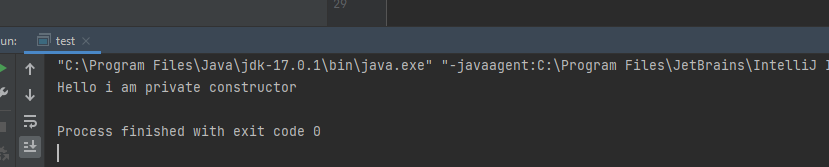
**Assignment no. 2 (OOPs)**

1)Write a singletone class . confirm that singletone class can not be inherited .

class Student {  
  
 static Student *st*= new Student();  
  
 private Student()  
 {  
 System.*out*.println("Hello i am private constructor ");  
 }  
  
 static Student getInstance() {  
 return *st*;  
 }  
  
}  
  
class test  
{  
 public static void main(String[] args)  
 {  
 Student st1 =Student.*getInstance*();  
 Student st2 =Student.*getInstance*();  
  
 }  
}



**All you need to extend a singleton class is a constructor with protected or package-default in the singleton class. If there are only private constructors you simply won't be able to extend it. If there are public constructors then it's not a singleton class.**

2. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.

**package** neww;

**public** **class** Employee {

String name;

**int** salary;

Employee(){

name=**null**;

salary=0;

}

Employee(String name , **int** salary){

**this**.name=name;

**this**.salary=salary;

}

**int** getSalary()

{

**return** salary;

}

}

**class** Manager **extends** Employee{

**int** incentive;

Manager(){

**super**();

incentive=0;

}

Manager(String n , **int** sal , **int** i)

{

**super**(n, sal);

**this**.incentive=i;

}

**int** getSalary() {

**return**(**super**.getSalary()+incentive);

}

}

**class** Labour **extends** Employee{

**int** overTime;

Labour(){

**super**();

overTime=0;

}

Labour(String n , **int** sal , **int** ot){

**super**(n , sal);

**this**.overTime=ot;

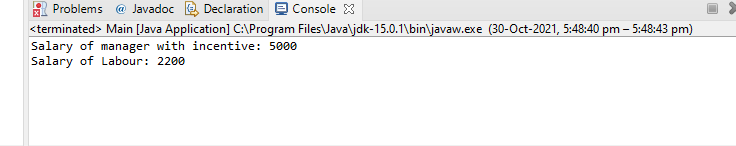
}

**int** getSalary() {

**return**(**super**.getSalary()+overTime);

}

}



3. Write a program to consider saving & current account in the bank. Saving account holder has 'Fixed Deposits' whereas Current account holder has cash credit. Apply polymorphism to find out total cash in the bank.

**package** bank;

**public** **class** Saving {

String CustomerName;

**int** totalCash;

String accType;

Saving(){

}

Saving(String name , **int** cash){

**this**.CustomerName=name;

**this**.totalCash=cash;

}

**void** AccountType(String accType) {

System.***out***.println("Fixed dipsite");

}

**int** gettotalCash()

{

**return** totalCash;

}

}

**package** bank;

**public** **class** Current **extends** Saving {

String CustomerName;

**int** totalCash;

String accType;

Current(String name, **int** cash)

{

**super**(name,cash);

}

**void** AccountType(String accType) {

System.***out***.println("Cash Credit");

}

**int** gettotalCash()

{

**return** totalCash;

}

}

**package** bank;

**public** **class** mainbank {

**public** **static** **void** main(String[] args) {

Saving s1=**new** Saving("rakesh",5000);

s1.AccountType("fixed");

System.***out***.println("Saving account Total cash : " + s1.gettotalCash());

Current c1 =**new** Current("Ramesh",7000);

c1.AccountType("cash");

System.***out***.println("Current account Total cash: "+ c1.gettotalCash());

}

}

4. Test the following principles of an abstract class:

1. If any class has any of its method abstract then you must declare entire class abstract.



Abstract class cannot be instantiated.

When we extend an abstract class, we must either override all the abstract methods in sub

class or declare subclass as abstract.

Abstract class cannot be private.

Abstract class cannot be final.

You can declare a class abstract without having any abstract method.

5. Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.

**package** Shappe;

**public** **abstract** **class** Shape {

String name;

**public** Shape() {

**this**.name = **null**;

}

**public** Shape(String name) {

**this**.name = name;

}

**public** **abstract** **void** draw();

}

**public** **class** Rectangle **extends** Shape{

@Override

**public** **void** draw() {

System.***out***.println("Rectangle drawn ");

}

}

**public** **class** Line **extends** Shape {

@Override

**public** **void** draw() {

System.***out***.println("Line drawn ");

}

}

**public** **class** Cube **extends** Shape {

@Override

**public** **void** draw() {

System.***out***.println("Cube drawn ");

}

}

**public** **class** mainShape {

**public** **static** **void** main(String[] args) {

Rectangle p=**new** Rectangle();

p.draw();

Line m=**new** Line();

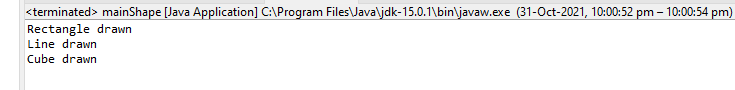
m.draw();

Cube k=**new** Cube();

k.draw();

}

}



6. Write an abstract class 'Persistence' along with two sub classes 'FilePersistence' &'DatabasePersistence'. The base class with have an abstract method persist() which will be overridden by its sub classes. Write a client who gets the Persistence object at runtime & invokes persist() method on it without knowing whether data is being saved in File or in Database.

**package** file;

**public** **abstract** **class** Persistance {

**public** **abstract** **void** persist();

}

**public** **class** filePersistance **extends** Persistance {

@Override

**public** **void** persist() {

System.***out***.println("data is being saved in in file");

}

}

**public** **class** databasePersistance **extends** Persistance {

@Override

**public** **void** persist() {

System.***out***.println("data is not being saved in in databse");

}

}

**public** **class** mainPersistance {

**public** **static** **void** main(String[] args) {

Persistance p =**new** filePersistance();

p.persist();

Persistance p1=**new** databasePersistance();

p1.persist();

}

}

