရှိ စတုရန်း တစ်ခု အတွင်းမှာ ရှိသည်။ ဘယ်လိုပဲဖြစ်ဖြစ် applet ၏ upper left hand corner သည် (0,0) မှစသည်။ (1,1) မှ မစပါ။ ဤဟာသည် applet မှာ 0 နှင့် 299 ကြားမှ x, y coordinates များ။ points များ ပါဝင်ပြီး 0 နှင့် 300 ကြားမှ points များ မပါဝင်ပါဟု ဆိုလိုသည်။ ကျွန်ုပ်တို့ဟာ 301 pixels hight နှင့် 301 pixels wide နှင့် rectangle တစ်ခုဆွဲခဲ့လျှင် အစွန်းများသည် ဘောင်ကျော်သွားသည်။ ထိုအခါ ကျွန်ုပ်တို့ကို error ၊ ဒါမှမဟုတ် တစ်စုံတစ်ခု ပြလာမည်။ Java မှာ applet တစ်ခုအတွက် coordinate system သည် upper left hand corner မှစပြီး right and down သို့ တိုးသွားသည်။ ၎ င်းသည် computer graphics များတွင် တူညီသော်လည်း y ၏ increasing direction သည် ပုံမှန်အပေါ်သို့ တိုးသွားသော Cartesian coordinate system နှင့်တော့ ခြားနားသည်။ ဘောင်ကျော်သွားသော အမှားအား ပြင်ဆင်ရန်မှာ လွယ်ကူပါသည်။ g.drawRect (0,0, height, width), ကို g.drawRect (0, 0 , height-1, width-1); သို့ပြောင်းပြီး ပြန်ရေးကြည့်ပါမည်။

```
// Draw a rectangle
import java.applet.*;
import java.awt.*;

public class Mondrian2 extends Applet {

int height, width;

public void init( ) {

Dimension d = size( );
Height = d.height;
Width = d.width;
repaint( );

}

public void paint (Graphics g) {
```



```
g.drawRect(0.0, height-1, Width-1);

}

}
```

၎ င်းကို compile ပြုလုပ်ပြီး သင့် browser ဆီသို့ ဆွဲတင်ကြည့်ပါ။ Problem မပြေလည်သေးလျှင် သေချာအောင် check လုပ်ပြီး new class file ကို classes directory တွင် သိမ်းပြီးနောက် HTML file ကို Mondrian 2 ကို ညွှန်းရန် modify လုပ်ပါ။ ကျွန်ုပ်တို့ Graphics class မှ method တစ်ခုဖြစ်သော drawRect အား new statement တစ်ခုအဖြစ် မိတ်ဆက်ပြီးပြီ။ g.drawRect (0,0, height-1, width-1) လိုင်းသည် point (0,0) တွင် စပြီး point (299,299) တွင် ဆုံးသော rectangle တစ်ခု ဆွဲရန် Graphics calss g ကို ညွှန်ပြသည်။ ဤ rectangle သည် applet ရဲ့ visible space တစ်ခုလုံးကို ဖော်ပြသည်။ Applet ရဲ့ အပြင်ဖက်ကို မထွက်စေပါ။ မှတ်ရန်မှာ rectangle ကို (300,300) သို့ အမှန်တကယ် ချဲ့ထွင်သော first version မှာ ဆွဲ၍ ရနိုင်သော်လည်း user မှ မမြင်နိုင်ပါ။

drawRect Method သည် open rectangle တစ်ခုကို ဆွဲသည်။ Filled rectangle တစ်ခုကို ဆွဲချင်လျှင် fill Rect method ကိုသုံးသည်။ Syntax ကတော့ အတူတူပါပဲ။ Mon-drian 3 မှာ ကျွန်ုပ်တို့ applet ရဲ့ center မှာ filled rectangle တစ်ခုကို ဆွဲပါမည်။

```
// Draw a rectangle

import java.applet.*;
import java.awt*;

public class Mondrian3 extends Applet {

int AppletHeight;
int AppletWidth;
int RectHeight;
int RectWidth
int RectTop;
int RectLeft;

pulic void init( ) {

Dimension d = size( );
AppletHeight = d.height;
Appletwidth = d.width;
RectHeight = AppletHeight/3;

RectWidth = AppletWidth/3;
int RectTop = (AppletHeight- RectHeight)/2;
```

```
int RecLeft = (AppleWidth-recWidth)/2;
repaint( );

}

public void paint (Graphic g) {
g.drawRect (0, 0, AppleWidth-1, AppletHeight-1);
g.fillRect(RectTp, RecLft, Recwidth-1,
   RectHeight-1);
}
}
```

နောက်ဆုံး example ကလည်း demonstrate ပြလုပ်တာပါပဲ။ ကျွန်ုပ်တို့ဟာ drawRect နှင့် fillRect methods များမှ points နှစ်ခုကို ဖြတ်ပြီး ၎င်းတို့ကို ဆက်သော rectangle ကို ဆွဲပြီးပြီ။ rectangles များဟာ ဘယ်လိုဖော်ပြသလဲ၊ ဘယ်လိုပဲ ဖြစ်ဖြစ် (100,100) နှင့် (100,100) ကြားတွင် ဆွဲသော ရှေ့မှ rectangle case များတွင် small rectangle ပဲ ဖြစ်မည်။ Last two variables သည် width နှင့် height ကို ဖော်ပြတဲ့အတွက် မှန်ရမည်။ drawRect နှင့် fillRect ကို ဆောင်ရွက်သော last two variables များသည် height နှင့် width ဖြစ်သော်လည်း မည်သည့်အရာဟာ ဘာဖြစ်မလဲ ဆိုတာ ဘယ်လိုသိမလဲ။ အရှင်းဆုံးနည်းမှာ non square rectangle တစ်ခုကို ဆွဲသော test program တစ်ခုကို ရေးရမှာ ဖြစ်သည်။ ကြိုးစားကြည့်ကြစို့။

```
//Draw a rectangle

import java.applet.Applet;
import java.awt.*;

public class Mondrian4 extends Applet {
int AppletHeight; AppletWidth; RectHeight, RectWidth;
RectTop; RectLeft;


public void init( ){

Dimension d = size ( );
AppletHeight = d.height;
AppletWidth = d.width;
RectHeight = AppletHeight/3;
RectWidth = (AppletWidth*2)/3;
RectTop = (AppletHeight - RectHeight)/2;
RecLeft = (AppleWidth - RectWidth)/2;

repaint( );

}
```



```
public void paint (Graphics g) {

g.drawRect (0, 0, AppletWidth-1, AppletHeight-1);
g.fillRect(RectTop, RectLeft, RectWidth-1;
   RectHeight-1);
}
}
```

အဲဒီမှာ third argument သည် width ဖြစ်ပြီး fouth သည် height ဖြစ်သည်။ ကျွန်ုပ်တို့ rectangles များကို filled နှင့် unfilled နှစ်ခုလုံးနှင့် မည်ကဲ့သို့ရေးဆွဲမလဲ လေ့လာပြီးပြီ။ ယခု rectangle ၏ position နှင့် size ကို randomly select ပြုလုပ်ကြည့်ရအောင်။ ၎င်းအတွက် java.lang.Math မှ Math. random ( ) method လိုပါသည်။

၎င်း method သည် 0.0 နှင့် 1.0 ကြားမှ double တစ်ခု ပြန်ရရှိစေတယ်။ ဒါ့ကြောင့် applet space နှင့် fit ဖြစ်မည့် reasonably sized rectangle တစ်ခုရရှိရန် applet ရဲ့ height နှင့် width ဖြင့် result ကို multiply ပြုလုပ်ရပါမည်။ အောက်ပါအတိုင်း ပြုလုပ်လျှင်

```
Randmize method;
private int Randomize(int range)
{
double rawResult;

rawResult = Math.random( );
return (int) (rawResult * range);

}
```

၎င်း method သည် Math.random ၏ result ကို လိုအပ်သည့် range မှာ int တစ်ခုအဖြစ် ရောက်ရှိစေသည်။ Last line ကို special attention ဖြစ်စေသည်။ int သို့မဟုတ် float ကဲ့သို့ parentheses မှ raw type တစ်ခုမြင်ရသည့်အခါ ၎င်းသည် cast တစ်ခု ဖြစ်သည်။ Casts များသည် value type တစ်ခုမှ အခြားတစ်ခုသို့ ပြောင်းသည်။ Double တစ်ခုမှ int တစ်ခုသို့ ပြောင်းမည်။ Java မှ casting သည် C သို့မဟုတ် အခြား languages များမှာထက် ပိုမို save ဖြစ်သည်။ Java သည် float တစ်ခုနှင့် int တစ်ခုကြားမှ cast တစ်ခုကဲ့သို့ sense ကို ပြုလုပ်သည့်အခါ casts များကိုသာ ဖြစ်ပေါ်စေသည်။ int တစ်ခုနှင့် String တစ်ခုကြားမှာ cast မပြုလုပ်နိုင်ပါ။ ဥပမာ

```
//Draw a rectangle

import java.applet.Applet;
import java.awt.*;
```

```
public class Mondrian5 extends Applet {

int RectHeight, RectWidth, RectTop, RectLeft,
   AppletWidth, AppletHeight;

public void init( ) {

Dimension d = size( );
AppletHeight = d.height;
Appletwidth = d.width;
RectTop = Randomize (AppletHeight);;
RectLeft = Randomize (AppletWidth;
RectHeight = Randomize (AppletHeigt- RectTop);
RecWidth = Randomize (AppleWidth - RectLeft);

repaint( );

}

public void paint (Graphics g) {

g.drawRect (0, 0, AppletWidth-1, AppletHeight-1,
g.fillRect(RectTop, RectLeft, RectWidth-1,
   RectHeight-1);

}

private int Ramdomize (int range)
{
double rawResult;

rawResult = Math.random( );
return 9int) 9rawResulr(range);

}

}
```

ဤ applet သည် rectangle တစ်ခုကို randomly produce ပြုလုပ်သည်။ အလွန်သေးငယ်တဲ့ မမြင်ရသော rectangle လည်း ဖြစ်နိုင်သည်။ အချိန် အနည်းငယ်စီတွင် reload ပြုလုပ်သည်။ အဲဒီအချိန်တစ်ခုစီမှာ different size နှင့် rectangle တစ်ခုဟာ different place မှာ မြင်ရပါမည်။

Adding Colours

ကျွန်ုပ်တို့၏ ကမ္ဘာကို ပိုပြီး ကြည့်ကောင်းအောင် အရောင်ခြယ် ကြည့်ကြပါမည်။ rectangle colour ကို အနီရောင် ပြောင်းပါမည်။ Graphic class ၏ part တစ်ခု ဖြစ်သည့် set Colour ( ) method အသစ်ကို သုံးမယ်။


```
//draw a rectangle

import java.applet.Applet;
import java.awt.*;

public class Mondrian6 extends Applet {

int RectHeight, RectWidth, RectTop, RectLeft,
   AppletWidth, AppletHeight;

public void init( ) {

Dimension d = size( );
AppletHeight = d.height;
Appletwidth = d.width;
RectTop = Randomize(AppleHeight);;
RectLeft = Randomize (AppletWidth);
RectHeight = Randomize (AppletHeight - RectTop);
RecWidth = Randomize(AppleWidth - RectLeft);

repaint( );

}

public void paint(Graphics g) {

g.setBackground(color.white);
.setColor(color.red);
g.drawRect(0, 0, AppletWidth-1, AppletHeight-1);
g.fillRect(RectTop, RectLeft, Rectwidth-1,
   RectHeight-1);

}

private int Randomize(int range)
{
double rawResult;

rawResult = Math.random( );
return (int) (rawResult * range);

}

}
```

awt မှာပါဝင်တဲ့ colour များကတော့

– black , blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, yellow

ငင်းသည် အချို့ colours များကို browser ရဲ့ elements များကဲ့သို့ အဓိပ္ပာယ်ဖွင့်ဆိုထားပါသည်။ ငင်းမှာ–

- editable Text
- menuBack
- menuBright
- menuDim
- menuFore
- menuHighlight
- readOnlyText

သင်၏ လိုအပ်ချက်များအတိုင်း ဖြစ်ရန် မလုံလောက်လျှင် web pages များပေါ်မှာ background colours များကို ဖြစ်စေသော တူညီသည့် RGB triple ကဲ့သို့ ပုံစံများကို define ပြုလုပ်နိုင်သည်။ bg color tag အတွက်သုံးရန် hex values များအစား decimal numbers များကို သုံးသည်။ Medium gray တစ်ခုကို ရွေးရန် example အတွက် color(127, 127,127)ကို သုံးသည်။ Pure white မှာ Color(255,255,255) ဖြစ်မည်။ Pure red သည် (255,0,0)အစရှိသဖြင့် ဖြစ်ကြသည်။ Colour constructor ကို သုံးခြင်းဖြင့် rectangle အတွက် random rectangle သာမက random colour ပါ ရွေးချယ်ရန် program ရေးဆွဲနိုင်သည်။ code ဖော်ပြရလျှင်

```
//Draw a randomly colored rectangle

import java.applet.Applet;
import java.awt.*;
public class Mindrian7 extends Applet {

intRectHeight, RectWidth, RectTop, RectLeft,
   AppletWidth, AppletHeight;
color Rectcolot;

public void init( ) {

Dimenson d = size( );
AppletHeight = d.height;
Appletwidth = d.width;
RectTop = Randomize . (AppletHeight);;
RectLeft = Randomize (AppletWidth;
RectHeight = Randomize (ApplatHeight - RectTop);
RecWidth = Randomize (AppleWidth - RectLeft);
RectColor = newColor(Randomize (255), Randomize(255),
   Randomize(255));

repaint( );

}

public void paint (Graphics g) {

g.setBackground(Color, white);
```



```
g.setColor(Rectcolor);
g.drawRect(0, 0, AppletWidth-1, AppletHeight-1);
g.fillRect(rectTop, RectLeft, Rectwidth-1,
   RectHeight-1);

}

private int Randomize(int range)
{
double rawResult;

rawResult = Math.random( );
return(int) (rawResult * range);

}

}
```

နောက်ထပ် ဥပမာမှာတော့ multiple randomly sized နှင့် randomly coloured rectangles များ ရေးဆွဲကြမည်။ rectangle တစ်ခုစီဟာ တူညီမှုမရှိစေရန် rectangle ရဲ့ shape, position နှင့် paint( ) method မှ colour များရဲ့ တွက်ချက်မှုတွေ ရှိမည်။

```
// Draw may randomly colored rectangles

import java.applet.Applet;
import java.awt.*;

public class Mondrian8 extends Applet{

int RectHeight, RectWidth, RectTop, RectLeft,
   AppletWidth, AppletHeight;
color RectColor;
int numberRectangle = 100;

public void init( ) {

Dimension d = size( );
AppletHeight = d.height;
Appletwidth = d.width
repaint( );

}

public void paint(Graphics g) {

g.setColor(Color.black);
g.drawRect(0, 0, AppletWidth-1, AppletHeight-1);

for (int I=0; i
```

နောက်ဆုံး pass တစ်ခုမှာ ရေးဆွဲသော rectangles များ၏ number များကို ဖော်ပြရန် HTML ကို သုံးကြည့်ကြစို့။ ကျွန်ုပ်တို့ default value ကို ရရှိပါမည်။ ပြီးလျှင် HTML နှင့် ၎င်းကို အစားထိုးပြီး PARAM number တစ်ခု ပါဝင်တယ်။

```
//Draw many random rectangles

import java.applet.Applet;
import java.awt.*;

public class Mondrian9 extends Applet {

int RetHeight, RerWidth, RectTop, RectLeft,
   AppletWidth,AppletHeight;
colorRectColor;
int numberRectangles = 100;

public void init( ) {

Dimension d = size( );
AppletHeight = d.heght;
AppletWidth = d.width;
String s = getParameter ("Number");
If (s ! = numm) {
Numberedectangle = Interger.valueOf(s).intValue( );
}

repaint( );

}

public void paint(Graphics g) {

g.setColor(Color.black);
g.drawRect(0, 0, AppletWidth-1, AppletHeight-1);
for (int i = 0, I)

}
```

Drawing Lines

awtမှာ အမျိုးမျိုးသော graphics primitives(ပုံစံအဟောင်းများ) များ ပါဝင်သည်။ Rectangles များသည် တစ်ခုအပါအဝင်ဖြစ်ပြီး ကျွန်ုပ်တို့လေ့လာပြီးကြပြီ။ နောက်တစ်ခုမှာ line များ ဖြစ်သည်။ Graphic context တစ်ခုမှာ line drawing method key တစ်ခု ရှိသည်။ ၎င်းမှာ draw line (int x1, int y1, int x2, int y2) ဖြစ်သည်။ ဤ method ကတော့ point (x1,y1) နှင့် point (x2,y2) ကြားမှာ straight line တစ်ခု ရေးဆွဲသည်။ applet frame ကို ဖြတ်၍ ထောင့်ဖြတ်လိုင်း တစ်ခုဆွဲသော simple applet တစ်ခုကို ကြည့်ရအောင်။


```
// Draw lines

import java.applet.Applet;
import java.awt.*;
public class SimpleLine extends Applet {

int AppletHeight, AppletWidth;

public void init( ) {
Dimension d = sixw( );
AppletHeight = d.height;
AppletWidth = d.width;
}

public void paint 9Graphics g) {

g.drawline (0, 0, AppletWidth, Appleteight);

}
```

Graphing Functions

ယခုအခါ ကျွန်ုပ်တို့ non-straight figures အရေအတွက် များစွာကို ဆွဲသားရန် drawLine ( ) method ကို သုံး၍ demonstrate ပြလုပ်ကြပါမည်။ ၎င်းသည် အရေအတွက် အတိအကျ သတ်မှတ်ထားသော straight lines များနှင့် အနီးစပ်ဆုံး ကောင်းစွာ ဖြစ်ပေါ်နိုင်သော function များ ပါဝင်သည့် advanced calculus မှ ပြထားသည်။ ကျွန်ုပ်တို့ သင့်ကို mathe-matical proof အသေးစိတ်ကို ပေးထားမည်။ သို့သော်လည်း ကျွန်ုပ်တို့ဟာ သင်ပြလုပ်မည့် graphic functions job ကောင်းတစ်ခု ဖြစ်စေသော applet တစ်ခု ပြလုပ်ခြင်းဖြင့် သင့်ကို ၎င်း၏ probability ကို ပြသပေးမည်။ တစ်ကြိမ်တည်း ပြလုပ်ရုံမျှ မဟုတ်ဘဲ အကြိမ်များစွာအတွက် ပြလုပ်ကြမည်။ skeleton applet နှင့် စပါမည်။ တစ်စုံတစ်ခု ရေးဆွဲလျှင် applet paint method code အချို့ လိုသည်။ Image ရဲ့ left hand side မှ right hand side သို့သွားသော sine wave တစ်ခုကို စဆွဲကြစို့။ Program အပြည့်အစုံကတော့

```
Import java.applet. *;
Import java.awt.*;

public class GraphApplet extend {

int x0, xN, y0, yN;

pulic void init( ) {
// How big is the Applet?
Dimension d = size( );

X0 = 0;
XN = d.width-1;
```

```
YO = 0
YN = d.height-1;
}

public void paint (Graphics g){

for (int x = x0); x

The meat of this applet is in the for loop of the paint method.

for (int x - x0, x < xN, x++){
g.drawLine(x,(int) (YN*Math. sin(x)), x+1, (int)
   (yN*Math.sin(x+1)));
}
```

ၚင်းတွင် applet ၏ x pixel တိုင်းကို ဖြတ်၍ loop ပတ်သည်။ တစ်ခုစီမှာ pixel ရဲ့ sine ကို တွက်သည်။ next pixel ရဲ့ sine ကိုလည်း တွက်သည်။ ကျွန်ုပ်တို့ကို 2-0 points နှစ်ခုပေးပြီး ၚင်းတို့ကြားမှာ line တစ်ခုဆွဲသည်။ ကိန်းစစ်ရဲ့ sine သည် အမြဲတမ်း one နှင့် negative one ကြားတွင် ရှိသည်အတွက် ကျွန်ုပ်တို့သည် y value ကို y N ဖြင့် စကေးထားမည်။ နောက်ဆုံးအနေဖြင့် sines များသည် အခြေခံအားဖြင့် floating point values များ ဖြစ်တဲ့အတွက် y values ကို ints များအဖြစ် သတ်မှတ်မည်။ သို့သော် drawline သည် ints များ လိုအပ်သည်။

ဤ applet သည် run သော်လည်း problems များ တွေ့ရှိရသည်။ ၚင်းတို့သည် အောက်ပါ factors နှစ်ခုနှင့် ဆက်စပ်နိုင်သည်။

(၁) Sine များသည် floating point operatings များ ဖြစ်ကြသည်။ အမှန်တကယ် အသုံးဝင်တဲ့ graphing applet ကို ပြုလုပ်ရန် floating point numbers များ သုံးနိုင်ရမည်။

(၂) Applet ရဲ့ coordinate system သည် upper left hand Corner (0,0) မှ right and down ကို ရေတွက်သည်။ standard cartesian coordinate system မှာတော့ lower left hand corner (0,0) မှ right and up ကို သွားသည်။ Systems နှစ်ခုလုံး၏ origin ကတော့ ရွေ့နိုင်သည်။ သို့သော် y down နှင့် y up coordinate ကြားမှ ပြောင်းလဲဖို့ လိုသည်။

ဤအချက်ကို ဖြေရှင်းနိုင်မည့် နည်းလမ်းများ ရှိသည်။ အားလုံးထဲမှ key ကတော့ မည်သို့ပဲ ဖြစ်ဖြစ် display မှ data ကို ခွဲခြားပေးရမည်။ ကျွန်ုပ်တို့သည် ကောင်းမွန်သည့် mathematical functions များကို ပိုမို၍ သို့မဟုတ် လျှော့၍ ရေးဆွဲနေသည့်အခါ ကျွန်ုပ်တို့ functions တစ်ခု ဖန်တီးလိုသော Cartesian space မှာ rectangle တစ်ခုဖြင့် ကျွန်ုပ်တို့၏ data ပြည့်စုံစွာ ဖော်ပြပေးသည်ဟု လက်ခံနိုင်သည်။ အခြားပုံစံမှ display ကတော့ size နဲ့ width points များ ပုံသေ သတ်မှတ်ထားတဲ့ rectangle တစ်ခုအဖြစ် ဖော်ပြသည်။ General Cartesian plane မှာ တွက်ချက်ရန် လိုအပ်ပြီး

particular applet window မှ display ပြုလုပ်ရမည်။

Applet window မှ point တစ်ခုကို Cartesian plane မှ point တစ်ခုသို့ ပြောင်းလဲသော method တစ်ခု ရှိရမည်။ ပြီးလျှင် မူလသို့ ပြန်ပြောင်းရမည်။ ၚင်းကို ဖော်ပြရလျှင်

```
import java.applet.*;
import java.awt.*;

public class GraphApplet extends Applet {

int x0, xN, y0, yN;
double xmin, xmax, ymin, ymax;
int AppletHeight, Appletwidth;

public void init( ) {
// How big is the Applet?
Dimension d = size( );
AppletHeight = d.height;
AppletWidth = d.width;
x0 = 0;
xN = AppletWidth - 1;
y0 = 0;
yN = AppletHeight-1;
xmin = -10.0;
xman = 10.0;
}

public void paint(Graphics g) {

double x1,y1,x2,y2;
int i, j1, j2;

j1 = yvalue(0);
for (i = 0; i
```

ဤ applet ကို run ကြည့်ပါ။ မြင်ရသည့် sine wave ဟာ အဆင်ပြေရဲ့လား။ ပိုပြီး ပြည့်စုံတဲ့ applet တစ်ခု ပြုလုပ်ရန် ပေါင်းပေးရမယ့်အရာများ ရှိနေပါသေးသည်။ အရေးကြီးဆုံးမှာ HTML မှ applet ၏ size ကို ဖော်ပြပေးနိုင်မည့် parameters အချို့ကို ပေါင်းထည့်ပေးရန် ဖြစ်သည်။ အောက်ပါ init ၏ modification နှင့် paint methods များသည် xmin, xmax, ymin နှင့် ymax တို့ကို parameters များမှတစ်ဆင့် ဖော်ပြရန် ရှာဖွေပေးသည်။

မည်သို့ပင်ဖြစ်စေ HTML ရေးဆွဲသူသည် ၚင်းတို့ကို ဖော်ပြရန် မေ့နေလျှင် ကျွန်ုပ်တို့က သင့်လျော်သော default values အချို့ကို ပေးပါသည်။

```
import java.applet.*;
import java.awt.*;

public class GraphApplet extends Applet {

int x0, xN, y0, yN;
double xmin, xmax, ymin, ymax;
int AppletHeight, AppletWidth;

public void init( ) {
String ParamString;
//How big is the applet?
Dimension d = size( );
Applet.Height = d.height;
AppletWidth = d.width;
x0 = 0;
xN = AppletWidth-1;
y0 = 0;
yN = AppletHeight -1;
ParamString = getParameter ("xmin");
if (ParamString ! = null) {
xmin = Double.valueOf (ParamString).doubleValue( );
}
else {
xmin = -1.0;
ParamString = getParameter ("xmax");
if (ParamString ! = null) {
xmax = Double, valueOf (ParamString).doubleValue( );
}
else {
xrax = 1,0;
}
ParamString.get Parameter (ymax");

if (ParamString ! = null) {
ymax = Double,valueOf (ParamString), douleValue( );
}
else {
ymax = 1.0;
}
ParmString = getParameter ("ymin");
if (ParamString ! = null) {
ymin = Double.valueOf (ParamString).doubleValue( );
}
else {
ymin = -1.0;
}
}

public void paint (Graphics g) {

double x3,y1,x2,y2;
```



```
int I, j1, j2;

j1 = yvalue(0);
for (i = 0; i
```

ကျွန်ုပ်တို့၏ code မှာ မဖော်ပြဘဲနှင့် graph ပေါ်မှာပဲ range တွေကို ညှိနိုင်သည်။ sine functions များနှင့်သာ graph ဆွဲပြီးပြီ။ cosines သို့မဟုတ် အခြား functions အမျိုး အစားများနှင့် graph ပြုလုပ်သည့် code များကို ကြိုးစားကြည့်ကြပါ။

An Infinite set that works with zero length

ယခုတစ်ခါ ကျွန်ုပ်တို့ fractal geometry နမူနာclassic တစ်ခုကို ဖော်ပြရန် Java ကို သုံးကြည့်ကြမည်။ ကျွန်ုပ်တို့ zero length ကို ဖော်ဆောင်သည့် ကန့်သတ်ချက် မရှိသော points အရေအတွက်နှင့် one dimensional set တစ်ခုကို ယူပါမည်။ middle third ဟုခေါ်သော set မှာ အစအဆုံး နှစ်ခုလုံးပါဝင်တဲ့ zero နဲ့ one ကြားမှ real numbers အားလုံးနဲ့ အစပြု ဖော်ပြထားသည်။ ပြီးမှ middle third ကို one third နှင့် two thirds တသီးတခြားစီဖြစ်အောင် ခွဲထုတ်မည်။ နောက် တစ်ခါ ကျန်ရစ်သော tw-line segments ရဲ့ middle third ကို one-minth နှင့် two-minths ကြားနှင့် seven minths နှင့် eight-minths ကြား ဖြတ်ထုတ်ပစ်ပါမည်။ ဤ process ကို အကန့်အသတ်မရှိ ဆက်သွားစေသည်။

သင် ရှုပ်ထွေးမှုကို တွေ့ရလိမ့်မည်။ picture တစ်ခုသည် words တစ်ထောင် တန်ဖိုးဖြစ်ပြီး java program ကောင်း တစ်ခုသည် pictures တစ်ထောင်တန်ဖိုး ရှိပါသည်။ ယခု middle-third set ကို ရှင်းလင်းရန် ဆက်တိုက် pictures များကို ဆွဲသော Java program တစ်ခုကို ပြမည်။

```
import java.applet.Applet;
import java.awt.*;
import java.util.Vector;

public class MiddleThird ectends Applet {

int AppletWidth;
int AppletHeight;

Vector endpoints = new Vector( );

public void init( ) {
Dimension d = size( );
AppletHeight = d.height;
AppletWidth = d.width;
endpoints.addElement(new Float(0.0f));
endpoints.addElement(new Float(1.0f));
```

```
public void paint(Graphics g) {

float x1, x2,
Float temFloat,
for (int i = 0; i < AppletHeight; I  += 5) {
// draw the lines
for (int j = 0, j
```

ဤ applet ကို compile ပြုလုပ်ပြီး (load) ဆွဲတင်ကြည့်ပါ။ ရှင်းရှင်းလင်းလင်း ရှိပါသလား။ middle-third set ၏ perfect representation တော့ မဖြစ်နိုင်ပါ။ သင်၏ applet window ရဲ့ အကျယ်အဝန်းပေါ်မှာ မူတည်ပြီး pixels အနည်းငယ်နှင့် စအလုပ်မလုပ်ခင် iteration ၆ ခုမှ ၁၂ ခုသာ မြင်ရဖို့ ရှိသည်။

Flying Lines

နောက် example တစ်ခုမှာ code ရေးခြင်းထက် ရှင်းပြရတာ ပိုခက်ပါသည်။ ရှေ့တွင် ဖော်ပြထားသည့်အတိုင်း၊ ၎င်းသည် အကန့်အသတ်မရှိ loop ပတ်သည်။ သို့သော် random images များထက် အနည်းငယ်ပိုကောင်းသည်။ အောက်ပါ code ကို compile ပြုလုပ်ကြည့်ရအောင်။ ပြီး run ကြည့်ပါမည်။ algorithm ကို နားလည်နိုင်လျှင် code ကို ခြိုပြီး ကြည့်ရအောင်ပါ။

```
Bounce lines around in a box
import java.applet.Applet;
import java.awt.*;

public class FlyingLines extends Applet {

int NUM_LINES = 25;
int gDeltaTop = 3, gDeltaBottom = 3;
int gDeltaLeft = 2, gDeltaRight = 6;
int AppletWidth, ApplerHeight;
int gLines [ ] [ ] = new int [NUM_LINES] [4];

public void init( ) {

Dimension d = size( );
AppletWidth = d.width;
AppletHeight = d.height;
gLines[0] [0] = Randomize (AppletWidth);
gLines[0] [1] = Randomize (AppletHeight);
gLines[0] [2] = Randomize (AppletWidth);
gLines[0] [3] = Randomize (AppletHeight);
for (int I = 1, i AppletHeight)) {
gDeltaTop += -1;
gLines [i] [1] += 2*gDeltaTop;
```



```
}

gLines [i] [3] += gDeltaBottom;
if ( (gLines[i] [3] AppletHeight) )
{
gDeltaBottom += -1;
gLines [i] [3] += 2*gDeltaBottom;
}

gLines [i] [0] += gDeltaLeft;
if((gLines [i] [0] =+ AppletWidth))
{
gDeltaLeft += -1;
gLines[i] [0] += 2*gDeltaLeft;
}

gLines [i] [2] += gDeltaRight;
if((gLines[i] [2] += AppletWidth) )
{
gDeltaRight += -1;
gLines[i] [2] += 2*gDeltaRight;
}
} //RecalcLine ends here
} //FlyingLines ends here
```

Drawing Images

Graphic class မှ ရေးကွဲသော graphics အစုအဝေးသည် အမှန်တကယ် အလွန်သေးငယ်ပါသည်။ ၎င်းတို့မှာ lines များ၊ rectangle, circles, ovals, polygons များ အစရှိသဖြင့် ဖြစ်ကြသည်။ ၎င်းတွင် Bezier curves များ သို့မဟုတ် အခြား အသုံးဝင်တဲ့ tools များ မရှိပါ။ HTML မှာသာ data types များကို တုပပြုလုပ်နိုင်ပါသည်။ မည်သို့ပင်ဖြစ်စေ applet များကို မပြဿနိုင်ခင် ဆွဲတင်ရန် GIF သို့မဟုတ် JPEG file များနှင့် applet တစ်ခုကို ပြုလုပ်သော bad idea တစ်ခုအနေဖြင့် large images များနှင့် HTML page တစ်ခုကို ပြည့်စုံစေသည်။

THREADS

သင်၏ operating system ပေါ်မှာ တည်မှီခြင်းနှင့် flying line programs များအား ကြည့်ထားသော Java enabled browser တို့သည် CPU မှာ နေရာယူကြသည်။ FlyingLines မှ paint loops များသည် thread တစ်ခု အတွက် စံနမူနာ ဖြစ်သည်။ threads မရှိဘဲနဲ့ ပြည့်ဝတဲ့ program တစ်ခုဟာ CPU intensive task တစ်ခုဖြင့် ကိုင်တွယ် အသုံးပြုနိုင်ပါသည်။ သို့မဟုတ် Fly-ingLines များမှ infinite loop ရတတ်ရက် ဖြစ်ပေါ်မှု သို့မဟုတ် အခြား ပုံစံများလည်း ရှိပါသည်။ မူတစ်ခုအနေနဲ့ကတော့ CPU intensive tasks များ အားလုံးသည် ၎င်း၏ own threads မှာပဲ နေရာယူပါသည်။ ပုံစံတစ်ခု ပြုလုပ်ကြရအောင်။

```
Draw infinitely many random rectangles
import java.applet.Applet;
impor java.awt.*;

public class ThreadedMondrian extends Applet implements
   Runnable
{
int RectHeight, RectWidth, AppletHeight, RectTOp,
   RectLeft, AppletWidth, AppletHeight;
Color RectColor;
Thread kicker = null,
int pause;
public void init( ) {
Dimension d = size( ),
AppletHeight = d.height;
AppletWidth = d.width;
repaint( );
}

public void paint(Graphics g)

g.SetColor(Color.black);
g.drawRect(0, 0, AppletWidth-1, AppletHeight-1);
for (int I = 0, i
```

ကျွန်ုပ်တို့ CPU friendly ပိုမို ရှိစေရန်နှင့် threaded ပြုလုပ်ရန် key ၄ မျိုး ထပ်ထည့်ပြီး ဖြစ်သည်။

1. We specified that our applet implements Runnable.
2. We added a run method.
3. We added a start method.
4. We added a stop method.

အခန်း (၆)

Object Orients Programming

Object Oriented Programming အကြောင်းကို မသိပေမည် ကြားဖူးနားဝ ရှိကြပါလိမ့်မည်၊ OOPs ဟာ အရိုးရှင်း ဆုံး programming ပုံစံဖြစ်ပါသည်။ Computer အတွက် programming design များက တူညီကြပါသည်။ OOP's languages အားလုံးသည် သင်တို့ရဲ့ natural ideas ကို ပိုပြီး ကောင်းမွန်လာအောင် ဖော်ပြပေးနိုင်စွမ်း ရှိပါသည်။ Various objects များအကြောင်းကိုလည်း ရှင်းပြပေးပါသည်။ Objects များကို အမျိုးမျိုးသော classes များအဖြစ်သို့ ၎င်းတို့၏ properties များနှင့် behaviours များအတိုင်း ခွဲခြားပေးပါသည်။ သင်၏ objects look က ဘယ်လို၊ objects behave က ဘယ်လိုဆိုတာတွေအပြင် new objects များကိုလည်း ရှင်းပြပါသည်။ old objects များကို ပြင်ဆင်ပြီး ပိုကောင်းအောင် လုပ်ပေးသည်။ သင်၏ objects များကို အခြားတစ်ခုဆီသို့ messages များပို့ပေးပြီး systematic manner တစ်ခုအဖြစ် အလုပ်လုပ်စေသည်။

၎င်းတို့သည် computers များ၏ capacity နှင့် limitations များကို ဂရုမစိုက်ဘဲ အဖြစ်နိုင်ဆုံး နည်းလမ်းမှာပဲ natural behaviours များနဲ့ attributes များကို တည်ဆောက်သည့် programs များကို ရေးနိုင်သည်။ Java သည် ၎င်း facilities များ ပေးနိုင်သည့် OOP language တစ်ခု ဖြစ်သည်။ Java သည် နောက်ဆုံးပေါ် OOP language ဖြစ်ပြီး အသုံးပြုရန်အသင့်ဖြစ်သော OOP languages အားလုံး၏ကြားတွင် အဆင့်မြင့်ဆုံးတစ်ခု ဖြစ်ပါတယ်။

Classes And Objects

Object ဆိုသည်မှာ သင့်အတွက် အလုပ်အချို့ကို လုပ်ပေးနိုင်သော အရာတစ်ခုပဲ ဖြစ်ပါသည်။ လူတစ်ယောက်သည် object တစ်ခုဖြစ်သည်။ Animals များလည်းပဲ objects ဖြစ်သည်။ ဤ objects များသည် သင့်အလုပ်ကို ဘယ်လိုပြုလုပ်မလဲ၊ သင့်တောင်းဆိုချက်အတိုင်း ဖြစ်မှာလား။ သင့်မှာ health problem အချို့ရှိလျှင် သင်သည် ဆရာဝန်ဆီကို သွားဖို့လိုသည်။ သင်သည် ဆရာဝန်များ အားလုံးထဲမှ special attributes နှင့် behaviours ရှိတဲ့သူကို ရွေးချယ်စဉ်းစားနိုင်သည်။ အဲဒီ attributes နှင့် behaviours စုတစ်ခုကို class တစ်ခုလို့ ခေါ်သည်။ Class တစ်ခု၏ member များကို

class ၏ objects များလို့ ခေါ်သည်။ ဆရာဝန်များ၏ class တစ်ခု၊ ရှေ့နေများ၏ class တစ်ခု၊ ဆရာများ၏ class တစ်ခု၊ ဘဏ်လုပ်ငန်းရှင်များ၏ class တစ်ခုဟူ၍ ရှိကြသည်။ အားလုံးကတော့ attributes ပုံစံအမျိုးမျိုးနဲ့ behaviours ပုံစံ အမျိုးမျိုးရှိသည်။ လူဘဝ ရှင်သန်မှုသည်လည်း class တစ်ခုဟု ဆိုနိုင်သည်။ အခြား class များသည် ၎င်းကနေ ဆင်းသက်ကြတာချည်း ဖြစ်သည်။ Mr X ဆိုတဲ့ ဆရာဝန်တစ်ယောက်ဟာ doctors class တစ်ခု၏ နမူနာတစ်ခုဖြစ်ပြီး အခြားဆရာဝန်ဖြစ်သည့် Mr Y သည်လည်း doctors class ၏ အခြားနမူနာတစ်ခု၊ တတိယ ဆရာဝန်ဖြစ်သည့် Mr Z သည် doctors class ရဲ့ တတိယနမူနာတစ်ခု စသည်ဖြင့် ဖြစ်ကြပါတယ်။

Mr X, Y, Z အားလုံးသည် doctors များ၏ characteristics များအဖြစ် ပြပြီး different objects များ ဖြစ်ကြသည်။ ၎င်းသည် objects တစ်ခု၏ analog၊ ၎င်း၏ variable တစ်ခုနှင့်အတူ class နှင့် ၎င်း၏ type ကို စိတ်ဝင်စားသည်။

မကြာသေးမီကပင် သင် numeric types များအကြောင်း (byte, short, int, long, float double, clar နှင့် boolean အခြားအမျိုးအစားများ) ကို လေ့လာပြီးကြပြီ။ တစ်ခုစီကတော့ အမှန်တကယ် type တစ်ခုထက် class တစ်ခုအဖြစ် ပိုပြီးဖော်ပြကြသည်။ int ၏ attributes သည် long နှင့် မတူကြပါ။ သူတို့၏ behaviours များ တူညီသော်လည်း ranges က မတူပါ။ အခြားဘက်မှာတော့ class int နှင့် class double တို့မှာ different behaviours များကဲ့သို့ different attributes များ ရှိကြသည်။ int case မှာ doubles case အဖြစ် မဖော်ပြဘဲနှင့် % operation ကို သုံး၍ အကြွင်း တွက်ချက်သည့်နည်း ရှိသည်။ သူတို့အားလုံးဟာ mathematical numbers ကိုယ်စားပြုထားကြသော်လည်း မတူညီသည့် classes များ ဖြစ်ကြတယ်။ တစ်ခုစီသည် ၎င်းတို့၏ own attributes နှင့် behaviours များနှင့် class တစ်ခုအဖြစ် ကြည့်နိုင်သည်။ အလားတူပဲ char နှင့် boolean တို့သည်လည်း classes များအဖြစ် ကြည့်နိုင်ပါသည်။

Teachers class တစ်ခုမှ ကျောင်းအုပ်ကြီးများ၏ subclass တစ်ခု ရှိလာသည်။ ပေးထားသည့် class တစ်ခုမှ ဖြစ်လာမည့် subclass အစိတ်အပိုင်းကတော့ ကန့်သတ်၍ မရပါ။ Class တစ်ခုမှ objects အားလုံးတွင် မှန်ကန်သော attributes နှင့် behaviours များ ရှိကြပါတယ်။ Subclass တစ်ခုမှာ original class ၏ attributes များနှင့် behaviours များ ပါလာပါသည်။ ထို့အပြင် original class မှာ မရှိသည့် ပိုမိုကောင်းသည့် attributes များနှင့် behaviours အချို့ကိုလည်း တွေ့ရပါသည်။ မတူညီသည့် additions များသည် မတူညီသည့် subclass များကိုသာ အဓိပ္ပာယ်ပေါ်လွင်စေပါသည်။ Subclass တစ်ခုကို ရှင်းပြပြီးနောက် original class သည် ၎င်း၏ superclass ဖြစ်သွားသည်။ ဥပမာ သင်ဟာ numbers အားလုံး၏ general class တစ်ခုကို စဉ်းစားနိုင်သည်။ ဤတွင် arithmetic operations ၄ ခုဖြစ်သည့် +, −, * နှင့် / တို့ဟာ behaviours methods များ ဖြစ်ကြသည်။ ထို့နောက် သင်ဟာ fixed point numbers နှင့် floating point numbers များအား numbers ရဲ့ subclasses နှစ်ခုအဖြစ် ကြည့်နိုင်သည်။

Fixed point subclass မှာ number အားလုံးအတွက် အသုံးမပြုနိုင်သော ၎င်း objects များကို ဖော်ပြသည့် % operation တစ်ခု တိုးလာတယ်။ Floating point class objects များမှာ numbers အားလုံးတွင် မသုံးနိုင်သည့် E အသုံးပြုသော scientific notation ရှိသည်။ Floating point numbers ရဲ့ subclasses အဖြစ် float နှင့် double, fixed point num-bers ရဲ့ subclasses အဖြစ် byte, short, int နှင့် long တို့ ဖြစ်လာကြသည်။ subclass တစ်ခုစီတွင် ၎င်း၏ superclass ရဲ့ properties များအပြင် အနည်းဆုံး different range အဖြစ် အရေးပါသည့် special properties များ ရှိကြသည်။

Class date အတွက် Java declarations များကတော့

```
class date
{
int day;
int month;
int year;
}
```

သို့မဟုတ် စုပေါင်းပြီး ကြေညာလို့ ရပါသည်။

```
class date
{
int day, month, year;
}
```

ဤသည်မှာ attributes များအကြောင်း ကြေညာရုံပဲ ရှိသေးသည်၊ သို့မဟုတ် date class ရဲ့ data part ပဲ ရှိပါသေးသည်။ Behavioural description ရဲ့ အခြား part တစ်ခုဖြစ်တဲ့ methods လည်းရှိပါသေးသည်။ date ပေါ်မှာ မည်သည့် operations တွေ ပြုလုပ်ပါမည်လဲ။ To set a date, to print a date, to get a date, to get the next date, to get the previous date အစရှိသဖြင့် ဆောင်ရွက်မလား။

Java မှာ ဤ operations တစ်ခု ပြုလုပ်ရန် method တစ်ခုရှိပါသည်။ methods နှင့် date class ရဲ့ definitions ကို ကြည့်လျှင်

```
class date
   {
   //_____Data(attrubutes).
   int day, month, year;
   // _____Methods (behaviour)
   void set-date(int dd, int mm, int yy)
      { }; // Empty body for the time being.
   void print_date( )
      { }; // Empty body for the time being.
   II only two methods for the time being.
   }
```

ပေးထားသော date ကို နေရာချရန် day (dd), month (mm), year (yy) တို့၏ value ကို ပေးရပါမည်၊ set-date () ပြီးနောက် brackets အတွင်းမှ ဤ variables ၃ ခုကို ကြေညာသည်။ print အတွက် မည်သည်ကိုမှ ကြေညာဖို့မလိုပါ။ set values ကို သုံးသည်။ နှစ်ခုလုံးမှ class ၏ members များအဖြစ် ကြေညာထားသော data နဲ့ methods များဖြစ်သည်။

Creating Objects

သင် class date ကို တစ်ကြိမ် define ပြုလုပ်ပြီးလျှင် ထို class ၏ objects များကို ကြေညာနိုင်သည်။ ဥပမာ class student တစ်ခု (ကျောင်းသားများဟာ objects များဖြစ်သည်။ သူတို့ကို class တစ်ခု တည်ဆောက်ထားတယ်။) ကို ဖွင့်ဆိုသည့်အခါ သင် date ကို သုံးရပါမည်။

```
class student
   {
   // _________Data
      String name;
      date_birth date, joining_date;
   // __________ Methods
   // _________No method for the time being.
   }
```

ဤ class မှာ birth-date နှင့် joining-date ဆိုသည့် date objects နှစ်ခု ရှိသည်။ birth-date object သို့ message တစ်ခုပို့ခြင်းဖြင့် class strudent တစ်ခုမှာ birth-date ကို နေရာပေးရမယ်။ အောက်ပါ statements တစ်ခု ရေးနိုင်သည်။

```
birth-date.set-date (20,8,1986);
```

birth-date နှင့် joining-date objects များကို class နှစ်ခုအဖြစ် ဆက်စပ်လို့ရသည်။ တစ်ခုက သူတို့၏ type ကို ဖော်ဆောင်သည့် date class ဖြစ်သည်။ ဤနှစ်ခုသည် date class ရဲ့ နမူနာများ ဖြစ်သည်။ နောက်တစ်ခုမှာ data အဖြစ်သရုပ်ဆောင်သည့် student class ဖြစ်သည်။ ဤ objects များသည် student class ရဲ့ data လို့လည်း ပြောလို့ရသည်။ ဤအခြေအနေမျိုးတွေဟာ အချိန်တိုင်း ဖြစ်နေတတ်သည်။ Object တစ်ခုသည် အချို့ class ၏ နမူနာတစ်ခုဖြစ်ပြီး အခြား class မှ data အဖြစ်လည်း သုံးသည်။

Java မှာ သင်ရေးလိုက်သည့် တစ်စုံတစ်ခုသည် data လည်း ဖြစ်နိုင်သလို methods များလည်း ဖြစ်နိုင်သည်။ အချို့ class အတွင်းမှာ အမြဲ ရှိသည်။ Pascal, FORTRAN, C, C++ ကဲ့သို့ language တွေမှာ ကန့်သတ်ချက်မရှိ။ အကြောင်းမှာ ၎င်းတို့သည် OOP languages များ မဟုတ်သည့်အတွက် ဖြစ်သည်။ ဤအခြေအနေမျိုးကို ရှေ့မှ programs များတွင် test ဟုခေါ်သော dummy class အချို့ကို ကြေညာခဲ့တာ

မြင်ပြီးပါပြီ။

THE NEW OPERATOR

သင့် Java မှ class ရဲ့ object တစ်ခုကို ကြေညာသည့်အခါ [illegible] ကြေညာမည်။

```
date birth-date;
```

သို့သော် object သည် computer ရဲ့ memory မှာ မမှတ်သေးပါ။ new operator ကိုသုံးပြီး တိတိကျကျ တည်ဆောက်ရပါမည်။

```
birth-date = new date ( );
```

ပြီးမှ လိုအပ်သည့် memory မှာ နေရာယူပြီး ဤ birth-date object တည်ဆောက်သည်။ Date စကားလုံးပြီးနောက် ( ) ရှိမည်။ ထိုအချက်သည် အရေးကြီးပါသည်။ new operator ကို သုံးရန် မမေ့ပါနဲ့။ ( ) brackets ကိုလည်း မမေ့ပါနဲ့။

သင် object တစ်ခု ပြုလုပ်ပြီးနောက် ၎င်းကို သုံးနိုင်သည်။ သင့် statements အစဉ်ကို ဖော်ပြလျှင်

```
date birth_date;
//..... .other code
birth_date = new date ( );
//.....other code
birth_date.set-date(26, 8, 1986);
```

Java တွင် declaration နဲ့ creation ကို single statement တစ်ခုအဖြစ် ပေါင်းစပ်၍ ရပါသည်။

```
date birth_date = new date ( );
```

Java terminology မှ creation သည် object တစ်ခု၏ initialization လည်း ဖြစ်ပါသည်။

အထူးအခွင့်အရေး တစ်ခုမှာ (declare, create နှင့် set) action သုံးခုကို statement တစ်ခုတည်း ပေါင်းနိုင်ခြင်းပင် ဖြစ်သည်။

```
Int x = 100; // new is not required!
```

Operations On Objects

ယခု Java မှ methods များအား defining နှင့် using ကို ရှုထောင့်အမျိုးမျိုးမှ ဆွေးနွေးပါမည်။ test 1.java ကို ကြည့်ရအောင်။

```
class date
{
//_____________Data (Attributes)
int day, month, year;
// _____________ Methods (behaviour)
```

```
void print-data( )
   {
   system.out.println
   (day + "-"+month + "-" + year);
   }
void set_date(int dd, int mm, int yy)
   {
   day = dd, month = mm; year = yy;
   };
// only two methods for the time being.
}
```

ဂင်းသည် class date အတွက် ပြည့်စုံသည့် de finition တစ်ခု ဖြစ်သည်။ date class မှာ variables ၃ ခုဖြစ်သည့် day, month, year ဟုခေါ်သော int type (data items) ရှိသည်။ date တစ်ခုအတွက် setting ပြုလုပ်ခြင်းသည် ဤ data အတွက် values သတ်မှတ်ခြင်း ဖြစ်သည်။ ဤ setting ကို ပြုလုပ်ရန် set-date method တစ်ခုရှိသည်။ နောက် method ကတော့ ဤ variable ၏ values များကို point ပြုလုပ်ရန် ဖြစ်သည်။

method ရဲ့ header သည် { မတိုင်ခင်ကအဝိုင်း ဖြစ်သည်။ Method ရဲ့ body သည် { နှင့် } ကြားမှ အဝိုင်းဖြစ်သည်။ print-date method မှာ ၎င်း၏ header မှ (နှင့်) ကြားတွင် ဘာမှမရှိပါ။ ၎င်းသည် new object ကို တွဲမသုံးဘဲ header တွင် ၎င်း၏ရှေ့မှ 'void' စကားလုံး ရှိသည်။ body မှာ println statement တစ်ခုသာသုံးသည်။ set-date method သည် ရက်၊ လ၊ နှစ် variables အတွက် values သုံးမျိုးနှင့်အတူ ဖော်ပြသည်။ ၎င်းရဲ့ header မှာ int variable သုံးမျိုး ကြေညာသည်။ ၎င်း variables ကို programming languages မှာ parameters များဟုခေါ်သည်။ Dummy parameters များဟုလည်း ခေါ်သည်။ အကြောင်းမှာ ၎င်းတို့သည် method သို့ ရောက်ရှိသော dd, mm, yy အတွက် သုံးတဲ့အမည် ဖြစ်တဲ့အတွက် အရေးမကြီးပါ။ ဤ dummies တစ်ခုစီအတွက် အမှန် int number, သို့မဟုတ် xx ကဲ့သို့ int variables သို့မဟုတ် xx + yy ကဲ့သို့ int expressions များအသုံးပြုရမည်။ dummies မရှိဘဲနှင့် header ကို အောက်ပါအတိုင်း တွေ့ရသည်။

```
void set_date (int, int, int)
```

ဤအချက်ကို Java မှ method ရဲ့ signature လို့ ခေါ်သည်။ အောက်ပါ test 1 class မှ ဤ method ကို ဖြစ်စေသော message ကို သတိထားမိမှာပါ။ d0. set-date ၏ dot သည် d0 object ၏ set-date method အကြောင်းပြောခြင်း ဖြစ်တယ်။ အခြား d1 object အချို့အတွက်တူညီသည့် method ကို လိုချင်လျှင် d1.set-date ကို သုံးရမည်။

```
class test1
   {
   public static void main(String args[ ])
   {
   date d0 - new date( );
   d0 set_date(20, 8, 1986);
   // setting to 20 Aug 1986
   d0.print_date( );
   }
```

test 1 class မှာ main method ရဲ့ ပထမစာကြောင်းသည် date object တစ်ခုအဖြစ် d0 ကို ကြေညာသည်။ ပြီး 'new' ကိုသုံးပြီး create လုပသည်။ ဒုတိယစာကြောင်းမှာ date ကို 20-8-1986 ကို နေရာယူရန် ပြုလုပ်သည်။ ပြီးနောက် message တစ်ခုသည် set date ကို print လုပ်သည်။ ဤ pro-gram ကို compile လုပ်ပြီး run ကြည့်လျှင် အောက်ပါအဖြေကို ရမည်။

```
20-8-1986
```

ယခု test 1 class ကို ပြင်ရေးကြည့်မည်။

```
class test 1
{
public static void main9String args[ ])
   {
   date d0 = new date( )
   date d1 = new date( );
   d0 set_date(20, 8, 2986);
   d1 = d0;
   d0.print_date( );
   d1print_date( );
   }
}
```

၎င်းအား compile ပြုလုပ်ပြီး run ပါက အောက်ပါအတိုင်း အဖြေရပါမည်။

```
20-8-1986
20-8-1986
```

'new' နှင့် create ပြုလုပ်သော class date ရဲ့ d1 object ကို အဓိပ္ပာယ် ဖွင့်ဆိုထားသော်လည်း set-date method ကို သုံးပြီး date ကို setting မပြုလုပ်ဘဲ d0 date မှ ကူးယူသည်၊ သို့မဟုတ် ရောက်ရှိစေသည်။ print လုပ်တဲ့အခါ အဖြေနှစ်ခုလုံး တူညီသည်။

PARAMETERS

Method ဆီသို့ information ရောက်ရှိဖို့အတွက် standard way ကတော့ ၎င်း၏ heading မှာ parameters (dummies) များကို ကြေညာရန်ပဲ ဖြစ်သည်။ ကြေညာပြီးမှ method ဆီ message သွားနေစဉ် ဆက်စပ်တဲ့ နေရာများမှာ values ပေးရမယ်။ date class ရဲ့ set-date method မှာ int type information အားလုံးဖြစ်သော dd, mm

နှင့် yy parameter သုံးခု ရှိသည်။ ၎င်းတို့ဟာ set-date method ကို ဆောင်ရွက်ရန် ကိုယ်စားပြုသော class variables များဖြစ်သည့် date, month နှင့် year တို့ assign ပြုလုပ်ပါမည်။

သင်ကြေညာလျှင်

```
void set-date (int dd, int mm, int yy)
```

message တစ်ခု ပေးမည်ဆိုပါက

```
d0.set-date (20, 8, 1986);
```

မှတ်ရန်မှာ ဤ parameters များသည် ကြိုတင်ဖွင့်ဆိုထားသည့် types များ၊ byte, short, int, long, float, double, char နှင့် Boolean တို့လည်း ဖြစ်နိုင်သည်။ တိကျသည့် တန်ဖိုးတစ်ခု အမြဲရရှိရန် မလိုသော်လည်း အသုံးပြုမည့် variable ၏ value ကို ရေးနိုင်သည်။ ထို့ကြောင့်မို့ သင့် program မှာ int xx ကို ကြေညာပြီး၍ ၎င်းကို value 20 assign လုပ်ပြီးလျှင် သင့်အနေဖြင့် 20 ရဲ့ နေရာမှာ variable name xx ကို သုံးနိုင်ပြီး အောက်ပါ message ကို ရေးမည်။

```
d0.set-date (xx, 8, 1986);
```

dd, mm, yy dummies များကို assign လုပ်ရမည့် နေရာကို သတ်မှတ်ပြီးပြီဆိုလျှင် 20 ဟု ရေးခြင်းသည် xx=20 နှင့် တူသည်။ xx ဟု ရေးခြင်းသည် dd=xx နှင့်တူသည်။

သတိပြုရမည့် အချက်ကတော့ 10+10 သို့မဟုတ် 4+10+6 သည် expression အတူတူပင် ဖြစ်သည်။ types များကို ကြိုတင်ဖော်ပြခြင်းထက် parameters များသည် အခြား class အချို့ကိုလည်း ဖြစ်နိုင်သေးသည်။ ဥပမာ အချို့ method declaration မှာ

```
xyz (date dd 11, dd 22)
```

၎င်းသည် date d0 နှင့် d1 ကို ဖြတ်ကျော်ရမည်။ မှတ်ရန်မှာ general classes များ၏ ဤကဲ့သို့ objects များအတွက် value တစ်ခု သို့မဟုတ် expressions ကို assign ပြုလုပ်ရန် မဖြစ်နိုင်ပါ။ အကြောင်းမှာ value, expressions များအား ဖြစ်လာနိုင်ခြေ မရှိသောကြောင့် ဖြစ်သည်။ d0 နှင့် d1 ကဲ့သို့ class ၏ အချို့ object ၏ current value ကိုသာ ထားရှိနိုင်ပါသည်။

Returning Values

ပြီးခဲ့သည့်အခန်းမှာ method သို့ message တစ်ခု ပို့နေစဉ် method သို့ information ရောက်ရှိနိုင်သော parameters များကို ကြော်ငြာပုံကို တွေ့ပြီးပြီ။ သို့သော် ၎င်းသည် ဆက်သွယ်ရေးနည်းလမ်းတစ်ခုသာ ဖြစ်ပါသည်။ Method သည် ဤ parameters များကိုဖြတ်၍ပို့သော message sender ဆီကို information ပြန်မပေးနိုင်ပါ။ Message sender ဆီသို့ method တစ်ခုမှ information ကို ဘယ်လို ပြန်ပို့နိုင်သလဲ ကြည့်ကြစို့။

သင်သည် mathematical objects ၏ class တစ်ခုနှင့် ချိတ်ဆက်လျှင် ပေးထားသည့် နံပါတ်ရဲ့ square (third power) ကို တွက်ချက်သော method တစ်ခုကို သတ်မှတ်ပေးထားရမည်။ ပြီးမှ ၎င်းကို caller (message sender) ဆီကို ပြန်ပို့ပါမည်။ Java မှာ အောက်ပါအတိုင်း return statement ကိုသုံးပြီး ပြုလုပ်နိုင်သည်။

```
class testmath
{
//_________Data (Attributes)
// _________Methods (behaviour)
double testsqure (double nn)
   // return type double; not void.
   {
   return(nn*nn)
   }
   // Only one methods for the time being.)

class test1
{
public static void main(String args[ ])
   {
   testmeth mm = new testmath( );
   double xx;
   xx = mm.testsqure (3.0);
   System.out.println (xx);
   }
}
```

out put မှာ 27 ဖြစ်သည်။

testsquare method သည် brackets ထဲမှ expression ၏ value ကို ပြန်ရရန် return (...) statement ကို သုံးသည်။ အပေါ်မှ ဖော်ပြချက်အတိုင်း nn*nn ကဲ့သို့ တိုက်ရိုက်ဖော်ပြပြီး တစ်ခါတည်း တွက်ချက်သည့်နည်း ရှိသလို နောက်တစ်နည်းက တွက်ချက်ပြီး value ကို temporary variable မှာ သိမ်းထားသည်။ ပြီးမှ ထို value ကို ပြသည်။

```
double testsquare (double nn)
   {
   double temp;
   temp = nn*nn;
   return (temp);
   }
```

တွက်ချက်မှုများသည် ရိုးရိုး type တစ်ခုထက် class တစ်ခုရဲ့ အချို့ object ကို create ပြုလုပ်တဲ့ ရှုပ်ထွေးမှုအချို့ ရှိသည်။ ဥပမာ ကျွန်ုပ်တို့၏ class date အတွင်းမှာ ပေးထားသော date ၏ next date ကို တွက်ချက်ရန် method တစ်ခု တည်ဆောက်ပါမည်။ code ဖြင့် ဖော်ပြလျှင်

```
date next-date ( )
// return type is date; not double or void.
   {
   date temp = new date( );
   // temp stores the calculated next date.
   // Do the actual calculations here....
   return(temp);
   };
```

အောက်ပါအတိုင်းရေးပြီး ခေါ်သုံးမည်။

```
d1 = d0.next-date ( );
```

ထိုအခါ d1 တွင် d0 ၏ next date ကို သိမ်းပါသည်။

Return statement သည် method ထဲက နေရာများတွင် အကြိမ်အများကြီး ဖြစ်နိုင်သည်။ ၎င်းကို execute လုပ်သည့်အခါ control သည် ချက်ချင်းပင် message sender ဆီသို့ကို ပြန်ပို့ပေးပါသည်။ အောက်ပါ return statement code မှာမူ execute မပြုလုပ်နိုင်ပါ။ ဥပမာ

```
class test1
   {
   int max(int a, int b)
      {
      return(a);
      retun(b);
      }
   }
```

error တစ်ခုပေးပါသည်။

```
   test1.java:6: statement not reached,
      return (b);
   1 error
```

numbers နှစ်ခုမှ maximum ကို တွက်ချက်လိုလျှင် အောက်ပါအတိုင်း ရေးပါမည်။

```
class cc1
   {
```



```
   int max(int a, int b)
      {
      if (a>b)
            return(a);
      else
            return(b);
      }
   }
```

a နှင့် b ၏ value ပေါ်မူတည်၍ return statement နှစ်ခုမှ တစ်ခုကိုသာ execute လုပ်သည်။ အောက်ပါအတိုင်း ပြင်ဆင်ရေးလျှင်လည်း တူညီသော effect ရပါမည်။

```
class cc1
   {
   int max(int a, int b)
      {
      if (a>b) return(a);
      return(b);
      }
   }
```

'a' သည် 'b' ထက်ကြီးလျှင် ပထမ return (a) statement ကို ပြုလုပ်ပါမည်။ ပြီးလျှင် control သည် message sender ဆီ ချက်ချင်း ပြန်ရောက်ပြီး ကျန် statement return (b) ကို အလိုလို လျစ်လျူရှုလိုက်သည်။

သို့မဟုတ် 'a' သည် 'b' ထက် ငယ်သည် သို့မဟုတ် တူညီလျှင် if condition သည် ပထမ return (a) statement ကို ခွင့်မပြုတော့ဘဲ control သည် next statement ကိုသွားပြီး return (b) ကို ပြုလုပ်ပါမည်။

Returning More Values

JAVA မှာ method code မှ return statement ကိုဖြတ်၍ message sender ဆီသို့ information ပြန်ပို့ရန် နည်းတစ်နည်းသာ ရှိသည်၊ သို့သော် ၎င်းသည် object တစ်ခုကိုသာ အတိအကျ ဖြတ်ခွင့်ပြုသည်။ Objects တစ်ခုထက် ပိုပြီး back ပြန်ပို့လိုသည့် case မှာ ဘာလုပ်ပါမည်နည်း။ ၎င်းပြဿနာကို ဖြေရှင်းရန် ပေးထားသော point တစ်ခု၏ x နှင့် y coordinates များကို တွက်ချက်သော get-xy( ) method တစ်ခုရှိသည်။ ထို point သည် x နှင့် y coordinates နှစ်ခုလုံးဖြင့် ဖော်ပြပြီး တစ်ကြိမ်စီမှာ နှစ်ခုလုံးကို တွက်ဖို့လိုသည်။ get-xy () method မှ values များ တွက်ချက်ပြီး ပြန်လာစေသည်။

Java မှာ parameters နှစ်ခု သို့မဟုတ် return statements နှစ်ခုကို သုံးပြီး မပြုလုပ်နိုင်ပါ။ အဖြေသည် object oriented fashion တစ်ခုအား လေ့လာခြင်းဖြင့် ရရှိနိုင်သည်။ x နှင့် y (သို့မဟုတ် ၂ ခု သို့မဟုတ် ၂ ခုထက်ပို) objects များသည် လုံးလုံး

မဆက်စပ်ကြပါ။ အဖြေကတော့ ရှင်းရှင်းလေးပင် ဖြစ်သည်။ တစ်ခုစီကို တွက်ချက်ရန် separate method တစ်ခုကို အဓိပ္ပာယ်ရှင်းပါ။ ပြီးလျှင် လိုအပ်သည့်အခါမှာ သင့်လျော်သော method တစ်ခုကို သုံးပါ။ ဤနေရာတွင် get-x()နှင့် get-y()ကို ခွဲခွဲခြားခြား ဖော်ပြရပါမည်။ သို့သော် return ပြန်လာမည်။ (နှစ်ခု သို့မဟုတ် နှစ်ခုထက်ပိုသော) objects များသည် အချို့နည်းလမ်းတွေမှာ တစ်ခုကိုတစ်ခု ဆက်စပ်နေကြသည့်အခါ ထိုဆက်စပ်မှုကို ရှင်းပြနိုင်ရန် ကြိုးစားကြည့်ပါ။

အပေါ်မှ ဥပမာတွင် အချို့ point ၏ coordinates များသည် အတူတကွ တည်ရှိနေကြသည်။ 'point' ဟုခေါ်သော class တစ်ခုကို ရှင်းပြပြီးနောက် အောက်ပါအတိုင်း ရေးနိုင်သည်။

```
class point
   {
   double x, y;
   //..... .other things.
   }
```

ပြီးလျှင် အချို့ point object ၏ x နှင့် y coordinates များကို တွက်ချက်သော get-xy( ) ဟုခေါ်သော method တစ်ခုထက် get-point ( ) အဖြစ် method ကို အဓိပ္ပာယ်ဖွင့်ဆိုပြီး ဤ point object ကို ပြန်ရရှိစေရန် code ဖြင့် ဖော်ပြလျှင်

```
point get_point (....)
// This returnd a point object.
   {
   point tmp;
   //......actual code
      temp.x = ...;
      temp.y = ...;
   return(temp);
   }
```

အောက်ပါအတိုင်းလည်း ရေးနိုင်သည်။

```
point pp = new point ( );
//....other code.
pp = get_point(...);
```

ထို့ကြောင့် objects များကို တစ်ပြိုင်နက်တည်း return ပြန်လိုချင်သော အခြေအနေတစ်ခုမဖြစ်မနေလိုအပ်လျှင် အချို့ non object oriented thinking ၏ ဖြစ်နိုင်ခြေကို စစ်ပြီး ၎င်းကို ရှောင်ရန်ကြိုးစားပါ။

Indirect Returning

Java မှာ Indirect Returning သည် parameter တစ်ခုအဖြစ်သုံးသော object ကို modify ပြုလုပ်နိုင်သည်။ ထို့ကြောင့် လိုအပ်သည့်တွက်ချက်မှုများကို ပြုလုပ်သည့်နေရာမှာ modify-point (point p) method တစ်ခု ရှိနိုင်သည်။ ပြီးလျှင် နောက်ဆုံး p ၏ x နှင့် y ကို values သတ်မှတ်ရမည်။ ထို့ကြောင့်

```
void modify-point (point p)
   {
   //. . . . . . your code.
      p.x = . . . ;
      p.y = . . . ;
   }
```

ဆက်လက်၍

```
point pp = new point ( );
//.....other code...
modify_point(pp);
```

မှတ်ရန်မှာ ဤ method နှင့် အပေါ်မှ get-point method ကြားတွင် ခြားနားချက်ကို ဂရုစိုက်ကြည့်ပါ။ ဤ method သည် user အတွက် defined objects များကို ကောင်းကောင်း အလုပ်လုပ်သည်။ pre defined types (byte, short,..) များအတွက် အလုပ်မလုပ်ပါ။ ထို့ကြောင့် ဤ indirect method ကို သုံးခြင်းဖြင့် values ပြန်ရရှိစေသည့်ပုံစံကို ပိုပြီး ရှောင်ရှားသည်။

The This Facility

Java သည် တောင်းဆိုသည့် method တစ်ခုအတွင်းမှာ object ကို ဆိုလိုရန် facility တစ်ခုပေးသည်။ ၎င်းမှာ 'this' စကားလုံးကိုသုံးပြီး ပြုလုပ်သည်။ ဥပမာ သင်ရေးသည့်အခါ

```
do.set-date ( );
```

set-date ( ) အတွင်းမှ 'this' သည် do object ကို ကိုယ်စားပြုသည်။ အခြားနည်းဖြင့်ရေးလျှင်

```
d1.set-date ( );
```

'this' စကားလုံးသည် d1 (d0 သို့မဟုတ်) object ကို ဆိုလိုသည်။ ယေဘုယျအားဖြင့် 'this' ရဲ့ facility သည် တူညီသည့် class ရဲ့ နှစ်ခု သို့မဟုတ် နှစ်ခုထက်ပိုသော objects များကို ဆက်သွယ် လိုသည့်အခါ လိုအပ်သည်။ ပြီး တစ်ခုစီရဲ့ method မှ လိုတာကို တောင်းပါသည်။ ဥပမာ dates နှစ်ခုကြားမှ ကုန်လွန်သွားသော နေ့ရက်များကို တွက်ရန်

method တစ်ခုမှာ

```
int elapsed (date oth); {.....};
```

ဤ method သည် class date ရဲ့ member တစ်ခု ဖြစ်သည်။ ၎င်းသည် ကုန်လွန်သွားသော နေ့များကို int value တစ်ခုအဖြစ် ပြန်ပေးပါသည်။ d0နှင့် d1 သည် ကုန်လွန်သွားသောနေ့များကို တွက်ချက်လိုသည့် dates နှစ်ခုဖြစ်သည့်အခါ သင့် message တစ်ခု ပေးရပါမည်။

```
xx = d0.elapsed (d1);
```

xx သည် result ဖြစ်မည့် int variable တစ်ခု ဖြစ်သည်။ d0 အတွက် elapsed ( ) ကို သင် အသုံးပြုပြီးပြီ။ elapsed ( ) method အတွင်းမှာ 'this' စကားလုံးသည် d0 ကို ညွှန်းပါသည်။ oth parameter သည် ဖြတ်သွားသည့် d1 ကို ညွှန်းသည်။ သင်သည် d0 ၏ elapsed ကို တောင်းကတည်းက parameter တစ်ခုအဖြစ် ၎င်းကို မဖြတ်ကျော်ရသေးပါ။ system က ယခုမှ ၎င်း၏အကြောင်းကို သိရုံရှိပါသေးသည်။

မှတ်ရန်မှာ d0 နှင့် d1 နှစ်ခုလုံးသည် date objects ဖြစ်သည့်အတွက် တူညီသည့် job ကို ပြုလုပ်ရန် အောက်ပါ message ကို နောက်တစ်မျိုး ရေးလို့ရပါသေးသည်။

```
xx = d1.elapsd (d0);
```

ဤ case မှာ elapsed ( ) method အတွင်းမှ 'this' word သည် d1 ကို ကိုယ်စားပြုသည်။ 'this' စကားလုံးအား အသုံးပြု၍ elapsed ကို ပုံစံတစ်မျိုးဖြင့် ရေးပါမည်။

```
elapsed (date oth)
   {
   if (this.year == oth.year)
      {
      //.......details here..
      };
   else
      if(this year<oth.year)
      {
      //.......details here..
      };
   //......details here.
   } // end of elapsed
```

Over loading

ရှေ့တွင် ဖော်ပြထားသော date class ကို ရေးကြည့်ရအောင်။

```
class date
   {
```



```
// -Data (attributes)
int dayJ Month, year;
// -Methods (behaviour).
void set-date(int dd, int mm, int yy)
   {
   day = dd, month = mm, year = yy;
void print..date( )
   {
   System.out.println
     day+"-"month+"-"+ year);
   }
// Only two methods for the time being.
}
```

၎င်းတွင် int values ၃ ခုလုံးကိုယူပြီး date သတ်မှတ်ပေးသော set-date method တစ်ခုရှိသည်။ သို့သော် တူညီသော လ၊ နှစ်မှ date အသစ်တစ်ခုပေးချင်သည် သို့မဟုတ် တူညီသည့် date နှင့် month တွေကို တစ်နှစ်အတွင်းမှာ ပြောင်းချင်တာတွေ ရှိမည်။ ထို အခြေအနေမှာ အချိန်တိုင်း 1996, 1996, 1996 လို့ ထပ်ခါထပ်ခါပေးရမည်။ ထို့ကြောင့် Java သည် default' values ကိုသုံးသော date ကို define လုပ်ဖို့ ခွင့်ပြုသော overloading method ကို ပေးပါသည်။

ဤ method ကတော့

```
void set-date (int dd, int mm)
   {
   day = dd; month = mm; year = 1996;
   };
```

၎င်းတွင် set-date name ချင်း တူသည်။ သို့သော် day နှင့် month parameters နှစ်ခုသာ ပါဝင်သည်။ ဤ method သည် ရွပ်ထွေးမှုများ မဖြစ်ပေါ်စေဘဲ အခြား method များနှင့်အတူ date class မှာ တွဲသုံးနိုင်သည်။ အချို့ d0 object ၏ date ကို သတ်မှတ်ရန် အောက်ပါ message ပို့လျှင်

```
d0.set-date (8,10,1996);
```

3 parameters နှင့် old set-date ကို သုံးသည်။

do.set-date (8,10); ဟူ၍သာ ရေးလျှင် system သည် အသစ်ဖော်ပြထားတဲ့ set-date ကိုသုံးပြီး ဖော်ပြခြင်းဟု နားလည်ပြီး ရှေ့ကို အလုပ်ဆက်လုပ်သည်။ အခြား set-data method ကိုလည်း ဖော်ပြ၍ ရပါသေးသည်။

```
void set-date (int dd)
   {
   day = dd; month = 10; year = 1996;
   };
```

ပြီးလျှင် do.set-date (8); ဟု ပေးလျှင် date ကို မှန်ကန်စွာ သတ်မှတ်ရရှိမည်။

ဤ over methods များ အားလုံးသည် 8-10-1996 ကို date မှာ သတ်မှတ်ပေးပါသည်။ methods များကို အမျိုးမျိုး အသုံးပြုရေးသားခြင်းဖြင့် ၎င်းတို့၏ common name ကို overload ပြုလုပ်ပေးပါသည်။ ၎င်းတွင် set-date method ကို overload ပြုလုပ်ပြီးပြီ။ မှတ်ရန်မှာ overloaded method အားလုံးသည် တူညီသည့် class မှာ ရှိသင့်သည်။ ပြီးလျှင် တစ်ခုစီတွင် မတူညီသည့် signature ရှိပါမည်။ parameters များ၏ number များ တူညီသော်လည်း ၎င်းတို့၏ types သို့မဟုတ် classes များ မတူညီလျှင် signature ကွဲပြားသည်။ အောက်ပါ methods များတွင် မတူညီသော signatures များ ရှိကြသည်။

```
set_date (int, int,int);
set_date (double, int,int);
set_date (int, double,int);
set_date (int, int,double);
set_date (double, double, int);
set_date (int,double, double);
set_date (double, double, double);
set_date (int,int);
set_date (double ,int);
set_date (int, double);
set_date (double, double);
set_date (int);
set_date (double);
set_date ( );
```

Overloading methods များသည် အောက်ပါအခြေအနေနှစ်ခုကို အဆင်ပြေစေသည်။

၁။ မတူညီသော signatures များမှ ရွေးချယ်သုံးခွင့် ရှိသည်။

၂။ Class ကို define လုပ်ပြီးနောက် ရှေ့မှ code များကို မထိခိုက်စေဘဲ overloading ကို တစ်ခုထက် ပိုပြီးအသုံးပြုခြင်းဖြင့် ၎င်း class ကို အမြဲတမ်း ပြုပြင်နိုင်သည်။

Constructor

Java မှာ 'new' operator ကို သုံးသည့်အခါ new object တစ်ခု create လုပ်ရသည်။ ၎င်းတည်ဆောက်မှုတွင် system သည် object အတွက် သင့်လျော်သော memory ရရှိမည့် constructor ဟု ခေါ်သော method တစ်ခုကို ခေါ်ယူရပါမည်။ သင့်ဤ constructor ကို မရေးဖူးသေးသော်လည်း system သည် ထိုအလုပ်မျိုး သင့်အတွက် ပြုလုပ်ပြီးနေပြီ။ constructor သည် class နှင့် အမည်တူညီသည်ဆိုခြင်းကို သတိပြုရပါမည်။ Java မှာ သင်၏ own constructor ကိုလည်း define ပြုလုပ်ထားနိုင်ပါသည်။ ၎င်းသည် class နှင့် အမည်တူညီပြီး သင်ပြုလုပ်ချင်သောအရာအားလုံးကို ပြုလုပ်ပါမည်။ ထို့ပြင် system ၏ default constructor မှ ပြုလုပ်သော အရာများကိုလည်း ပြုလုပ်လိမ့်မည်။

ဥပမာ date class မှ ကျွန်ုပ်တို့၏ set-date method သည် date ကို သတ်မှတ်သည့် အလုပ် ပြုလုပ်ပါသည်။ ပြီးလျှင် constructor တစ်ခု ရှိမည်။ 'void' စကားလုံးကို ဖျက်

ပြီး date ကို အမည်တစ်ခု ပြောင်းရုံပဲ ဖြစ်ပါသည်။ ၎င်းအား ကျွန်ုပ်တို့၏ class date တွင်ကြည့်လျှင်

```
class date
   {
   //Data (attributes).
   int day, month, year;
   // Methods (behaviour).
   date(int dd, int mm, int yy)
   // constructor
      {
      day = dd, month = mm, year = yy;
      };
   void print-date( )
      {
      system.out.println
            (day = "-"+ month + "-" + year);
      }
// Only two methods for the time being.
}
```

constructor ကို ထပ်ခါထပ်ခါ သုံးနိုင်သည်။

```
class date
{
// Data (attributes).
int, day, month, year;
//Methods (behaviour).
//Onr method & 3 Consrtuctors.
date(int dd, int mm, int yy)
// constructor 1
{
day = dd; month = mm; year=yy;
};
date(int dd) //_______constructor 2
{
day = dd, month = 10, year = 1996;
};
date(int dd, int mm) // _______constructor 3
{
day = dd; month = mm, year = 1996;
} ;
void print-date( )
      {
      System.out.println
        day+"-"+month+"-"+ year);
   }
}
```

The Finalize Method

Java သည် သင်ဖော်ပြထားသည့် class တိုင်းတွင် finalize ဟူသော special

method တစ်ခုကို define ပြုလုပ်ဖို့ အခွင့်အရေးတစ်ခု ပေးပါသည်။ run time အတွင်း တွင် Java သည် class ၏ object တစ်ခုကို နေရာသတ်မှတ်ပေးသော memory ကို ရှာဖွေရယူလိုသည့်အခါ ဤ method ကို အလိုအလျောက် ခေါ်ယူပါသည်။ finalize method မှာ ကြာရှည်စွာကျန်ရစ်နေသော ခေါ်ယူထားသည့် special font တစ်ခု သို့မဟုတ် ဖွင့်ထားသော ဖိုင်ကဲ့သို့ ပြင်ပ resource များကို ဖယ်ရှားစေသော code ကို ရေးသင့်သည်။ အချို့ memory တွေကိုတော့ ဖယ်ရှားဖို့မလိုပါ။ အကြောင်းမှာ Java သည် အချို့ နေရာတွေမှာ ၎င်းတို့ကို အသုံးပြုသည်။ ယေဘုယျအားဖြင့်တော့ သင်နေ့စဉ်ရေးနေကျ programming တွေမှာ ၎င်းသည်မလိုအပ်သော်လည်း တချို့ case တွေမှာတော့လိုအပ်သည်။

၎င်း method မှာ

```
finalize (...)
{
//.....write your code
}
```

Static Data And Methods

ယေဘုယျအနေဖြင့် သင်သည် class တစ်ခုကို define ပြုလုပ်သည်။ ထို့နောက် declare, create ပြုလုပ်ပြီး အချို့ class (d0 နှင့် d1) များတွင် class ၏ objects များကို အသုံးပြုသည်။ Objects တစ်ခုစီမှာ ၎င်းတို့ရဲ့ ကိုယ်ပိုင် data ၏ methods များ ရှိကြသည်။ d0.month ရဲ့ value သည် 10 ဖြစ်ရမည်။ တစ်ချိန်တည်းမှာ d1.month ရဲ့ value သည် 6 ဖြစ်ရမည်။ သင်သည် d0.set-date ကို ရေးတဲ့အခါ d0 ၏ set-date method ကိုသုံးသည်။ ၎င်းတွင် d0.day, d0.month, d0.year values များကို ပြောင်းလို့ရသည်။ ၎င်းသည် d1 ၏ တူညီသော values များ (d1.data, d1.month, d1.year) နှင့် မသက်ဆိုင်ပါ။

ဘယ်လိုပင်ဖြစ်ဖြစ် ထို class ၏ objects အားလုံးအတွက် အများသုံး data methods များအဖြစ် class တစ်ခုမှာ အချို့ data နှင့် methods များကို ဖော်ပြဖို့လိုပါသည်။ ဥပမာ မိသားစုတစ်ခုကို ကိုယ်စားပြုသည့် class တစ်ခု ရှိသည်။ members အားလုံးမှာ တူညီသည့် surname ရှိမည်။ ထို့ကြောင့် surname ကို object variable တစ်ခုအဖြစ် ကြေညာနေ၍လည်း အကျိုးမရှိပါ။ class ရဲ့ objects အားလုံးကို common class variable တစ်ခုအဖြစ် ဖော်ပြသင့်ပါသည်။ အခြားဥပမာတစ်ခုမှာ lawyers များ၏ class ၏ objects အရေအတွက် ဖြစ်သည်။ ဤအရေအတွက်ကို class တစ်ခုလုံး၏ variable တစ်ခုအဖြစ် ကြေညာပြီးမှ class ၏ object အားလုံးကို တူညီတဲ့ value ပြသင့်သည်။ ၎င်းကို ပြုလုပ်ရန် datum (variable သို့ object) ၏ ရှေ့မှ 'static' စကားလုံးတစ်ခု ထားရှိရမည်။ subject တစ်ခုမှာ ရရှိသော ကျောင်းသားများ၏ ရမှတ်

များနှင့်ဆက်စပ်ပြီး အများဆုံးဖြစ်နိုင်သည့် အမှတ်များသည် အားလုံးအတွက် တူညီကြပြီး ကျောင်းသား တစ်ဦးစီအတွက် datum ထက် class datum အဖြစ် သဘောထားသင့်သည်။

class marksheet ကို ဖော်ပြပါမည်။

```
class marksheet
{
static int max-marks;
int marks-obtained;
//....other things
   }
```

ဤ variable max-marks ကို သင့် code ၏ အချို့နေရာမှာ တန်ဖိုးတစ်ခုနှင့် initialize လုပ်သင့်ပါသည်။

```
marksheet.max-marks = 100;
```

defining နှင့် initialization ကို တစ်ခုတည်းနှင့်လည်း ပေါင်းစပ်ဖော်ပြနိုင်ပါသည်။

```
class marksheet;
{
static int max_marks = 100;
//defned and anitialized also.
int marks-obtained;
//.....other things
}
```

Organization တစ်ခုအတွက် organization ၏ foundation date သည် employees အားလုံးအတွက် တူညီသည်။ ထို့ကြောင့် class employee ကို သတ်မှတ်သည့်အခါ ၎င်းကို class variable တစ်ခုအဖြစ်သာ ကြေညာသင့်ပါသည်။ 'static' စကားလုံးကို အသုံးပြုရပါမည်။

```
class employee
{
// ..........Data
String name;
date birth_date, joining-date;
static date FoundationDate;
//.......... Methods
//.......No method for the time being.
}
```

Code ၏ အချို့နေရာမှာ employee ၏ variable ee 1 ကို အောက်ပါအတိုင်း ကြေညာပြီး

```
employee ee1;
```

FoundationDate ကို သတ်မှတ်ချင်လျှင်

```
employee.FoundationDate. Set-date (20,8,1986);
```

ဟု ရေးနိုင်ပါသည်။

သင်သည် static variable နှင့်အတူ တစ်စုံတစ်ခု ပြုလုပ်သည့်အခါ သို့မဟုတ် **print** လုပ်သည့်အခါ၊ object name အစား class name ကို သုံးနိုင်ပါသည်။ ဘယ်လိုပဲ ဖြစ်ဖြစ် ၎င်းသည် ရှုပ်ထွေးမှုတစ်ခု မဟုတ်ပါ။ class datum သည် object datum အတွက်လည်း အသုံးဝင်ပါသည်။

```
ee1.foundationDate.set-data (20,8,1986);
```

ဤကဲ့သို့လည်း ရေးနိုင်သည်။ တူညီသည့် အချက်များမှာ static methods များကို လည်း အသုံးပြုကြသည်။

ပို၍ အသေးစိတ်ကျတဲ့ ဥပမာကို ဖော်ပြထားပါသည်။

```
class cc1
{
void xx( )
{Systemout.pribtln("xx invoked");}
static void yy( )
{System.out.println("yy invoked");}
   }
class test1
{
public static void main =(String args[1)
{
clobj1c = new cc1( );
System.out.print
{"Using object obj1. ...");
obj1.xx( );
System.out.print
("Using class cc1. ...")
cc 1.yy ( ) ; ) )
```

output မှာ

```
Using object obj 1..... xx invoked
Using object obj 1 : yy invoked
Using class cc1 ..... yy invoked
```

yy0 method သည် static ဖြစ်သည်။ ဖော်ပြပါအတိုင်း class တစ်ခု သို့မဟုတ် object တစ်ခုသို့ message ပို့ခြင်းဖြင့် တောင်းဆိုနိုင်ပါသည်။ အခြားတစ်ဖက်တွင် xx ( ) သည် non-static method တစ်ခု ဖြစ်သည်။ object တစ်ခုအစား class သို့ message ပေးပို့ခြင်းဖြင့် no-static datum သို့မဟုတ် method ကို ချဉ်းကပ်ရန် ကြိုးစားလျှင် compiler သည် အောက်ပါ error ကို တွေ့ရှိသည်။


```
test1.java:20: Can't make static refrence to method void xx(
)in class cc1.
Cc1.xx( );
^
1 error
```

ယခုတစ်ခါ 'static' ၏ အခြားသာဓကတစ်ခု ကြည့်ကြစို့။

```
Class cc1
{
int value;
void yy( )
{
system.out.println("value = " + value);
   }
class cc2
{
static cc1 iiii, hhhh, // Declaration.
//-____Creation inside a block.
static {hhhh=new cc1( ); hhhh. value = 4;}
//--Creation inside another block.
static {iiii=new cc1( ); iiii. value=5;}
   }
class test1
{
public staic void main(String args [ ])
   {
cc2 obj2 = nw cc2( );
system.out.print
("Using object obj2 and object hhhh');
obj1:hhhh.yy( );
system.out.print
("Using class cc2 and object iiii");
cc2.iiii. yy ( );
   }
}
```

output မှာ

```
Using object obj2 and hhhh value = 4
Using class cc2 and object iiii value = 5
```

Class cc2 မှာ class cc1 ၏ hhhh နှင့် iiii data ရှိသည်။ နှစ်ခုလုံးကို static နှင့် ဖော်ပြသည်။ နှစ်ခုလုံးကို တစ်နေရာတည်းမှာ ကြေညာပြီး static blocks အတွင်း တစ်နေရာစီမှာ initialize လုပ်သည်။ ၎င်းသည် ပုံစံအသစ်တစ်ခုဖြစ်ပြီး ၎င်းကို ဂရုတစိုက် ကြည့်ကြပါ။ static block တစ်ခုသည် static datum တစ်ခု၏ declaration လုပ်ပြီးနောက် class အတွင်း တစ်နေရာမှာ ဖြစ်ပေါ်နိုင်သည်။ အပေါ်မှ ဥပမာတွင် ပြထားသည့်အတိုင်း class အတွင်းမှ မတူညီသည့်နေရာများမှာ data တွေကို ပို၍ initialize ပြုလုပ်ဖို့ static blocks များ အများအပြား ရှိနိုင်သည်။ output ကို မထိခိုက်

ဘဲ single static block တစ်ခုတည်းမှာ ၎င်းတို့အားလုံးကို ထားရှိနိုင်သည်။

```
class cc2
{
static cc1, hhhh; // Declaration.
// -Creation inside a block.
static
{
hhhh = new cc1( ); hhhh.value = 4;
iiii = new cc1( ), iiii.value = 5;
} .. end of static block
      }
```


အခန်း (၇)
နောက်ထပ် ဘာတွေရှိသေးလဲ

ယခု လေ့လာပြီးသည့် Java အကြောင်းက အစိတ်အပိုင်းတစ်ခုမျှသာ ဖြစ်ပါသည်။ Java ၏ features အားလုံးကို ဖော်ပြရန်မှာ ဤမျှနှင့် မလုံလောက်ပါ။ ယခု ဒီသင်ခန်းစာမှာ Java ရဲ့ အခြားပုံစံအမျိုးမျိုးကို မိတ်ဆက်အကျဉ်းမျှ ဖော်ပြပေးပါမည်။ ၎င်းတို့၏ အသေးစိတ်ကို သိရန်မှာမူ ဤ subjectsများရဲ့ individual book များကို ဆက်လက် ဖတ်ရှုလေ့လာရပါမည်။

Java Servlets

Servlet များသည် web connection ရဲ့ server side မှာ ဆောင်ရွက်သော program ငယ်များ ဖြစ်ပါသည်။ Servlets ၏ အကျိုးအာနိသင်များကို နားလည်ရန်သင့်မှာ Web browser များနှင့် server များသည် user ကို ကျေနပ်မှုရရှိရန် မည်သို့ ဆောင်ရွက်မလဲ ဆိုတဲ့ အခြေခံနားလည်မှု ရှိရပါမည်။ ပုံသေ Web Page တစ်ခုအတွက် တောင်းဆိုမှုတစ်ခု စဉ်းစားကြည့်ပါ။ User တစ်ယောက်သည် browser တစ်ခုဆီကို Uniform Resource Locator (URL) တစ်ခု စတင်ဝင်ရောက်သည်။ Browser သည် သင့်လျော်သော Web server ဆီကို HTTP တောင်းဆိုမှုတစ်ခု ပြုလုပ်လိုက်သည်။ Web server သည် ဤတောင်းဆိုမှုကို တိကျတဲ့ဖိုင်တစ်ခုသို့ စီစဉ်ပို့ပေးသည်။ ထိုဖိုင်သည် browserဆီကို HTTP response တစ်ခု ပြန်ပို့သည်။ Response ထဲမှ HTTP header သည် လိုချင်တဲ့ပုံစံကို ညွှန်ပြပေးပါသည်။ Multipurpose Internet Mail Extensions (MIME) ကို ဤရည်ရွယ်ချက်အတွက်အသုံးပြုပါသည်။ ဥပမာ ရိုးရိုး ASCII text မှာ text/plain ၏ MIME type တစ်ခု ရှိသည်။ Web page တစ်ခုရဲ့ Hypertext Markup Language (HTML) source code မှာ text/html ရဲ့ MIME type တစ်ခု ရှိပါသည်။ ယခု dynamic content ကို စဉ်းစားကြည့်ပါ။ Online bookstore တစ်ခုသည် book prices, avilability, orders အစရှိသဖြင့် ပါဝင်သော လုပ်ငန်းများ၏ information ကို သိမ်းဆည်းရန် database တစ်ခုသုံးသည် ဆိုပါစို့။ ၎င်းသည် Web page မှတစ်ဆင့် customers များဆီသို့ ဤ information ကို အလွယ်တကူ ရောက်ရှိချင်သည်။ Web page များ၏ ကျေနပ်မှုအတွက် database မှာ နောက်ဆုံးပေါ် information ကို တုံ့ပြန်ရရှိရန် မြန်မြန်ဆန်ဆန် ပြုလုပ်ပေးရပါမည်။

စောစောပိုင်းမှ ၎င်းသည် HTTP request မှ data ဖတ်ရန်နှင့် HTTP responce ဆီသို့ data ရေးရန် ပြုလုပ်နိုင်သော Common Gateway Interface (CGI) လို့ ခေါ်သည့် interface တစ်ခုမှတစ်ဆင့် Web server နှင့် ဆက်သွယ်ခဲ့သည်။ အမျိုးမျိုးသော languages များကို C, C++ နှင့် Perl တို့ ပါဝင်သော CGI program များ တည်ဆောက်ရန် သုံးသည်။ မည်သို့ပင်ဖြစ်စေ CGI သည် ပြဿနာများကို တွေ့ကြုံဖြေရှင်းရတတ်သည်။ Client request တစ်ခုစီအတွက် processor တစ်ခု တည်ဆောက်ခြင်းမှာ ဈေးမြင့်သည်။ processor နှင့် memory resource များ ပါဝင်သည်။ ထို့ပြင် client request တစ်ခုစီအတွက် database connections များအား ဖွင့်ခြင်း၊ ပိတ်ခြင်းတို့အတွက် လည်း ကုန်ကျစရိတ် မြင့်မားသည်။ ထို့အပြင် CGI program များသည် platform-independent မဖြစ်ပါ။ အခြား နည်းပညာများသည် servlets ကို မိတ်ဆက်သည်။

Servlet များသည် CGI အပေါ်မှာ အမျိုးမျိုးသော အကျိုးကျေးဇူးများကို ပေးသည်။

Performance ၎င်းသည် ပို၍ထင်ရှားသည်။ Servlet များသည် Web server ရဲ့ address space အတွင်းမှာ စီမံဆောင်ရွက်သည်။ Client request တစ်ခုစီကို ဖြစ်စေသည့် process ကို တည်ဆောက်ခြင်းသည် မလိုအပ်ပါ။

Platform Servlet များသည် platform-independent ဖြစ်သည်။ အကြောင်းမှာ ၎င်းတို့သည် Java နှင့် ရေးသားထား၍ ဖြစ်သည်။ Vendors များဖြစ်သော Sun, Netscape နှင့် Microsoft တို့မှ အမျိုးမျိုးသော Web servers များသည် Servlet API ကို ပေးသည်။ API အတွက် တီထွင်ထားတဲ့ program များသည် recomplica-tion မလုပ်ဘဲနဲ့ သူတို့ရဲ့ နေရာဝန်းကျင်မှာ ရွှေ့လျားနိုင်သည်။

Security Server ပေါ်မှ Java Security Manager သည် server machine ပေါ်မှ resource များကို ကာကွယ်ရန် တားမြစ်ချက်များကို လိုက်နာစေသည်။ အချို့ servlet များကို ယုံသုံး၍ရပြီး အချို့ကို ယုံသုံး၍ မရတာကို သင်တွေ့ရလိမ့်မည်။

Functionality Java class library ၏ ပြည့်ဝသော functionality မှာ servlet တစ်ခုကို အသုံးဝင်စေသည်။ ၎င်းသည် socket နှင့် RMI mechanism မှတစ်ဆင့် applets, databases သို့မဟုတ် အခြား software များနှင့် ဆက်သွယ်နိုင်သည်။

သင် servlets အကြောင်းကို ပိုပြီး လေ့လာရန် စိတ်အားထက်သန်လျှင် Sun နှင့် တိုက်ရိုက် စာအဆက်အသွယ်ဖြင့် တိုင်ပင်မေးမြန်းပါ။ Servlets API ကို အထောက်အကူပြုသော Web servers အမျိုးမျိုးနဲ့ servletrunner လို့ ခေါ်တဲ့ ဖိုင်ရှိပါသည်။

Java BEANS

Java Bean တစ်ခုသည် မတူညီသည့် နေရာဝန်းကျင်များတွင် အသုံးပြုရန် တီထွင်ထားပြီး ဖြစ်သည့် software component တစ်ခု ဖြစ်သည်။ Bean ၏ စွမ်းဆောင်မှုတွင် တားမြစ်ချက် မရှိပါ။ Simple function တစ်ခု (ဥပမာ - spelling check) သို့မဟုတ် complex function တစ်ခုကို ဆောင်ရွက်ပေးသည်။ Bean တစ်ခုသည် end user တစ်ယောက်ကို မြင်စေသည်။ ဥပမာ graphical user interface ပေါ်မှ button တစ်ခု ဖြစ်သည်။ Bean တစ်ခုသည် user တစ်ယောက်ကို မမြင်စေပါ။ Real time မှာ multimedia information ၏ stream တစ်ခုကို ပြန်လည်ဖော်ထုတ်ရန် software သည် building block အမျိုးအစား၏ ဥပမာတစ်ခု ဖြစ်သည်။ အချုပ်ပြောရလျှင် Bean တစ်ခုကို အခြားဖြန့်ခွဲထားသော component များနှင့် စုပေါင်း အလုပ်လုပ်ရန် သို့မဟုတ် user ၏ workstation တစ်ခုမှာပဲ ကိုယ်ပိုင်လုပ်ရန်အတွက် တီထွင်ထားသည်။ Data points အစုတစ်ခုမှ pie chart တစ်ခု ပြုလုပ်ရန် software သည် locally execute ပြုလုပ်နိုင်သည့် Bean တစ်ခု ဖြစ်သည်။ မည်သို့ပင်ဖြစ်စေ stock သို့မဟုတ် commodities exchange မှ real-time price information ပေးသော Bean သည် ၎င်း၏ data များ ရရှိရန် အခြား ဖြန့်ခွဲထားသည့် software များနှင့် ပူးပေါင်းဆောင်ရွက်ဖို့ လိုအပ်သည်။

Java Beans technology သည် Java programming ရဲ့ အလွန်အရေးကြီးသော အစိတ်အပိုင်းတစ်ခုဖြစ်ပြီး component software တည်ဆောက်ခြင်းသည် ရှေ့အနာဂတ်မှာ Java programmer ၏ jobs အများစုတွင် အရေးကြီးသော အပိုင်းတစ်ခု ဖြစ်လာပါမည်။

Java And Networking

Java သည် network concepts အားလုံးနှင့် မည်သို့ ဆက်စပ်သလဲ။ Java သည် ရှိပြီးသား I/O interface stream ကို ချဲ့ထွင်ခြင်းဖြင့်လည်းကောင်း network တစ်လျှောက် I/O objects တည်ဆောက်ရန် လိုအပ်သည့် features များ ပေါင်းထည့်ခြင်းဖြင့်လည်းကောင်း TCP/IP ကို အထောက်အကူပြုပါသည်။ JAVA သည် TCP နှင့် VDP protocol နှစ်ခုစလုံးကို အထောက်အကူပြုပါသည်။ TCP ကို network မှ stream-based I/O အတွက် သုံးတယ်။ UDP ကတော့ point-to-point datagram-oriented ပုံစံ ပြုလုပ်ပေးပါသည်။

Java Swings

Swing သည် AWT မှာထက် ပို၍ စွမ်းဆောင်ရည်ရှိပြီး ပြုပြင်လွယ်သော components များကိုပေးသော Classes အစုတစ်စုဖြစ်သည်။ ထို့အပြင် familiar component များဖြစ်သော buttons, check boxes နှင့် labels များအပြင် tabletd

panes, scroll panes, trees နှင့် tablets များကိုလည်း ပေးသည်။ Button ကဲ့သို့ familiar components များပင်လျှင် swing တွင် ပိုမိုစွမ်းဆောင်နိုင်သည်။ ဥပမာ button တစ်ခုမှာ image တစ်ခုနှင့် ၎င်းနှင့်ဆက်စပ်သော text-string တစ်ခု ရှိသည်။ AWT components နှင့် မတူတာကတော့ swing components များသည် platform-specific code ဖြင့် မဖော်ပြကြပါ။

သူတို့ကို Java နှင့် လုံးလုံးလျားလျား ရေးသားထားခြင်း ဖြစ်ခြင်းကြောင့် platform-independent ဖြစ်သည်။ Light weight ကို elements များ ရှင်းပြရန် သုံးသည်။ Swing packages မှ classes နှင့် interfaces သည် ခိုင်ခံ့ကောင်းမွန်ပြီး ဉာဏ်သင်ခန်းစာသည် အနည်းငယ် လေ့လာရုံမျှနှင့် ကျယ်ကျယ်ပြန့်ပြန့် သိနိုင်ပါသည်။ Sewing သည် သင့်ကိုယ်ပိုင် အနှာဂတ်မှာ စူးစမ်းရှာဖွေလေ့လာရန်ကောင်းတဲ့ နယ်ပယ်တစ်ခု ဖြစ်ပါသည်။

Swing-related classes များသည် javax.swing တွင် ပါဝင်ပြီး ၎င်း၏ subpackages များဖြစ်သည့် javax.swing. tree မှာလည်း ပါဝင်သည်။

JDBC

၎င်းမှာ client/server software components များကို develop လုပ်ရန် သုံးနိုင်တဲ့ classes အစုတစ်ခု ဖြစ်တယ်။ သင်သည် အချိန်တိုင်းမှာ web's A Client Program, image files နှင့် URL ကိုသုံးသော အခြား documents များနှင့် ရှိနေသည်။ မတူညီသော Server program များသည် requested information ကို ပေးသည်။ JDBC library တွင် database usage နှင့် ဆက်စပ်သော အလုပ်တစ်ခုစီအတွက် classes များ ပါဝင်သည်။

ကိုယ်တိုင်လေ့လာ JAVA သင်ခန်းစာ

ဒီစာအုပ်မှာ...

- Java Programming နှင့် ပတ်သက်၍ စတင်ဖြစ်ပေါ်လာပုံ၊ အဆင့်ဆင့် တိုးတက် ပြောင်းလဲပုံများကို အခြေခံမှစပြီး အသေးစိတ် ဖော်ပြထားသည်။
- Java ဘာသာစကားနှင့် ဆက်နွယ်ပြီး ပရိုဂရမ်ရေးသော ဘာသာစကားများ၊ ပရိုဂရမ်ရေးသားမှု အခြေခံများကို အကျဉ်းချုံး၍ ဖော်ပြထားသည်။
- Java ဘာသာစကားနှင့် C ++ ဘာသာစကားတို့တွင် ပါရှိသည့် တူညီသော စွမ်းဆောင်ရည်များ၊ C++ တွင် မရှိသော Java feature များ၊ Java တွင် အသုံးမပြုသော C++ စွမ်းဆောင်ရည်များ အကြောင်း အသေးစိတ် ဖော်ပြထားသည်။
- Java ပရိုဂရမ်တစ်ပုဒ် လက်တွေ့ရေးသားနိုင်ရန် အခြေခံမှစ၍ သင်ကြားပေးထားပြီး ready made ပရိုဂရမ်များ (Applets) အသုံးပြုနည်း၊ Applet များကို အင်တာနက်တွင် ရှာဖွေ အသုံးပြုနည်းများ ပါဝင်သည်။
- OOP ခေါ် Object Oriented Programming အကြောင်း မိတ်ဆက် တင်ပြပေးထားသည်။

THE DUMMIES WAY®

ဥပုန်ကြားချက်အဆင့်ဆင့်ကို လိုရင်းတိုရှင်း ဖော်ပြထားသည်။ Task နှင့် topic များကို အလွယ်တကူ ရှာဖွေကြည့်နိုင်ပြီး ဖြေရှင်းချက်များကိုလည်း မြန်မြန်ဆန်ဆန် လေ့လာနိုင်သည်။

Myanmar Heritage