https://www.hcltss-lms.com/login/index.php

Technical class practice links

python practice link

<https://www.online-python.com/>

Scratch

<https://scratch.mit.edu/>

The following gives common keywords used in pseudocodes –

1.**//:**This keyword used to represent a comment.

2. **BEGIN, END:** Begin is the first statement and end is the last statement.

3. **INPUT, GET, READ**: The keyword is used to inputting data.

4. **COMPUTE, CALCULATE**: used for calculation of the result of the given expression.

5. **ADD, SUBTRACT, INITIALIZE**used for addition, subtraction and initialization**.**

6. **OUTPUT, PRINT, DISPLAY**: It is used to display the output of the program.

7. **IF, ELSE, ENDIF**: used to make decision.

8. **WHILE, ENDWHILE**: used for iterative statements.

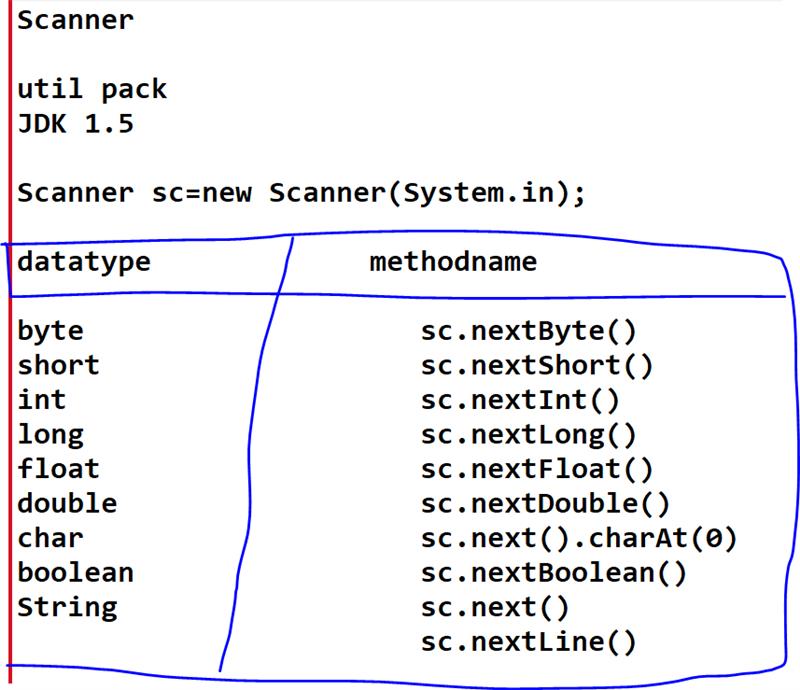
9. **FOR, ENDFOR**: Another iterative incremented/decremented tested automatically.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LP-5CG0411VPG | Bejagama chandana | VJWTB0821028 | .\itdtpadmin | VlzP+[T2OY |

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;

public class SumDemo {

public static void main(String[] args) throws Exception {  
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  
System.out.println("enter 2 int no:");  
int x=Integer.parseInt(br.readLine());  
int y=Integer.parseInt(br.readLine());  
int z=x+y;  
System.out.println("sum of 2 int no:"+z);



package hcl;  
import java.util.Scanner;

public class SumDemo {

public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter 2 int no:");  
int x=sc.nextInt();  
int y=sc.nextInt();  
int z=x+y;  
System.out.println("sum of 2 int no:"+z);  
sc.close();  
}}

cmd arg based  
---------------------  
public class singleCmdLineArg {

public static void main(String[] args) {  
String s1=args[0];  
System.out.println(s1);  
  
}}

cmd arg based  
---------------------  
public class singleCmdLineArg {

public static void main(String[] args) {  
String s1=args[0];  
System.out.println(s1);  
  
}}

package hcl;

public class TwoCmdLineArg {

public static void main(String[] args) {  
String s1=args[0];  
String s2=args[1];  
System.out.println(s1);  
System.out.println(s2);

}}

package hcl;

public class MultipleCmdLineArg {

public static void main(String[] args) {  
for(int i=0;i<args.length;i++)  
{  
System.out.println(args[i]);  
}}

package hcl;

import java.util.Scanner;

public class WideninigDemo {

public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter any char");  
char ch=sc.next().charAt(0);  
int n=ch;  
System.out.println("given char AsciiNo: "+n);  
sc.close();  
}}

package hcl;

import java.util.Scanner;

public class NarrowingDemo {

public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter any int no");  
int n=sc.nextInt();  
char ch=(char) n;  
System.out.println("give no Ascci char:"+ch);  
sc.close();  
}}

import java.util.Scanner;

public class BitWiseOpeDemo {

public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter 2 no");  
int a=sc.nextInt();  
int b=sc.nextInt();  
System.out.println(a&b);  
System.out.println(a|b);  
System.out.println(~a);  
System.out.println(a^b);  
System.out.println(a<<4);  
System.out.println(a>>2);  
System.out.println(a>>>2);  
}}

package hcl;

public class IncDecDemo {

public static void main(String[] args) {  
int a=55;  
System.out.println(a);  
System.out.println(++a);  
System.out.println(a++);  
System.out.println(a);  
System.out.println(a--);  
System.out.println(a);  
System.out.println(--a);  
}}

if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))

package hcl;  
import java.util.Scanner;

public class SumDemo {

public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter 2 int no:");  
int x=sc.nextInt();  
int y=sc.nextInt();  
System.out.println("sum of 2 int no:"+(x+y));  
System.out.println("sub of 2 int no:"+(x-y));  
System.out.println("mul of 2 int no:"+(x\*y));  
System.out.println("div of 2 int no:"+(x/y));  
System.out.println("rem of 2 int no:"+(x%y));  
  
int a=90;  
System.out.println(a);  
a\*=x;  
System.out.println(a);  
System.out.println("big="+(x>y));  
if(x>y)  
{  
System.out.println(" x is biggest");  
}  
else  
{  
System.out.println("y is biggest");  
}  
sc.close();  
}}

double t=56.9988767;  
System.out.println(t);  
DecimalFormat u=new DecimalFormat("0.00");  
System.out.println(u.format(t));

package hcl;

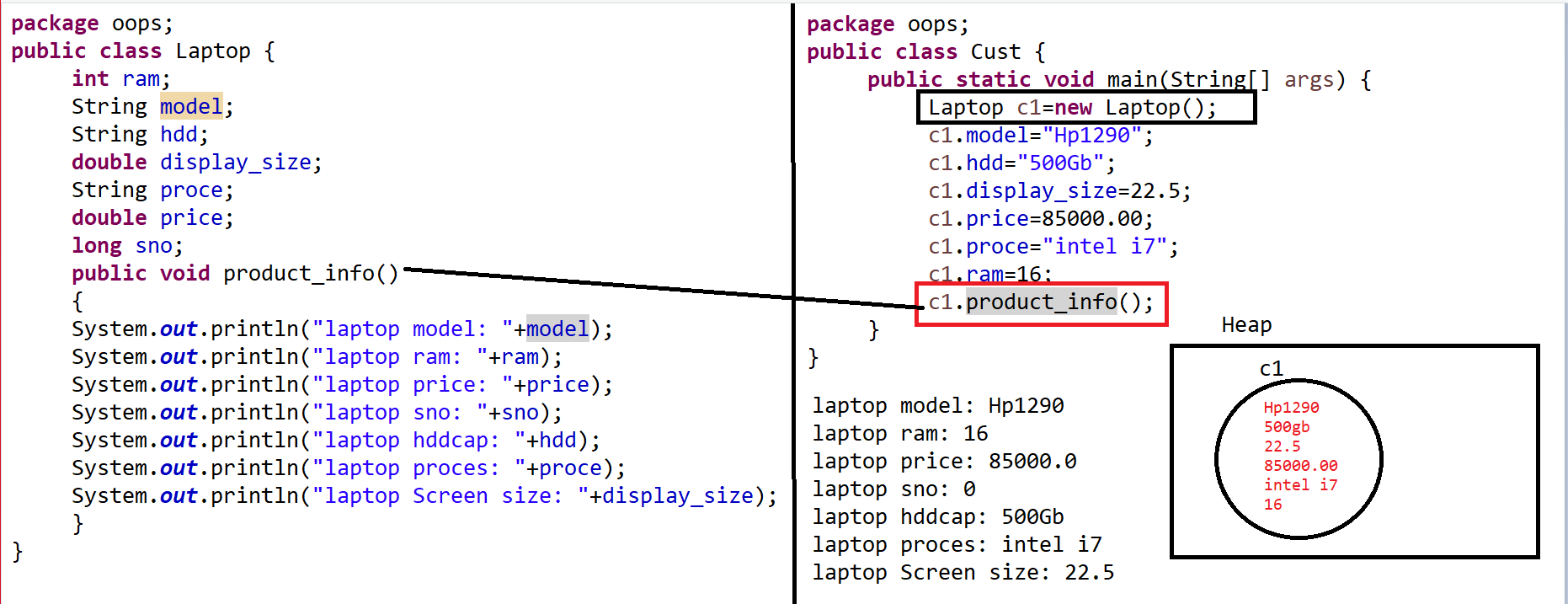
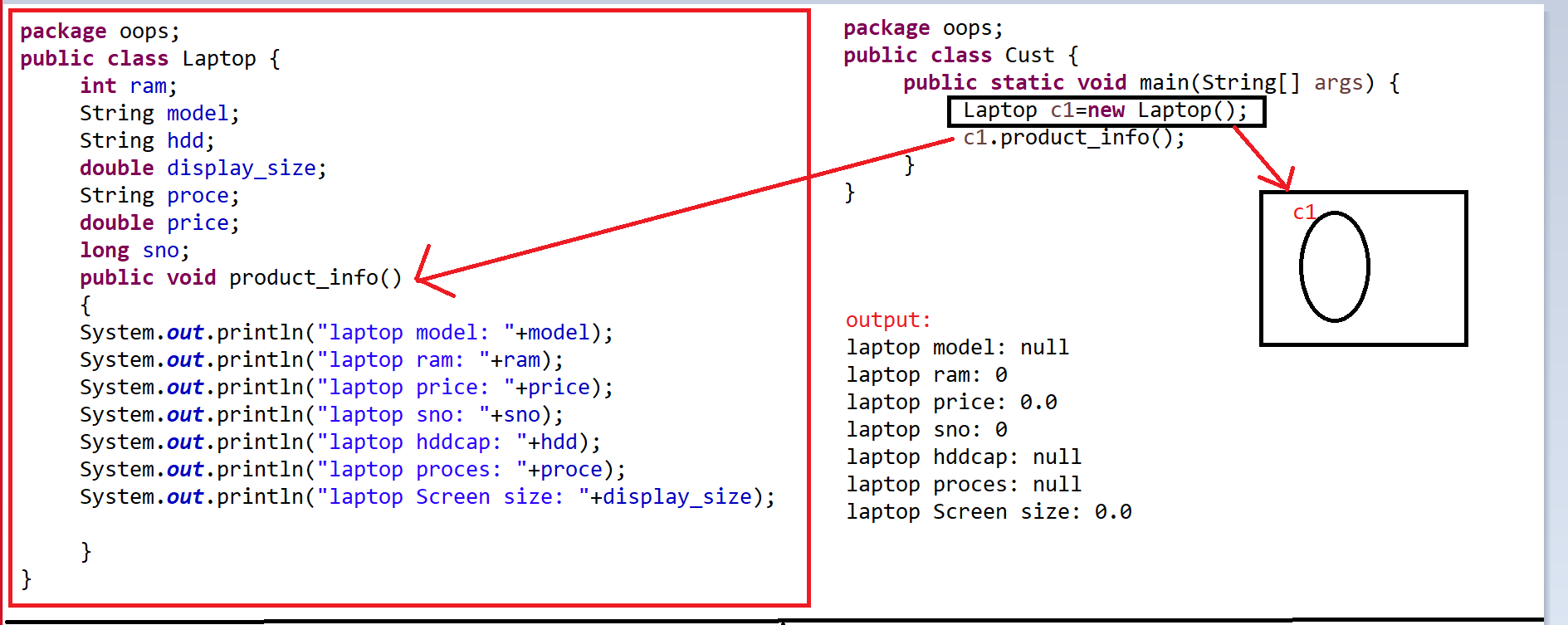
import java.util.Scanner;

public class CondStmtDemo {

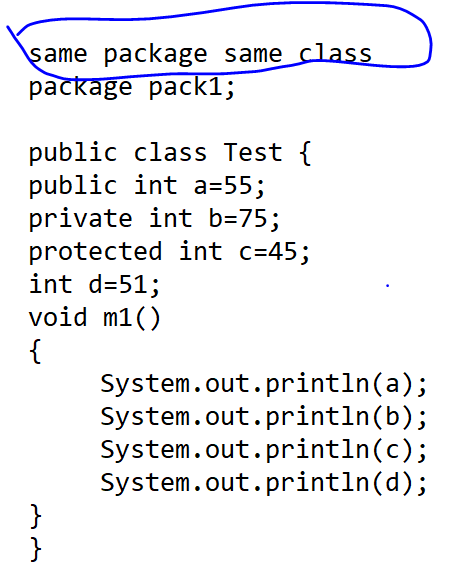
public static void main(String[] args) {  
Scanner sc=new Scanner(System.in);  
System.out.println("enter country name:");  
String cname=sc.next();  
if(cname.equalsIgnoreCase("india"))  
{  
System.out.println("conutryname is ok");  
System.out.println("enter the person age:");  
int age=sc.nextInt();  
if(age>=18)  
{  
System.out.println("person ok voteing");  
}  
else  
{  
System.out.println("age is not matching");  
}  
}  
else  
{  
System.out.println("u r ele for voter enrol");  
}

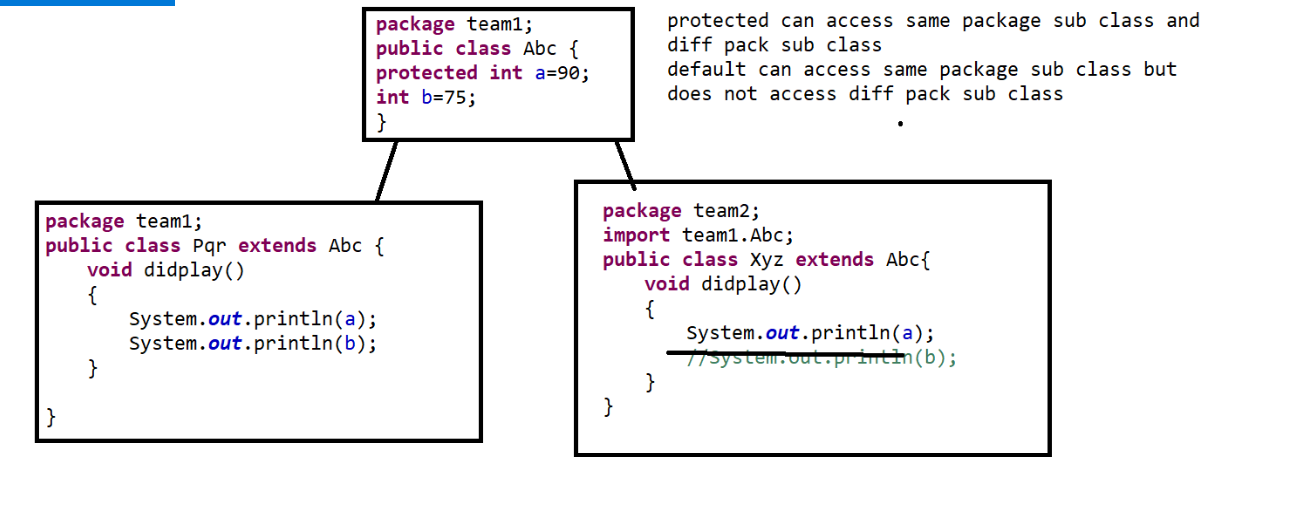
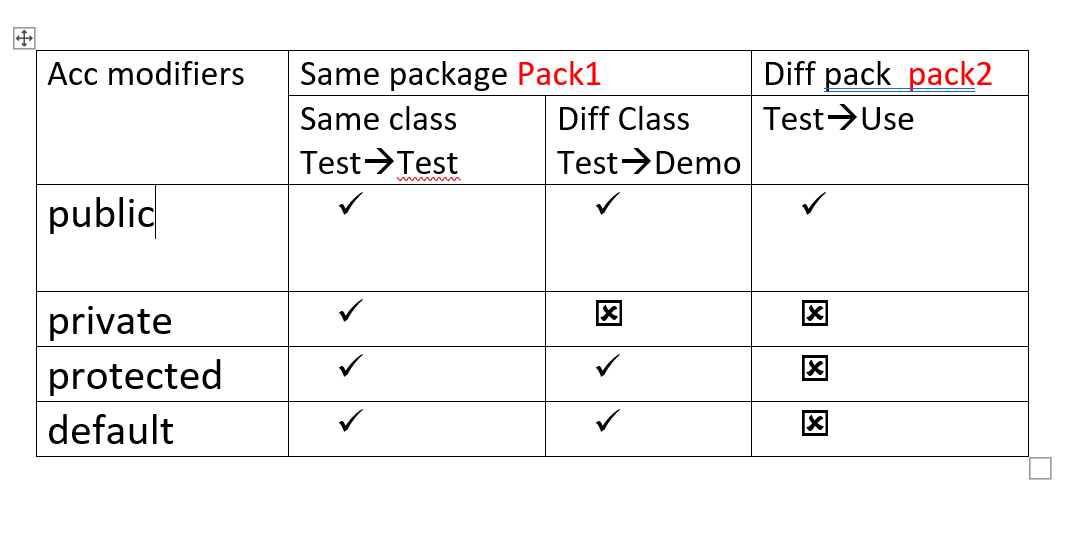
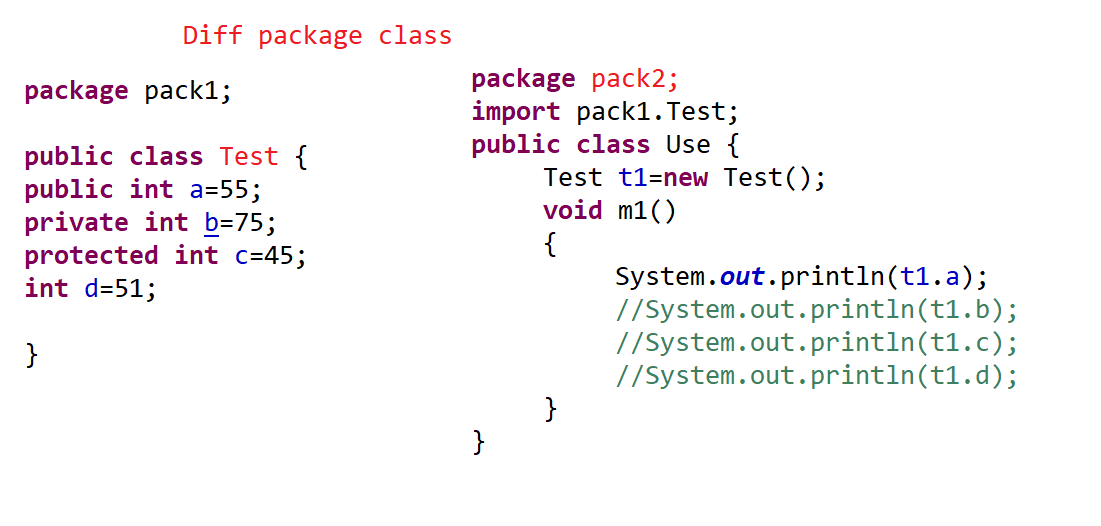
}

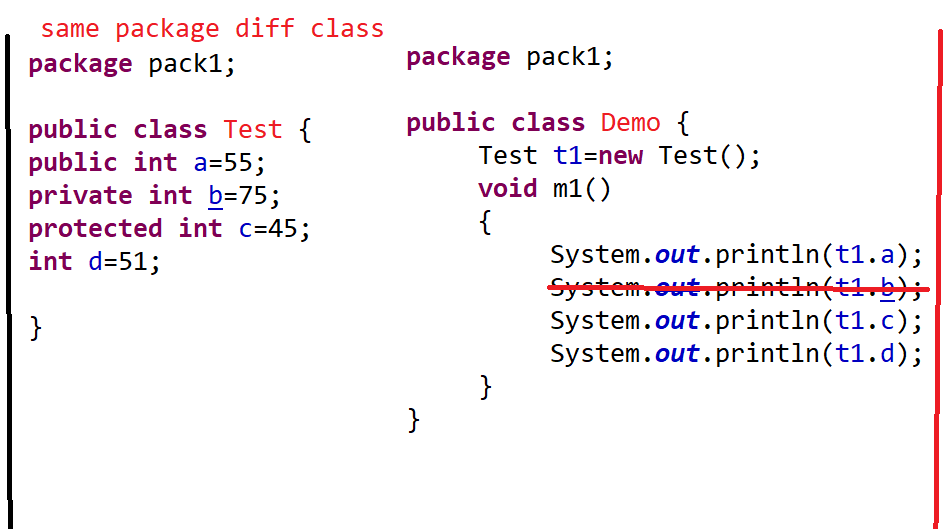
}







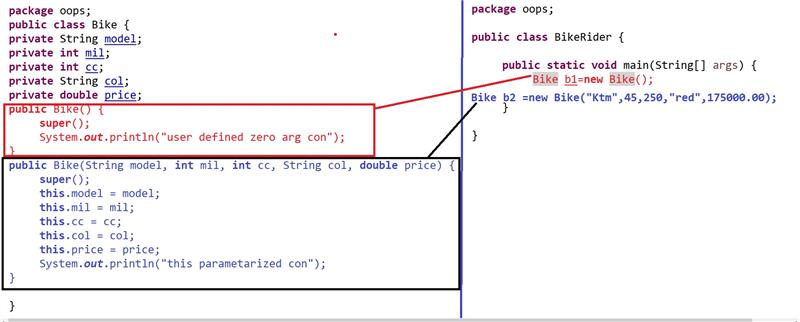
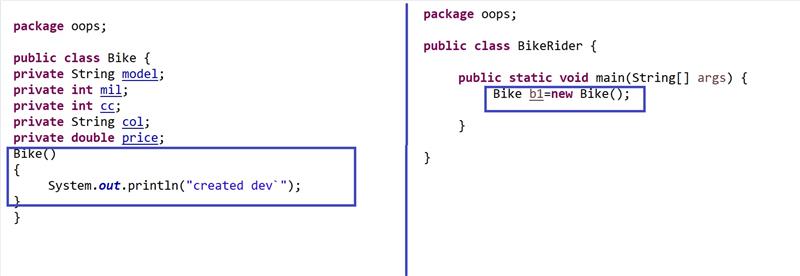




Types of Constructors

2. Zero Argument constructor

3. parameterized Constructor



package inher;  
public class Hcl {  
double fee=200000.00;  
public void exams()  
{  
System.out.println("get ready for exam");  
}  
}

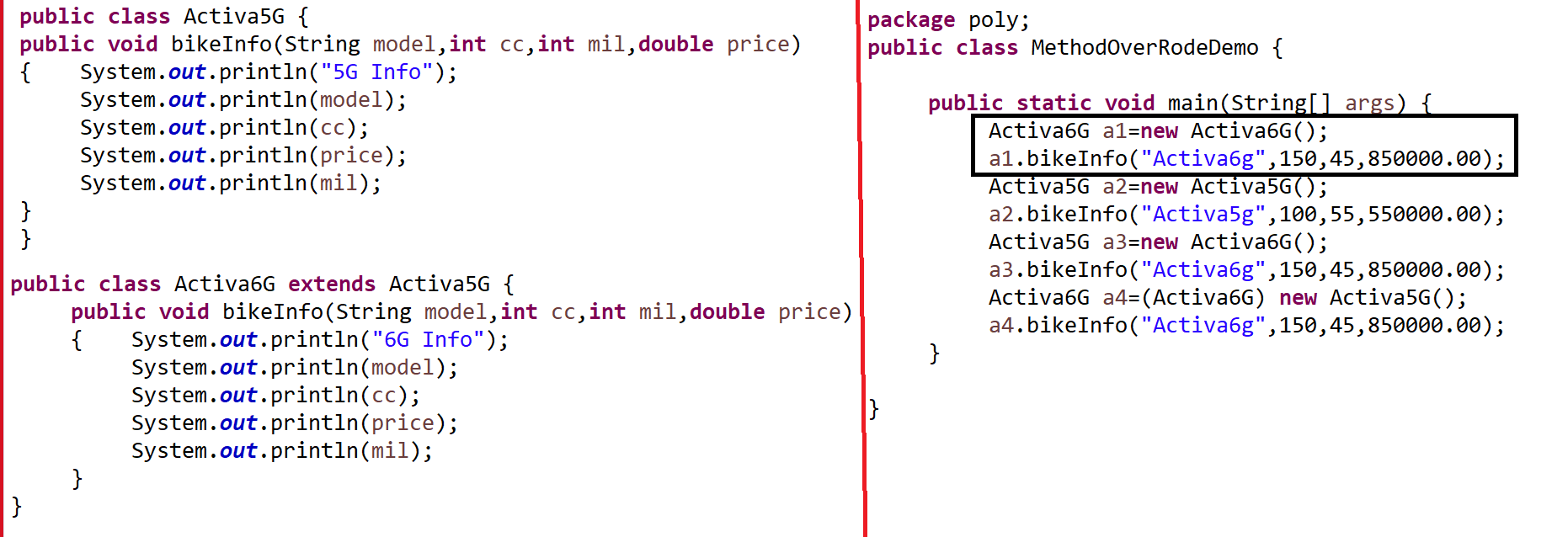
package inher;  
public class Sai extends Hcl {  
double fee=500000.00;  
void feeInfo()  
{  
System.out.println(fee);  
System.out.println(super.fee);  
this.exams();  
super.exams();  
}  
public void exams()  
{  
System.out.println("exams are over");  
}  
}  
package inher;

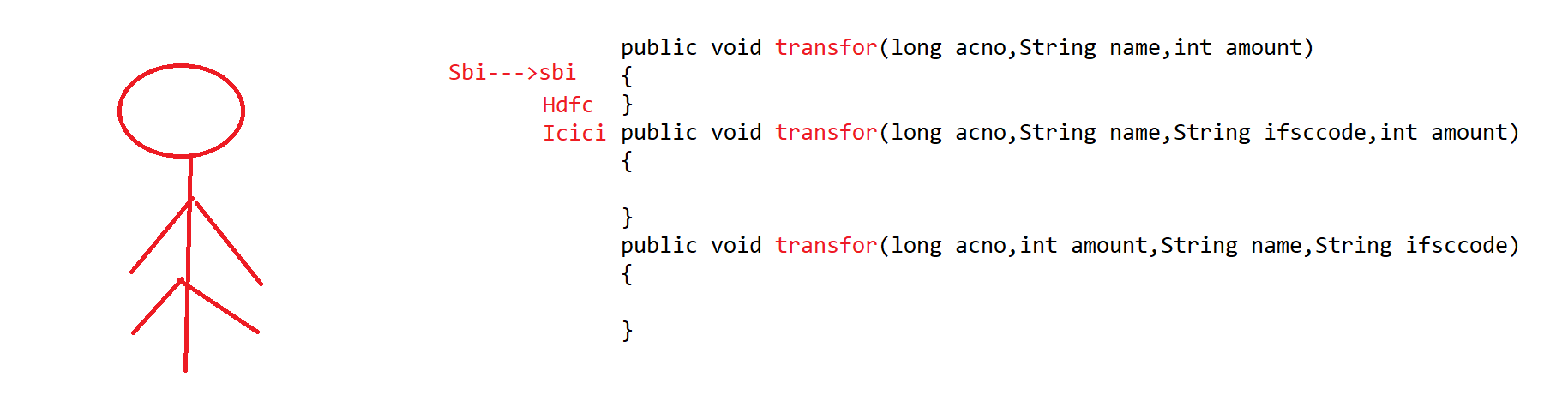
public class SuperDemo {

public static void main(String[] args) {  
Sai s=new Sai();  
s.feeInfo();

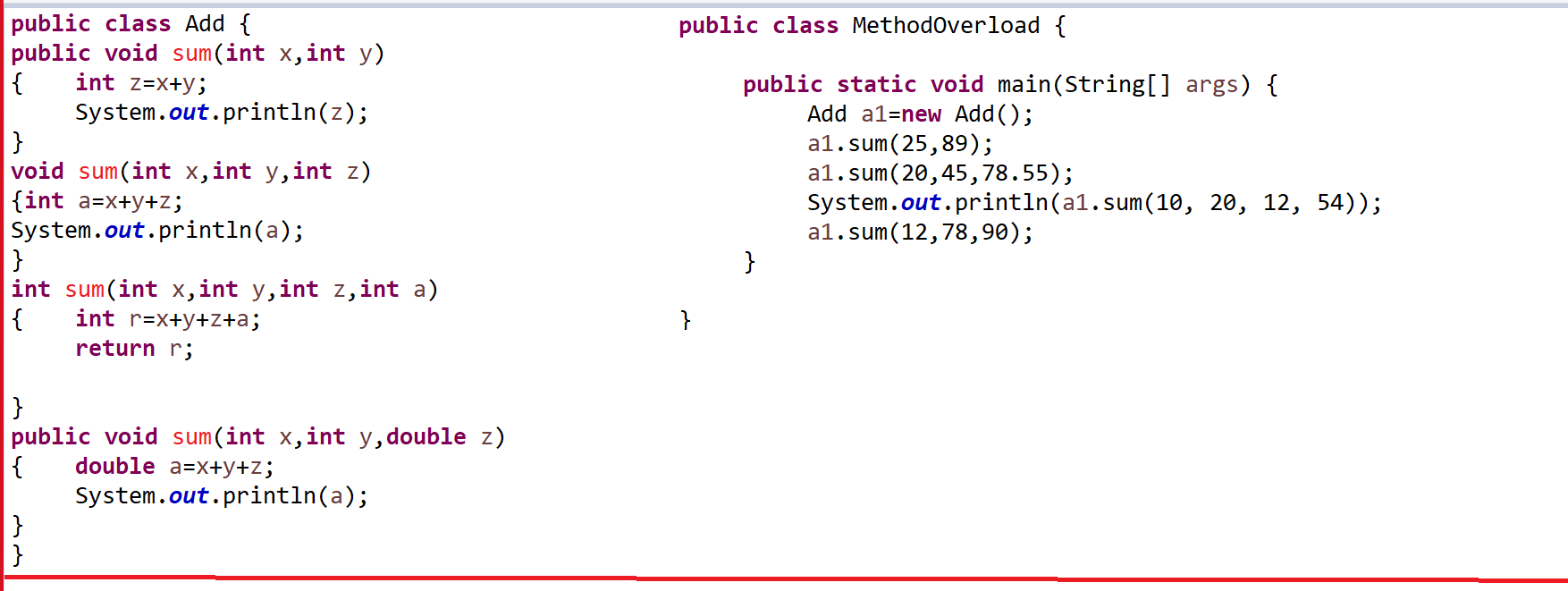
}}

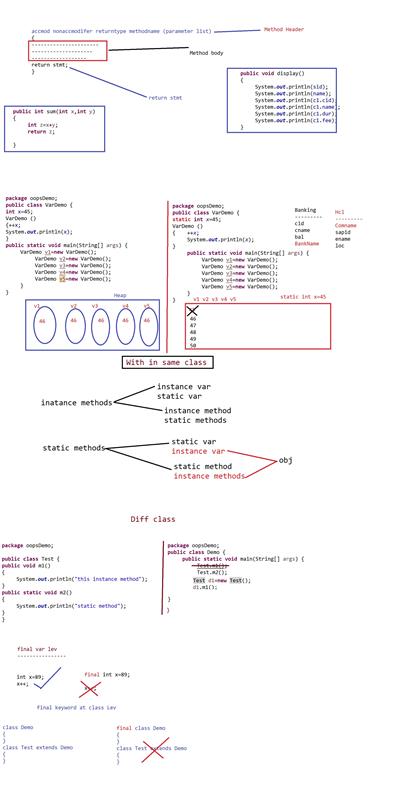
Method override

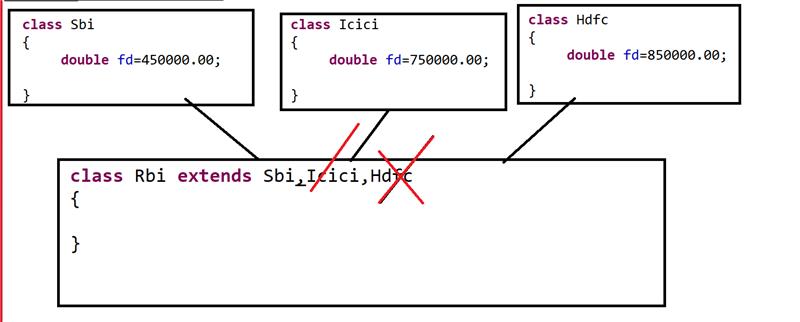




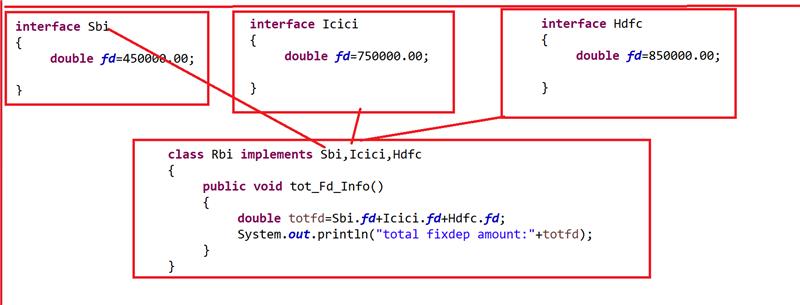
Method over load







Multiple Inheritance



package oopsDemo;

public class Course {  
int cid;  
String name;  
int dur;  
double fee;  
public Course(int cid, String name, int dur, double fee) {  
super();  
this.cid = cid;  
this.name = name;  
this.dur = dur;  
this.fee = fee;  
}

}

package oopsDemo;

public class Student {  
int sid;  
String name;  
Course c1;  
public Student(int sid, String name, Course c1) {  
super();  
this.sid = sid;  
this.name = name;  
this.c1 = c1;  
}  
public void display()  
{  
System.out.println(sid);  
System.out.println(name);  
System.out.println(c1.cid);  
System.out.println(c1.name);  
System.out.println(c1.dur);  
System.out.println(c1.fee);  
}  
}

package oopsDemo;

public class Main {

public static void main(String[] args) {  
Course c1=new Course(25,"java",60,85000.00);  
Course c2=new Course(32,"ms.net",80,65000.00);  
Student s1=new Student(123,"pavan",c1);  
Student s2=new Student(165,"adi",c2);  
Student s3=new Student(569,"naman",c1);  
s1.display();  
s2.display();  
s3.display();

package oopsDemo;

public class MethodDemo {  
int x=45;  
static int y=25;  
public void m1()  
{  
System.out.println(x);  
System.out.println(y);  
}  
public static void m2()  
{  
MethodDemo md1=new MethodDemo();  
System.out.println(md1.x);  
System.out.println(y);  
}  
public void m3()  
{  
this.m1();  
this.m2();  
}  
public static void main(String[] args) {  
MethodDemo md2=new MethodDemo();  
md2.m3();  
m2();

}

}

package oopsDemo;

public class ParametrizedMethodDemo {  
public static void interchange(int x,int y)  
{  
System.out.println("output before swaping");  
System.out.println(x);  
System.out.println(y);  
int z=x;  
x=y;  
y=z;  
System.out.println("output after swaping");  
System.out.println(x);  
System.out.println(y);  
  
}  
public static void swap(int x,int y)  
{  
System.out.println("output before swaping");  
System.out.println(x);  
System.out.println(y);  
x=x+y; //30  
y=x-y; //10  
x=x-y; // 20  
System.out.println("output after swaping");  
System.out.println(x);  
System.out.println(y);  
  
}  
public static void main(String[] args) {  
interchange(12,65);  
swap(10,20);  
}}

1.by useing exteds

i create u r own class That must b e sub class For Thread

public class MyThread extends Thread  
{}  
ii.Write u r class Logic in Run method  
public void run()  
{  
System.out.println("Thread is created");  
}

iii.create u r class object  
MyThread mt=new MyThread();

iv att that obj to Thread

Thread t=new Thread(mt);

v to rubn Thread invoke Start methods  
t.start();

package threadsdemo;

public class MyThread extends Thread {  
public void run()  
{  
System.out.println("Thread is created");  
}  
public static void main(String args[])  
{  
MyThread mt=new MyThread();  
Thread t=new Thread(mt);  
t.start();  
//t.run();  
}  
}

-----------------------------------

package threadsdemo;

public class MyThread extends Thread {  
public void run()  
{  
System.out.println("Thread is created");  
}  
public static void main(String args[])  
{  
MyThread mt=new MyThread();  
mt.start();  
//t.run();  
}  
}  
--------------------------------------------  
1.by useing implements Runnable

i create u r own class That must be implementation class Runnable interface

public class OurThread implements Runnable {}  
ii.Write u r class Logic in Run method  
public void run()  
{  
System.out.println("Thread is created");  
}

iii.create u r class object  
MyThread mt=new MyThread();

iv att that obj to Thread

Thread t=new Thread(mt);

package threadsdemo;

public class OurThread extends Test implements Runnable {

@Override  
public void run() {  
System.out.println("Thread is created");

}  
public static void main(String args[])  
{  
OurThread ot=new OurThread();  
Thread t=new Thread(ot);  
t.start();  
}  
}

with out MT  
package threadsdemo;

public class Atm {  
private int cid;  
private String cname,loc;  
public Atm() {  
super();  
// TODO Auto-generated constructor stub  
}  
public Atm(int cid, String cname, String loc) {  
super();  
this.cid = cid;  
this.cname = cname;  
this.loc = loc;  
}  
public int getCid() {  
return cid;  
}  
public void setCid(int cid) {  
this.cid = cid;  
}  
public String getCname() {  
return cname;  
}  
public void setCname(String cname) {  
this.cname = cname;  
}  
public String getLoc() {  
return loc;  
}  
public void setLoc(String loc) {  
this.loc = loc;  
}  
  
public void run()  
{  
for(int i=1;i<=10;i++)  
{  
System.out.println(cid+"-->"+cname+"-->"+loc +"-->"+i);  
}  
}  
}  
-------------------  
package threadsdemo;

public class MutiThreadingDemo {

public static void main(String[] args) {  
Atm c1=new Atm(1234,"Hrusikesh Sahoo","Odisha");  
Atm c2=new Atm(4205,"AASHISH SHAKYA","DELHI");  
Atm c3=new Atm(3489,"Balakrishna ","Hyd");  
c1.run();  
c2.run();  
c3.run();  
}

}

1234-->Hrusikesh Sahoo-->Odisha-->1  
1234-->Hrusikesh Sahoo-->Odisha-->2  
1234-->Hrusikesh Sahoo-->Odisha-->3  
1234-->Hrusikesh Sahoo-->Odisha-->4  
1234-->Hrusikesh Sahoo-->Odisha-->5  
1234-->Hrusikesh Sahoo-->Odisha-->6  
1234-->Hrusikesh Sahoo-->Odisha-->7  
1234-->Hrusikesh Sahoo-->Odisha-->8  
1234-->Hrusikesh Sahoo-->Odisha-->9  
1234-->Hrusikesh Sahoo-->Odisha-->10  
4205-->AASHISH SHAKYA-->DELHI-->1  
4205-->AASHISH SHAKYA-->DELHI-->2  
4205-->AASHISH SHAKYA-->DELHI-->3  
4205-->AASHISH SHAKYA-->DELHI-->4  
4205-->AASHISH SHAKYA-->DELHI-->5  
4205-->AASHISH SHAKYA-->DELHI-->6  
4205-->AASHISH SHAKYA-->DELHI-->7  
4205-->AASHISH SHAKYA-->DELHI-->8  
4205-->AASHISH SHAKYA-->DELHI-->9  
4205-->AASHISH SHAKYA-->DELHI-->10  
3489-->Balakrishna -->Hyd-->1  
3489-->Balakrishna -->Hyd-->2  
3489-->Balakrishna -->Hyd-->3  
3489-->Balakrishna -->Hyd-->4  
3489-->Balakrishna -->Hyd-->5  
3489-->Balakrishna -->Hyd-->6  
3489-->Balakrishna -->Hyd-->7  
3489-->Balakrishna -->Hyd-->8  
3489-->Balakrishna -->Hyd-->9  
3489-->Balakrishna -->Hyd-->10  
--------------------------------------

with mutiThreading  
------------------------  
package threadsdemo;

public class Atm extends Thread{  
private int cid;  
private String cname,loc;  
public Atm() {  
super();  
// TODO Auto-generated constructor stub  
}  
public Atm(int cid, String cname, String loc) {  
super();  
this.cid = cid;  
this.cname = cname;  
this.loc = loc;  
}  
public int getCid() {  
return cid;  
}  
public void setCid(int cid) {  
this.cid = cid;  
}  
public String getCname() {  
return cname;  
}  
public void setCname(String cname) {  
this.cname = cname;  
}  
public String getLoc() {  
return loc;  
}  
public void setLoc(String loc) {  
this.loc = loc;  
}  
  
public void run()  
{  
for(int i=1;i<=10;i++)  
{  
System.out.println(cid+"-->"+cname+"-->"+loc +"-->"+i);  
}  
}  
}

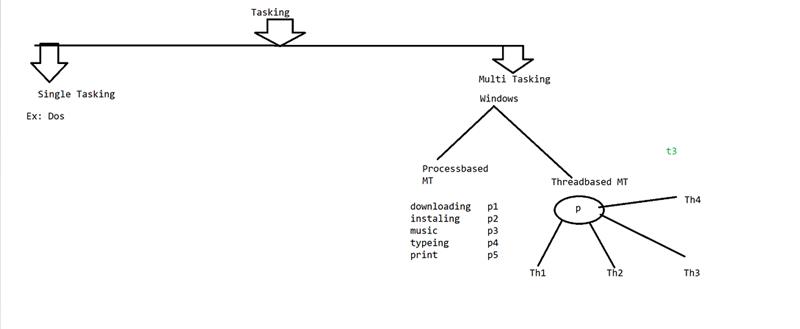
----------------------------  
package threadsdemo;

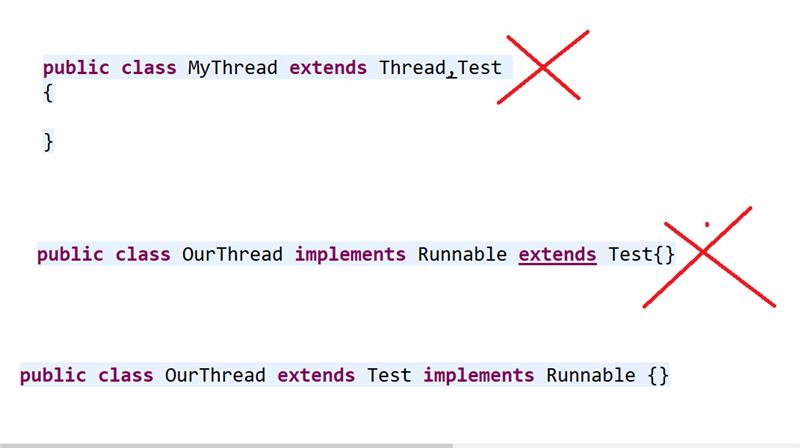
public class MutiThreadingDemo {

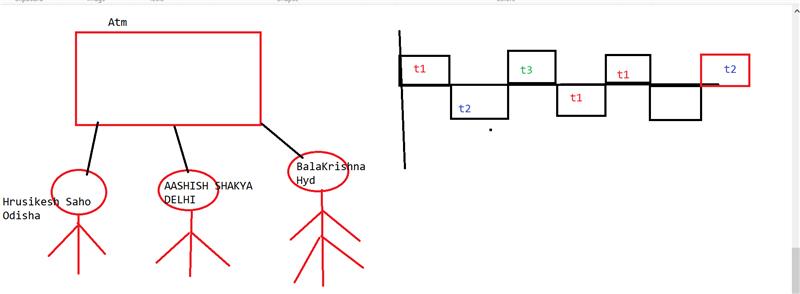
public static void main(String[] args) {  
Atm c1=new Atm(1234,"Hrusikesh Sahoo","Odisha");  
Atm c2=new Atm(4205,"AASHISH SHAKYA","DELHI");  
Atm c3=new Atm(3489,"Balakrishna ","Hyd");  
c1.start();  
c2.start();  
c3.start();  
}}

output:

4205-->AASHISH SHAKYA-->DELHI-->1  
4205-->AASHISH SHAKYA-->DELHI-->2  
4205-->AASHISH SHAKYA-->DELHI-->3  
4205-->AASHISH SHAKYA-->DELHI-->4  
3489-->Balakrishna -->Hyd-->1  
3489-->Balakrishna -->Hyd-->2  
1234-->Hrusikesh Sahoo-->Odisha-->1  
3489-->Balakrishna -->Hyd-->3  
3489-->Balakrishna -->Hyd-->4  
3489-->Balakrishna -->Hyd-->5  
3489-->Balakrishna -->Hyd-->6  
4205-->AASHISH SHAKYA-->DELHI-->5  
3489-->Balakrishna -->Hyd-->7  
1234-->Hrusikesh Sahoo-->Odisha-->2  
3489-->Balakrishna -->Hyd-->8  
4205-->AASHISH SHAKYA-->DELHI-->6  
3489-->Balakrishna -->Hyd-->9  
1234-->Hrusikesh Sahoo-->Odisha-->3  
3489-->Balakrishna -->Hyd-->10  
1234-->Hrusikesh Sahoo-->Odisha-->4  
4205-->AASHISH SHAKYA-->DELHI-->7  
1234-->Hrusikesh Sahoo-->Odisha-->5  
4205-->AASHISH SHAKYA-->DELHI-->8  
1234-->Hrusikesh Sahoo-->Odisha-->6  
4205-->AASHISH SHAKYA-->DELHI-->9  
1234-->Hrusikesh Sahoo-->Odisha-->7  
1234-->Hrusikesh Sahoo-->Odisha-->8  
1234-->Hrusikesh Sahoo-->Odisha-->9  
1234-->Hrusikesh Sahoo-->Odisha-->10  
4205-->AASHISH SHAKYA-->DELHI-->10



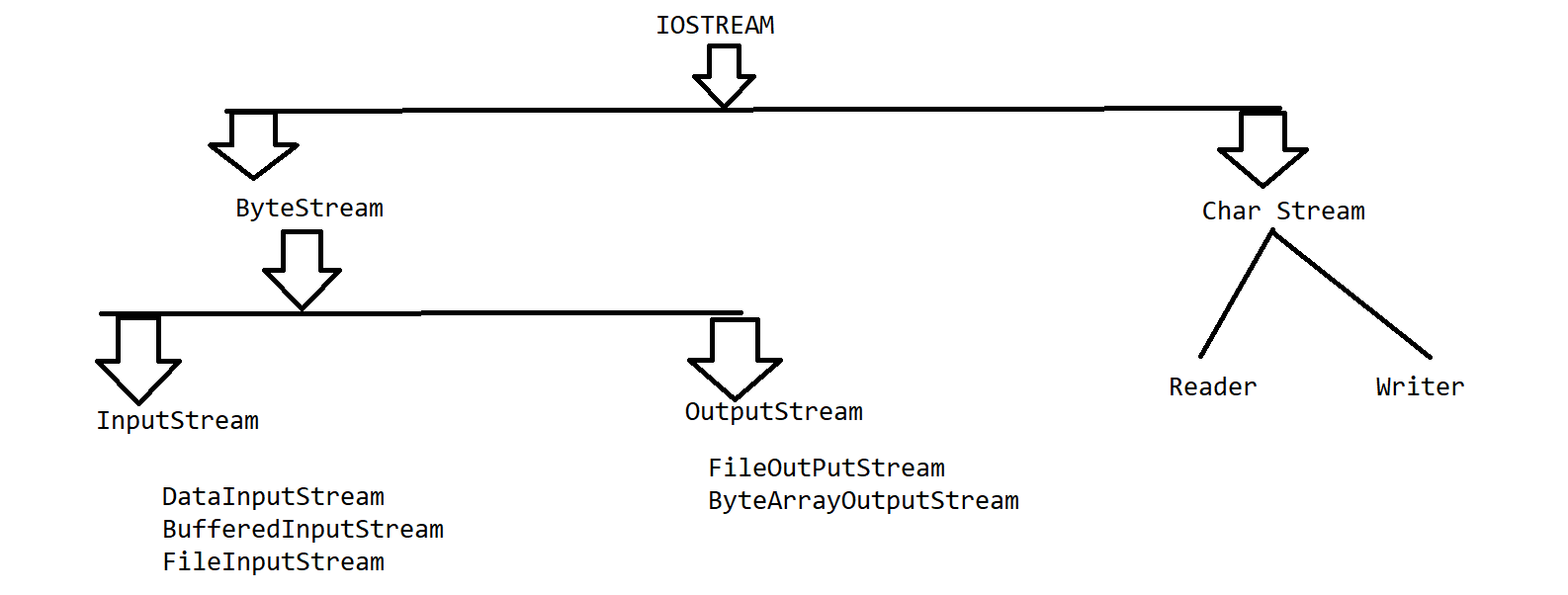




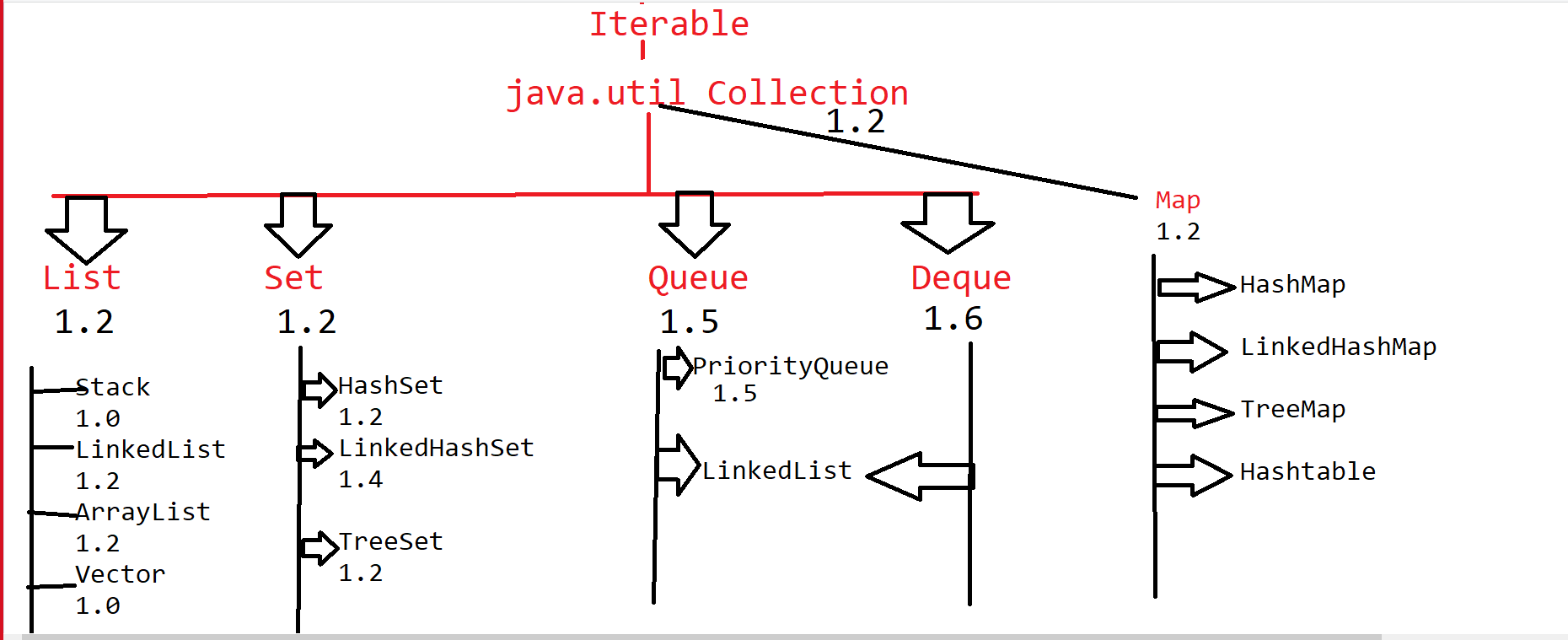
package threadsdemo;

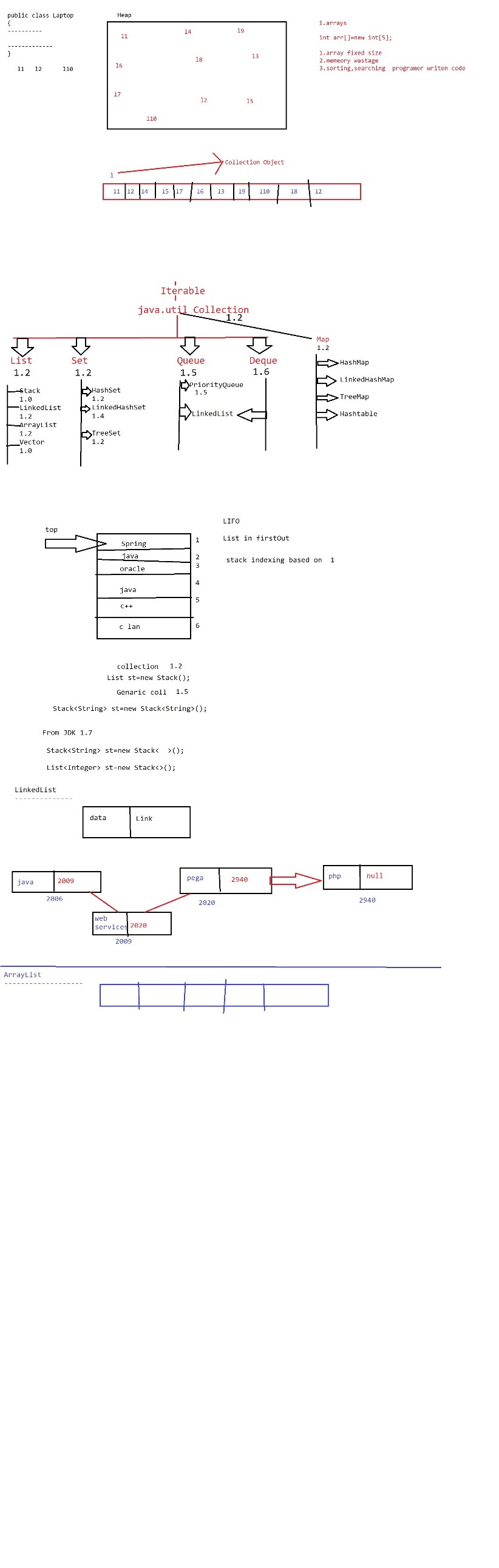
public class MyThread extends Thread {  
public void run()  
{  
System.out.println(Thread.currentThread());  
System.out.println(Thread.currentThread().getId());  
System.out.println(Thread.currentThread().getName());  
System.out.println(Thread.currentThread().getPriority());  
System.out.println(Thread.currentThread().getState());

}  
public static void main(String args[])  
{  
MyThread mt=new MyThread();  
Thread t1=new Thread(mt);  
Thread t2=new Thread(mt);  
System.out.println(t1.getState());  
t1.setName("java");  
t1.setPriority(2);  
t1.setDaemon(true);  
t1.start();  
t2.start();  
//t.run();  
}  
}



COLLECTIONS





package coll;

import java.util.Stack;

public class StackDemo {

public static void main(String[] args) {  
Stack<Integer> st=new Stack<>();  
System.out.println(st.empty());  
//st.pop();  
st.push(18); //autoboxing  
st.push(19);  
st.push(20);  
st.push(18);  
st.push(22);  
System.out.println("top="+st.peek());  
System.out.println(st);  
st.pop();  
System.out.println(st);  
System.out.println(st.search(19));  
System.out.println("top="+st.peek());  
}}

package coll;

public class Laptop {  
private int ram;  
private String model;  
private double price;  
public Laptop() {  
super();  
// TODO Auto-generated constructor stub  
}  
public Laptop(int ram, String model, double price) {  
super();  
this.ram = ram;  
this.model = model;  
this.price = price;  
}  
public int getRam() {  
return ram;  
}  
public void setRam(int ram) {  
this.ram = ram;  
}  
public String getModel() {  
return model;  
}  
public void setModel(String model) {  
this.model = model;  
}  
public double getPrice() {  
return price;  
}  
public void setPrice(double price) {  
this.price = price;  
}  
@Override  
public String toString() {  
return "Laptop [ram=" + ram + ", model=" + model + ", price=" + price + "]\n";  
}  
}

package coll;

import java.util.LinkedList;

public class LinkedListDemo {

public static void main(String[] args) {  
LinkedList<Laptop> ll=new LinkedList<>();  
ll.add(new Laptop(4,"hp5409",85000.50));  
ll.add(new Laptop(2,"Dell",25000.50));  
ll.add(new Laptop(8,"hp5409",95000.50));  
ll.add(new Laptop(16,"hp5409",125000.50));  
System.out.println(ll);  
ll.add(1, new Laptop(1,"lenv",5000.50));  
Laptop l6=new Laptop(2,"lenv",8000.50);  
Laptop l9=new Laptop(20,"lenv",145000.50);  
ll.addFirst(l6);  
ll.addLast(l9);  
System.out.println(ll);  
}

}

package coll;

import java.util.ArrayList;  
import java.util.Scanner;

public class ArrayListDemo {

public static void main(String[] args) {

ArrayList<String> al=new ArrayList<>();  
int i=1;  
boolean result;  
Scanner sc=new Scanner(System.in);  
do  
{  
System.out.println("enter person name:"+i++);  
String pname=sc.nextLine();  
al.add(pname);  
System.out.println("do u wanted to insert more data enter yes or no");  
String ans=sc.nextLine();  
result =ans.equals("yes");  
  
}while(result);  
System.out.println("enterd person info");  
for(String s1:al) //unboxing  
{  
System.out.println(s1);  
}  
}

}

[3:27 PM] Balakrishna Palla

**Finally**

One of the technical requirements that you might often expect is to "execute a set of lines" irrespective of the flow that goes through try/catch. A simple scenario that you would know is to close a database connection/release resources if it goes through successfully in a try block and happens to take an exception route.

Write a program to track transactions by using the Finally keyword.  
  
Consider a class **Transaction**with private member variables.

|  |  |
| --- | --- |
| **DataType** | **Variable Name** |
| String | accountNumber |
| Double | amount |

Include apporapiate **getter** and **setters** for above class.  
Include appropriate **default**and **parameterized constructors** for the above class.  
  
Includewith the following method in **Transaction**

|  |  |
| --- | --- |
| **Return Type** | **method name** |
| Boolean | validate(transactionAmount) |

In **validate** method if the transaction amount is greater than current balance or if the current balance is in minimal balance (500) then throw a manual exception then display "Insufficient Balance". Otherwise return true. Use finally to display the available balance after transaction completed.

**Sample Input and Output :**

Enter the transaction details

Enter the account number

**123456**

Enter the available amount

**5000**

Enter the transaction amount

**500**

Do you want to enter more ?(yes/no)

**yes**

Enter the transaction amount

**1000**

Do you want to enter more ?(yes/no)

**yes**

Enter the transaction amount

**2000**

Do you want to enter more ?(yes/no)

**yes**

Enter the transaction amount

**800**

Do you want to enter more ?(yes/no)

**yes**

Enter the transaction amount

**850**

Insufficient Balance

Your available balance 700.0

[3:28 PM] Balakrishna Palla

**Inheritance - Flight**

Write a program to implement single inheritance for the given scenario.

**[Note: Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]**

Create a class **Flight** with the following protected variables:

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | clientName |
| String | idProof |
| Date | flightschedule |

Include appropriate getters and setters method in the Flight class.

Define a constructor  with parameters passed in the same order as declared in the class.

Here idProof may be PAN Card, Voter's ID, Driving License, Aadhar Card

Create a class **InternationalFlight** (which should inherit **Flight**) with the following private variables:

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | passportNumber |
| String | natureOfVisit |

Include getters and setters method in the InternationalFlight class.  
Define a constructor by invoking the base class constructor with parameters passed in the same order as declared in the class.

Include the following methods in the **InternationalFlight**class

|  |  |
| --- | --- |
| **Method** | **Description** |
| void displayFlightDetails() | This method should display all the details of the flight like clientName, idProof, flightschedule, passportNumber, natureOfVisit. |

Consider the driver class **Main**. In the main( ) method, read inputs, create a reference to the parent class and assign the child instance to it and call the appropriate method in it.

**Note:**

The format for parsing date should be **“dd/MM/yyyy”**

The link to download the template code is given below  
[Code Template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/5881?op=download&langid=6)

**[All text in bold corresponds to input and the rest corresponds to output.]**

**Sample input and output:**

Enter the client name  
**Irene**  
Enter the id proof  
**PAN Card**  
Enter the flight schedule  
**01/01/2017**  
Enter the passport number  
**AU7634905**  
Enter the nature of visit  
**Higher Education**  
Flight Details :  
Name : Irene  
ID Proof : PAN Card  
Flight Schedule : 01/01/2017  
Passport Number : AU7634905  
Flight Schedule : Higher Education

Login User

[3:30 PM] Balakrishna Palla

**Abstract Problem 1**

Let us consider the concept of cricket for our  Abstract  Problem. Consider the climax of the cricket match is going on (i.e second innings of the match) and we need to calculate the require run rate for the team to win.  
  
If we consider cricket match there are 3 international formats Test ,ODI and T20 each has 90overs per day, 50overs and 20overs respectively. Each over has 6 balls.In our scenario we will consider as its 5th day of the Test match, so the team need to achive the particular runs in 90overs.  
  
Create a abstract class **Match** with 3 private data member varibales -**currentScore** of type int, **currentOver** of type float, **target** of type int and methods - **display()** and an abstract method **calculateRunrate()**of return type as float and **calculateBalls()** of return type as int.  
  
Create the class **ODI** which extends the class **Match** .  
  
Create the class **T20** which extends the class **Match** .  
  
Create the class **Test** which extends the class **Match** .  
  
Use Appropriate Getters Setters for above classes.  
  
Create a driver class named Main which creates an instance of the above mentioned classes. Require run rate and remaining balls must be calculated seperately for all the base class  (value must be round to 2 decimal place).  
  
Require run rate is calculated as runs required/ remaining balls.  
  
**Sample Input and Output 1:**  
**[All text in bold corresponds to input and the rest corresponds to output.]**  
Enter the Cricket Format  
1.ODI  
2.T20  
3.Test  
**2**  
Enter the Current Score  
**195**  
Enter the Current over  
**19.5**  
Enter the Target  
**196**  
Requirements:  
Need 1 Run in 1 ball  
Require Run Rate - 6.00  
  
**Sample Input and Output 2:**  
  
Enter the Cricket Format  
1.ODI  
2.T20  
3.Test  
**1**  
Enter the Current Score  
**156**  
Enter the Current over  
**45.1**  
Enter the Target  
**185**  
Requirements:  
Need 29 Runs in 29 balls  
Require Run Rate - 6.00  
  
**Sample Input and Output 3:**  
Enter the Cricket Format  
1.ODI  
2.T20  
3.Test  
**5**  
Invalid Format type

[3:31 PM] Balakrishna Palla

**Discounts - Inheritance**

One of the easier ways to identify the scenarios that reflect inheritance is to look for a "is-a" relationship in the requirements document. On trying to check if we have such hierarchies, we find that there are different types of customers/account holders in the Bank. Customers can be Normal, Priviledged, SeniorCitizen and so on.   
The Bank also introduces an offer where privileged customers get a 30% off on the bill while senior citizens get 12% off.  
  
Lets implement the inheritance for the given scenario yet again for a better understanding.  
  
1. Create Customer, Privileged & SeniorCitizen class with data members as given below.  
2. Implement generateBillAmount Method as per the specification.

Create a class **Customer** with the following private data members

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | name |
| String | address |
| Integer | age |
| String | mobileNumber |

Methods in class **Customer**

|  |  |
| --- | --- |
| **Method Name** | **Method description** |
| displayCustomer() | To display the details of the customer. |

Use Appropriate **Getters & Setters**for **Customer**class.

Create a class **SeniorCitizenCustomer** which extends the class **Customer**.  
Methods in class **SeniorCitizenCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Method description** | **Return Type** |
| generateBillAmount(amount) | To calculate the payment amount where the discount is 12% . | Double |

Create a class **PrivilegeCustomer** which extends the class **Customer**.  
Methods in class **PrivilegeCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Method description** | **Return Type** |
| generateBillAmount(amount) | To calculate the payment amount where the discount is 30% . | Double |

Create a driver class named **Main** which creates an instance of the above mentioned classes.  
Use setters to set the values to objects and display all details using getters from the main method.

**Note :**

**Strictly adhere to the object oriented specifications given as part of the problem statement.**

**Use the same class names and member variable names.**

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**[All text in bold corresponds to input and the rest corresponds to output.]  
Sample Input and Output 1:**

1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
**1**  
Enter The Name  
**Ram**  
Enter The Age  
**25**  
Enter The Address  
**CBE**  
Enter The Mobile Number  
**9576531641**  
Enter The Purchased Amount  
**5000**  
Bill Details  
Name Ram  
Mobile 9576531641  
Age 25  
Address CBE  
Your bill amount is Rs 5000.0. Your bill amount is discount under privilege customer  
You have to pay Rs 3500.00

**Sample Input and Output 2:**

1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
**3**  
Invalid Customer Type