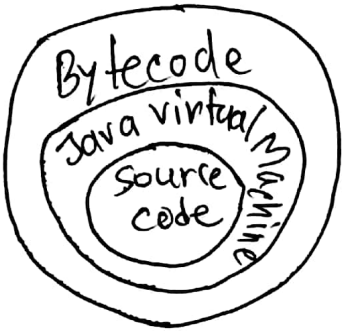
SA | CSE 1203 | 8/A | 10.10.17

Java

Java is platform Independent

Bytecode ← Converted in machine language



SA | CSE 1203 | 8/B | 11.10.17

```
public class MyClass {
    public static void main(String[] args) {
        // body;
        student ob = new student ("  ", 123, 3.12);
    }
```

(Array out of bound) when array range এর বাইরে Data output এ দেয়।

Important

super() {

SA madam

CSE 1203

8/C

22.10.17

Methods (function):

```
import java.util.Scanner;
class Hello{
  public static void main(String args[]){
     Scanner in = new Scanner(System.in);
     double a =in.nextDouble();
     double b = in.nextDouble();
     System.out.println (max(a,b));
  }
pubic static double max(double num1, double num2)
{
   if(num1>num2)
     return num1;
   else
     return num2;
}
```

Method Overload (like function overload):

public static ~~voi~~ int max (int num1, int num2);

public static double max (double num1, double num2);

try to remove ambiguity, if occurs,

random number generation: Math.random()

(**) Math.exp(x)
Math.log(2x)
Math.sin(x)

Array:

(old) → datatype array_name = new datatype [size];

datatype[] array-name = new datatype [size];

datatype array_name[] = new datatype [size];

i.e. → int x[] = new int [5];

Super class / Base class / Parent class
↓ ↓ ↓
Sub class / derive class / child class

class A {}; class B extends public A

```
class  circle {
    public: double radius = 10;
    double area();
    }

public class  filename {
   public static void main (String arg[])
{
    circle ob = new circle ();
    circle ob1 = new circle (10);
    circle Ob(10)
    System.out.println (ob.area());

}
```

SA madam CSE 1203 2/D

29 ~~28~~ 10.17

Immutable objects and class:

-occationally it is desireable to create and object whose content can not be changed once the object is created. We call sueh an object is immutable object and its class an immutable class.

for a class to be Immutable, it must meet the following requirments;

1) Declare all data field private
2) provide no mutator method
3) provide no accessor method that returns a reference to the data field that is mutable.

```
public class student {
    private int id;
    private String name;        } data field
  // public student (int x, string N) { id=x; name=N;}
or, public getid (int x) { id = x; }
```

```
public class Test {
public static void main (string args[]) {
   student s = new student (1603108,"Mr. X");
```

or `s.getid(1603108);`

```
string s = new String();
```

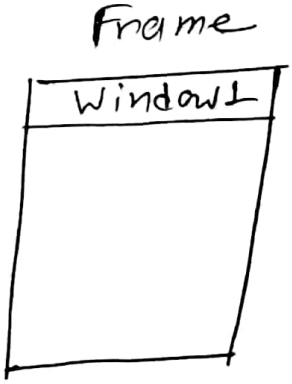
w	o	r	l	d

```
s.charAt(0); //returns the 0th index's
                character of the string.
```

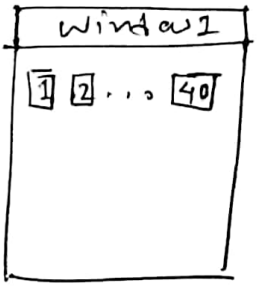
```
class B {
  public void println()
        { for (i=0; i<40; i++)
            System.out.println("_");
        }
class A extends B
  public A()
  { super();
    System.out.println(" * ");
     Super.println();
}
```

```
public class Test {
public static void main(String args[]){
    A ob = new A();
```

```
JFrame F = new JFrame();
F.setTitle("Windeo 1");
F.setlocation(200, 100);
F.setvisible(true);
```




```
JPanel p = new JPanel();
for (i=0; i<40; i++)
```




```
JButton t = new JButton();
JLabel l = new JLabel("First name");
JcheckBox JB = new JcheckBox("Bold");
JComboBox JCB = new JComboBox(new
        String[] {"January",
        "February"...);
```

```
JPanel p = new JPanel();
p.add(jt);
p.add(l);
p.add(JB);
p.add(JCoB);
add(p);
setSize(450,70);
setDefaultcloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
JPanel p = new JPanel();
p.add(bt);
p.add(l);
p.add(JB);
p.add(JCoB);
add(p);
setSize(450,70);
setDefaultcloseOperation(JFrame.EXIT_ON_CLOSE);
```

SA madam | CSE 1203 | 10/A | 31.10.17

- Abstract class

```
public class abstract geometryObject {
  // Data fields  private double x_cor;
                  private double y_cor;
  // Methods
  // at least one abstract method (শুধু declaration থাকে definition নেই)

  public geometryObject() { }
  public double getarea() { }
  public double getperemeter() { }
```

Interface :

```
class modifier interface interface_name {
  public static final int k = 1;
  abstract method
  }
class circle extends geometric implements interface_name{
  }
```

Example:

```
public interface edible {
   public abstract String howtoeat () { }

}
```

chicken → Animal → superclass

Apple → Fruit

Orange → Fruit

```
chicken c = new chicken();
class chicken extends Abst Animal implement edible{
  String howtoeat () {
    system.out.println (" oh ok chicken, fry it ");
```

Java তে Mutiple inheritance support করে না but Multiple interface support করে।

HandelEnvent.java

```
import javax.swing.*;
import java.awt.event.*;
public class HandleEvent extends JFrame{
  public HandleEvent(){
    JButton Jbtok = new JButton("ok");
    JButton Jbtcancel = new JButton("cancel");
    JPanel P = new JPanel();
    P.add(Jbtok); p.add(Jbtcancel);
    add(P);
    Oklistenerclass listener1 = new OklistenerClass();
    cancellistenerclass listener2 = new cancelListenerclass();
    Jbtok.addActionListener(listener1);
    Jbtcancel addActionListener(listener2);
  }
  public static void main(String args[]) {
  JFrame Frame = new HandleEvent();
  JFrame settitle(,);
  JFrame setsize(,);
  JFrame setlocation(,);
  }
```

```
class okListenerclass implements ActionListener {
public void actionRerformed (ActionEvent ) {
  System.out.println ("Ok button Clicked");
  }
class cancellistener class implements Actionlistener {
public  void action performed (ActionEvent)
 System.out.println ("Cancel button Clicked");
 }
```

SA.madam CSE 1203 10/B 1.11.17

Exeption Handling

```
import util java.util.scanner;
public class exeptionhandling {
public static void main (String args[i]){
    scanner input = new scanner (System.in);
    int number1 = input.nextInt();
    int number2 = Input.nextInt();
//System.put.println ("number1/number2");
 }
```

```
try {
    if(number2==0)
     throw new ArithmeticExeption("Divisor cannot
                                   be zero");
system.out.println(number1/number2);
}
catch (Arithmatic exception ex) {
  system.out.println("Exeeption occured");
}
System.out.println("Execution continues");
}
try {
    statement;
}
catch (exception ex) {
      handling ex;
}
finally {
        final statement;
        }
```

SA madam | CSE 1203 | 10/D | 05.11.17

```
datatype[] array_name = new datatype[Array_size]

int[] a = new int[10];
int[] a = {1, 5, 7, 3}
  for (i=0; i<a.length ; i++)
     a[i] = input.nextInt();

String s = new String(String_level);
String s = new String("Welcome to RUET");
equal String s;
      s = "Welcome to RUET";
char[] a={'W','e','l','c','o','M','e'};
String message= new String(a);
System.out.println(message.charAt(0));
string s1 = "Welcome to";
string s2 = "RUET";
string s3 = s1+s2;
equal or, String s3 = s1.concat(s2);
```

□ checkPalindrom.java

```
~~public~~
import java.util.scanner;
public class checkPalindrom {
    public static void main(String[] args){
      Scanner in= new Scanner(System.in);
      String s= in.next();
      if (isPalindrom(s))
        System.out.println(s+" is palindrom");
    else
      System.out.println(s+" is not palindrom");
}
public static boolean isPalindrom(String s) {
  int low=0;
  int high = s.length -1;
 while (low < high){
 if(S.charAt(low) != S.charAt(high))
  return false;
 low++;
 high--;
} return true; } }
```

File Input & Output:

```
public static void main(String args[]){
   java.io.File file =new javaio. File("Test.txt");
   if (file.exist()) {
      System.out.println("File Already exists ");
      System.exit(0);
   }
   java.io.PrintWriter output = new java.io.PrintWriter(file);
   Output.print ("Mr.X");
   Output.print (100);
   Output.print ("Mr.Y");
   Output.println(90);
   output.close();
}
```

file থেকে Input নেওয়ার জন্য ↴

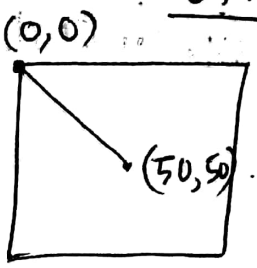
```
Scanner input =new Scanner (file);
while (in.hasNext()) {
   String name = input.next();
   int i = input.nextInt();
   }
in.close();
}
```

SA madam | CSE 1203 | 11/A

07.11.17

drawline (x_1, y_1, x_2, y_2)

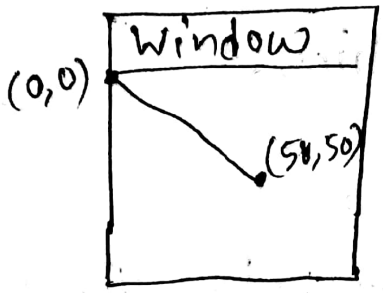
drawline (0, 0, 50, 50) ⟶




```
import java.awt.Graphics;
import javax.swing.*;
public class TestLine extends JFrame {
  private JLabel t = new JLabel ("Banner");
  public TestLine () {
    add (t);
    System.out.println( t.getGraphics());
  }
 public static void main ( String args[]) {
    TestLine frame = new TestLine ();
     frame.setTitle ("Window");
     frame.setSize (200, 200);
     frame.setVisible (true);
     frame.setLocationRelativeTo (null);
     frame.setDefaultCloseOperation (null, EXIT_ON_CLOSE);
```

```
    Graphics G1 = frame.t.getGraphics();
    G1.drawLine (0, 0, 50, 50);
  }
}
```

Output :

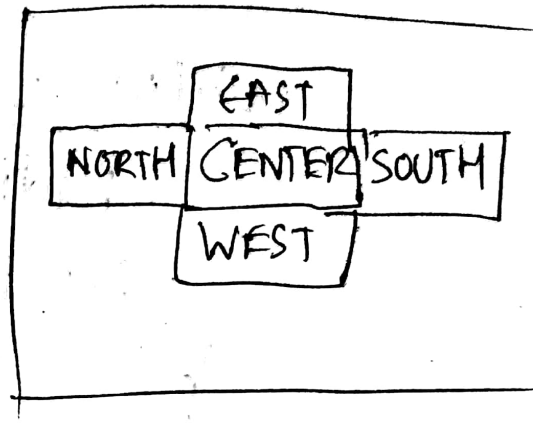


LAYOUT MANAGER

1) Border Layout
2) Flow Layout
3) Grid Layout

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ControlBall extends JFrame {
  private JButton Jbtenlarge = new JButton("Enlarge");
  private JButton Jbtshrink = new JButton("Shrink");
  private BallCanvas canvas = new Ballcanvas ();
  public controlBall() {
     JPanel Panel = new JPanel();
     panel.add(Jbtenlarge);
     panel.add(Jbtshrink);
```

```
this.add(canvas, BorderLayout.CENTER);
this.add(panel, BorderLayout.SOUTH);
Jbtenlarge.addActionListener(new
        ActionListener()){
  public void actionPerformed(ActionEvent e){
    canvas.enlarge();
  };
```




```
Jbtshrink.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    { canvas.shrink();
    }
public static void main(String args[]){
   controlBall frame = new ControlBall();
   frame.settite
   frame.setSize
   fra
}
```

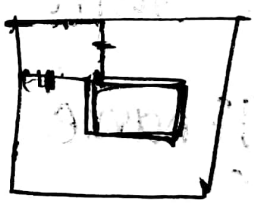
```
public static BallCanvas extends JPanel {
  private int radius = 5;
public void enlarge() {
      radius += 1;
      repaint();
  }
public void shrink() {
      radius -= 1;
      repaint();
  }
protected void paintComponent (Graphics G) {
  super.paintComponent (G);
  G.drawOval (getwidth()/2 - radius,
              getheight()/2 - radius,
              2*radius, 2*radius);
  }
Jbtshrink.addActionlistener( new Action
                              listner(){
public void ActionPerformed(Action Event e).
{ canvas.shink();
}
```

```
import java.awt.*;
import javax.swing.*;
public class MyFrame extends JFrame {
 public myframe(){
JFrame frame = new JFrame();
JLabel label = new JLabel("My Frame");
frame.setsize(200,300);
Frame.setLocation(100,200);
 JLabel label = new JLabel("Welcome to Java");
 JPanel JP = new JPanel();
 JP.add(label);
 add(JP);
}

public static void main(String args[]) {
    JFrame frame = new Jframe myframe();
    frame.setTitle("My Frame");
    frame.setsize(200,300);
    frame.setLocation(100,200);
    frame.setDefaultcloseOperation(JFrame,EXIT_ON_
                                   CLOSE);
    frame.setvisible(true);
}
```

```
import java.JOptionPane;

public class DialogBox {
   public static void main(String args[]) {
      JOptionPane.showMessageDialog(null,"Welcome
                                    to Java","Dialog
                                    Box",
                                    JOptionPane.EXIT_
                                    ON_CLOSE);
   }
```

1) FlowLayout

```
JPanel p = new JPanel(FlowLayout.LEFT, 2, 2);
```



2) GridLayout

```
JPanel p = new JPanel.(GridLayout(4,2,5,5));
```

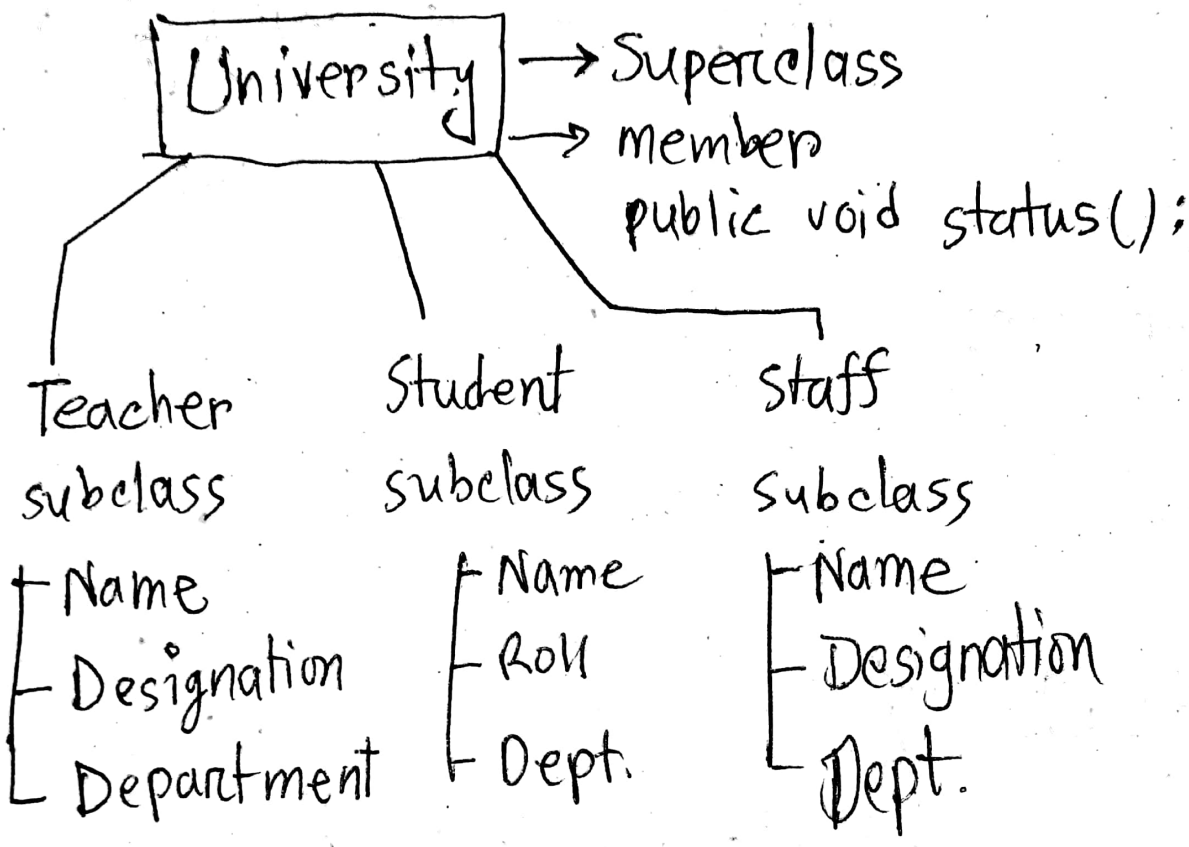
GridLayout(row size, column size, pixel distance(x), pixel distance (Y));

3) Border Layout

```
JPanel p = new JPanel(BorderLayout.CENTER);
```

SA madam | CSE 1203 | 11/B

08.11.17




```
class university {
  public void working_hour() {
public university ( ) {          }
class. Teacher extends university {
super. working_hour();

}
```

```
public university {
  private int n, string N, D;
  public university (string Name, string designation,
                     String dept, int roll) {

  N = Name;
  D = designation;
  R = roll
}

university () {

public Teacher extends university {
  super( Name, designation, dept, roll)

  }

public static void main( String argsL]) {
  Teacher T = new Teacher("X", "Teacher", "CSE"
                          123);
```

SA madam | CSE 1203 | 12/B
21.11.17

Multithreading

Thread:

A thread is a flow of execution from beginning to an end, of a task in a program.

*Multitask গুলো যখন CPU share করে নেয় তাকে Multithreading বলে।

interface

Runnable

```
public void run() { int i;
  for(i=0; i<100; i++)
    System.out.println(S);
  }
```

```
class Test implements Runnable { String str;
  public Test (String s) { str = s; }
}
```

এখানে হবে

```
class Test1 implements Runnable { int i1;
  public Test1 (int i) { i1=i; }
public void run() { int i;
  for(i=0; i<100; i++)
    System.out.println (i);
  }
}
```

```
public class ThreadDemo {
  public static void main(String args[]) {
    Test t1 = new Test("A");
    Test t2 = new Test("B");
    Test1 t3 = new Test1(100);
  Thread thread1 = new Thread(t1);
  Thread thread2 = new Thread(t2);
  Thread thread3 = new Thread(t3);
  thread1.start();
  thread2.start();
  thread3.start();
  }
}
```

```
public class ThreadDemo extends Thread {
 class Test { string str;
  public test(s) { str = s; }
  public void run() {
   for(int i=0; i<100; i++)
    System.out.println(s);
  }
 }
```

Thread Synchronization

৳ 1000

ATM Booth → ৳100 (উঠানো হলো) → 900

check book → ৳100 (উঠানো হলো) → 900

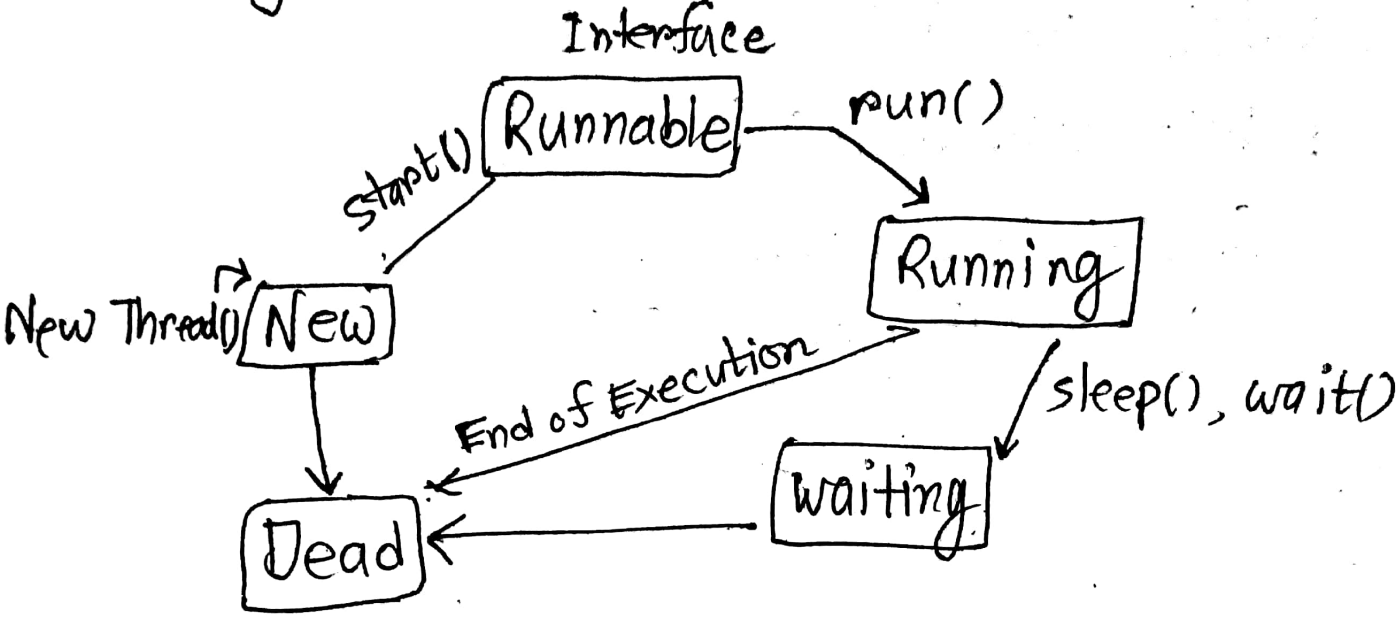
But actually ৳200 was retrieved so the balance is now ৳ ৳800.

To avoid this mismanagement Thread Synchronization is used.

Lock method (যা Boolean Output দেয়) ব্যবহার করে Thread Synchronization এর কাজ করা হয়।

SA madam | CSE 1203 | 12/D | 25.11.17

□ Life Cycle of a thread



Thread Methods :

1. public void start
   → starts the thread in a seperate path of execution., then invokes the run method on this thread object.

2. public void run
   → if this thread object was initiated using a seperate runnable target, then the run method is invoked on that Runnable.

3. public final void setPriority

→ sets the priority of this Thread object the possible values are between 1 to 10.

Thread t = new Thread (ob, priority number);

4. public void interrupt

→ interrupts the thread, causing it to continue execution if it was blocked for any reason.

5. public final boolean isAlive()

→ Returns true if the thread is alive, which is any time after the thread has been started before it runs to completion.

29.9 → Thread Synchronization

29.10 → Lock Method