**Intermediate OOPS Assignment**

[**Q1**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2149)**. Write a singleton class. Confirm that singleton class cannot be inherited.**

**Singleton Class:-**

The singleton design pattern is used to restrict the instantiation of a class and ensures that only one instance of the class exists in the JVM. In other words, a singleton class is a class that can have only one object (an instance of the class) at a time per JVM instance.

**Specifications :**

class SingletonInheritanceCheck{ }  
public class Assignment2Q1 {}

Code:

package JavaAssignment;

class SingletonInheritanceCheck

{

private static SingletonInheritanceCheck *obj*=new SingletonInheritanceCheck();

public String s;

private SingletonInheritanceCheck()

{ s="Singleton class";

}

public static SingletonInheritanceCheck getInstance()

{return *obj*;}

}

public class Assignment2Q1 {

public static void main(String[] args)

{

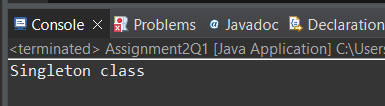
SingletonInheritanceCheck x=SingletonInheritanceCheck.*getInstance*();

System.***out***.println(x.s);

}

}

Output:



[**Q2**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2260)**. 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 of 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.**

**Description:-**

Write a program to claculate the total salaries given to all the employees by an organization. In this organization there are employees category under which Manager and Labour are working where manager will recieve incentive and labourer will receive overtime benifits.

**Specifications:**

class Manager extends Assignment2Q2 {  
    @Override  
    public int getSalary(int salary) {  
        int incentive = 5000;  
    }  
}  
  
class Labour extends Assignment2Q2 {  
    @Override  
    public int getSalary(int salary) {  
        int overtime = 500;  
    }  
}  
  
public class Assignment2Q2 {  
    int salary = 10000;  
    public int getSalary(int salary){}  
    public int totalEmployeesSalary(ArrayList<Integer> employeeSalaries){}  
    public static void main(String[] args) {}  
}

Code:

package JavaAssignment;

import java.util.ArrayList;

class Manager extends Assignment2Q2 {

*@Override*

public int getSalary(int salary) {

int incentive = 5000;

return salary+incentive;

}

}

class Labour extends Assignment2Q2 {

*@Override*

public int getSalary(int salary) {

int overtime = 500;

return salary+overtime;

}

}

public class Assignment2Q2 {

static int *salary* = 10000;

public int getSalary(int salary)

{

return salary;

}

public int totalEmployeesSalary(ArrayList<Integer> employeeSalaries){

int total = 0;

for(int i = 0; i < employeeSalaries.size(); i++)

total += employeeSalaries.get(i);

return total;

}

public static void main(String[] args) {

Manager m=new Manager();

int msalary=m.getSalary(*salary*);

Labour l=new Labour();

int lsalary=l.getSalary(*salary*);

ArrayList<Integer> employeeSalaries=new ArrayList<Integer>();

employeeSalaries.add(msalary);

employeeSalaries.add(lsalary);

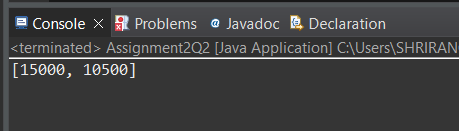
Assignment2Q2 o=new Assignment2Q2();

System.***out***.println(employeeSalaries);

}

}

Output:



[**Q3**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2261)**. 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.**

**Description:-**

Write a program to claculate the total cash available in the bank. In this bank,bank will provide two types of accounts one is savings accounts and another is current account where current account has cash credits and savings account has fixed deposit options.

**Specifications:**

class CurrentAccount extends Assignment2Q3 {  
    int totalDeposits = 10000;  
    int creditLimit = 2000;  
    @Override  
    public int getCash() {}  
}  
class SavingsAccount extends Assignment2Q3 {  
    int totalDeposits = 10000;  
    int fixedDepositAmount = 5000;  
    @Override  
    public int getCash() {}  
}  
public class Assignment2Q3 {  
    public int totalCashInBank(ArrayList<Integer> cash){}  
    public int getCash(){}  
    public static void main(String[] args) {}  
}

Code:

package JavaAssignment;

import java.util.ArrayList;

class CurrentAccount extends Assignment2Q3 {

int totalDeposits = 10000;

int creditLimit = 2000;

*@Override*

public int getCash() {

return totalDeposits-creditLimit;}

}

class SavingsAccount extends Assignment2Q3 {

int totalDeposits = 10000;

int fixedDepositAmount = 5000;

*@Override*

public int getCash() {

return totalDeposits+fixedDepositAmount;}

}

public class Assignment2Q3 {

public int totalCashInBank(ArrayList<Integer> cash){

int total = 0;

for(int i = 0; i < cash.size(); i++)

total += cash.get(i);

return total;

}

public int getCash(){

return 0;

}

public static void main(String[] args) {

CurrentAccount cAccount=new CurrentAccount();

SavingsAccount sAccount=new SavingsAccount();

int cAmount=cAccount.getCash();

int sAmount=sAccount.getCash();

ArrayList<Integer> cash=new ArrayList<Integer>(2);

cash.add(cAmount);

cash.add(sAmount);

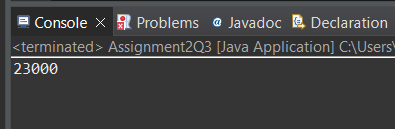
Assignment2Q3 o=new Assignment2Q3();

System.***out***.println(o.totalCashInBank(cash));

}

}

Output:



**4. Test the following principles of an abstract class:**

**• 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.**

**Description:-**

Write a program in such a way that all the conditions above for abstract class should satisfy.

Code:

package JavaAssignment;

abstract class A{

int abstract\_var;

}

abstract class B extends A{

public abstract void fun();

}

public class Assignment2Q4 extends B {

public void fun() {

System.***out***.println("Hello World");

}

public static void main(String args[])

{

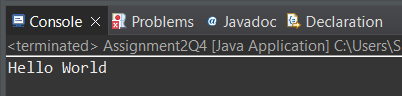
B obj =new Assignment2Q4();

obj.fun();

}

}

Output:



[**Q5**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2163)**. 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.**

**Description:-**

Write a java class named Shape which has abstract draw() and also classes Line, Rectangle, Cube etc. extends Shape to implement the draw method.

**Specifications:-**

class Rectangle5 extends Shape5 {  
    @Override  
    public String draw() {}  
}  
  
class Line5 extends Shape5{  
    @Override  
    public String draw() {}  
}  
  
class Cube5 extends Shape5 {  
    @Override  
    public String draw() {}  
}  
abstract class Shape5 {  
    abstract public String draw();  
}  
public class Assignment2Q5{  
    public static void main(String[] args) {}  
}

Code:

package JavaAssignment;

class Rectangle5 extends Shape5 {

*@Override*

public String draw() {

return "Rectangle";}

}

class Line5 extends Shape5{

*@Override*

public String draw() {

return "Line";}

}

class Cube5 extends Shape5 {

*@Override*

public String draw() {

return "Cube";

}

}

abstract class Shape5 {

abstract public String draw();

}

public class Assignment2Q5 {

public static void main(String[] args) {

Shape5 s1=new Cube5();

Shape5 s2=new Line5();

Shape5 s3=new Rectangle5();

System.***out***.println(s1.draw());

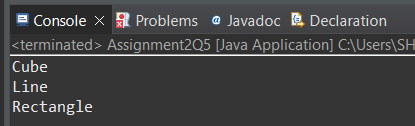
System.***out***.println(s2.draw());

System.***out***.println(s3.draw());

}

}

Output:



[**Q6**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2145)**. 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.**

**Description:-**

Write a program having an abstract class "Persistence" which have two child classes "FilePersistence","DatabasePersistence".The base class will have a persist() method and it is overridden by sub classes.wite a seperate class of client which will get persistence object and invoke persist method on it.(Polymorphism)

**Specifications:**

abstract class Persistence {  
    abstract public String persist();  
}  
class FilePersistence extends Persistence{  
    @Override  
    public String persist() {}  
}  
class DatabasePersistence extends Persistence{  
    @Override  
    public String persist() {}  
}  
  
public class Assignment2Q6 {  
    public static void main(String[] args) {}  
}

Code:

package JavaAssignment;

abstract class Persistence {

abstract public String persist();

}

class FilePersistence extends Persistence{

*@Override*

public String persist() {return "Persistence File";}

}

class DatabasePersistence extends Persistence{

*@Override*

public String persist() {return "Persistence Database";}

}

public class Assignment2Q6 {

public static void main(String[] args) {

Persistence f=new FilePersistence();

Persistence d=new DatabasePersistence();

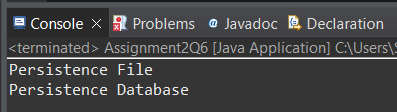
System.***out***.println(f.persist());

System.***out***.println(d.persist());

}

}

Output:



[**Q7**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2169)**. Develop an application for Dessert shop. The application should allow owner to add items like Candy, Cookie or Ice Cream in the shop storage. Also customers should be able to place an order.**

**DessertItem is an abstract class having an abstract method getCost(). Every dessert item has tax associated. Candy item is sold in dollar currency, Cookie in Euro currency & Ice Cream in Rupees currency. The sub classes are supposed to override these methods. When we run the application, it should ask us our role i.e. owner or customer. If role is owner, we should be able to add dessert items in our storage. If role is customer, then we should be able to place an order. The currency conversion rates are:**

**1 dollar = 60 rupees.**

**1 euro = 70 rupees.**

**Specifications:-**

abstract class DesertItem {

    abstract public int getCost();  
}  
class Candy extends DesertItem {  
    public int addCandies(int candies){}  
}  
class Cookie extends DesertItem {  
    public int addCookies(int candies){}  
}  
class IceCream extends DesertItem {  
    public int addIceCreams(int candies){}  
}  
public class Assignment2Q7 {  
    public static void main(String[] args) {}  
    private void selectRoles(){}  
    private void roles(String role){}  
    private void addItems() {}  
    private void addItemsOperation(int choice) {}  
    private void placeOrder() {}  
    private void placeOrderOperation(int choice) {}  
}

Code:

package JavaAssignment;

import java.util.Scanner;

abstract class DesertItem

{

abstract public int getCost();

}

class Candy extends DesertItem

{ //Rate of 1 candy=2 dollars=2\*60=120rs.

int cost=120;

public int getCost()

{return cost; }

public int addCandies(int candies)

{return candies; }

}

class Cookie extends DesertItem

{ //Rate of 1 cookie=3 euro=3\*70=210rs.

int cost = 210;

public int addCookies(int cookies)

{return cookies;}

public int getCost()

{return cost; }

}

class IceCream extends DesertItem

{ //Rate of 1 Icecream=100rs;

int cost=100;

public int addIceCream(int icecream)

{return icecream;}

public int getCost()

{return cost; }

}

public class Assignment2Q7 {

public static void main(String[] args)

{

Assignment2Q7 obj = new Assignment2Q7();

obj.selectRoles();

}

private void selectRoles()

{

System.***out***.println("Select 1 for owner \nSelect 2 for customer");

int choice;

Scanner s=new Scanner(System.***in***);

choice=s.nextInt();

switch(choice){

case 1:

roles("owner");

break;

case 2:

roles("customer");

break;

default:

System.***out***.println("Invalid Choice");

break;

}

}

private void roles(String role)

{

if(role=="owner")

{addItems(); }

else if(role=="customer")

{placeOrder(); }

}

private void addItems()

{

System.***out***.println("Press 1 to add Candies\n Press 2 to add Cookies \n Press 3 to add IceCreams");

int choice=0;

Scanner s = new Scanner(System.***in***);

choice=s.nextInt();

switch(choice)

{

case 1:

addItemsOperation(1);

break;

case 2:

addItemsOperation(2);

break;

case 3:

addItemsOperation(3);

break;

default:

System.***out***.println("Invalid choice");

break;

}

}

private void addItemsOperation(int choice)

{

Scanner s = new Scanner(System.***in***);

if(choice == 1) {

System.***out***.println("Enter number of candies");

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

Candy obj1= new Candy();

System.***out***.println(obj1.addCandies(quantity));

}else if(choice == 2) {

System.***out***.println("Enter number of cookies");

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

Cookie obj2= new Cookie();

System.***out***.println(obj2.addCookies(quantity));

}

else {

System.***out***.println("Enter number of ice creams");

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

IceCream obj3 = new IceCream();

System.***out***.println(obj3.addIceCream(quantity));

}

s.close();

}

private void placeOrder()

{

System.***out***.println("Enter your choice: \n 1. Candies\n 2. Cookies\n 3. Ice Cream");

int choice;

Scanner s = new Scanner(System.***in***);

choice=s.nextInt();

switch(choice) {

case 1:

placeOrderOperation(1);

break;

case 2:

placeOrderOperation(2);

break;

case 3:

placeOrderOperation(3);

break;

default:

System.***out***.println("Invalid choice");

}

}

private void placeOrderOperation(int choice)

{

Scanner s = new Scanner(System.***in***);

DesertItem obj = null;

if(choice ==1) {

System.***out***.println("Enter quantity");

try {

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

obj = new Candy();

int cash = quantity\*obj.getCost();

System.***out***.println("Pay: "+cash);

}catch(Exception e){

e.printStackTrace();

}

}else if(choice == 2) {

System.***out***.println("Enter quantity");

try {

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

obj = new Cookie();

int cash = quantity\*obj.getCost();

System.***out***.println("Pay: "+cash);

}catch(Exception e){

e.printStackTrace();

}

}else {

System.***out***.println("Enter quantity");

try {

int quantity = 0;

if(s.hasNext())

quantity = s.nextInt();

obj = new IceCream();

int cash = quantity\*obj.getCost();

System.***out***.println("Pay: "+cash);

}catch(Exception e){

e.printStackTrace();

}

}

s.close();

}

}

Output:

