1. Write a singletone class. Confirm that singletone class cannot be inherited.

```java
package Assignment2.java;

public class SingleInherit {
    private static SingleInherit obj=null;
    private SingleInherit() {
    }
    public static SingleInherit getSingleton() {
        if(obj==null)
            obj=new SingleInherit();
        return obj;
    }
}
```

```java
package Assignment2.java;

public class A {

    public static void main(String[] args) {
        SingleInherit obj = SingleInherit.getSingleton();
        System.out.println(obj);
        SingleInherit obj1 = SingleInherit.getSingleton();
        System.out.println(obj1);
    }

}
```

Console × Problems

<terminated> A [Java Application] C:\Users\valaanus\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp

```
Assignment2.java.SingleInherit@626b2d4a
Assignment2.java.SingleInherit@626b2d4a
```

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 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.

```
1  package Assignment2.java;
2  class Employee {
3          void work(long salary) {
4          System.out.println("Employee Salary : "+salary);
5      }
6  }
7  class Manager1 extends Employee {
8       void work(long salary,long incentive) {
9          System.out.println("Manager Salary : "+(salary+incentive));
10     }
11 }
12 class Labour extends Employee {
13     void work(long salary,long overtime) {
14         System.out.println("Labour Salary : "+(salary+overtime));
15     }
16 }
17 public class Manager {
18
19     public static void main(String args[]) {
20         Employee e=new Employee();
21         Manager1 m=new Manager1();
22         Labour l=new Labour();
23         e.work(25000);
24         m.work(30000,2000);
25         l.work(456738,3348);
26
27     }
28 }
29
```

Console × Problems

<terminated> Manager [Java Application] C:\Users\valaanus\.p2\pool\plugins\org.eclipse

```
Employee Salary : 25000
Manager Salary : 32000
Labour Salary : 460086
```

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.

```java
package Assignment2.java;

class Rectangle extends Pr {
    public void draw() {
        System.out.println("Rectangle Created");
    }
}
class Line extends Pr {
    public void draw() {
        System.out.println("Line created");
    }
}
class Cube extends Pr {
    public void draw() {
        System.out.println("Cube created");
    }
}
abstract class Pr {
    abstract public void draw();
}
public class Shape {

    public static void main(String args[]) {
        Pr r = new Rectangle();
        r.draw();
        Pr l = new Line();
        l.draw();
        Pr c = new Cube();
        c.draw();
    }

}
```

Console × Problems

<terminated> Shape [Java Application] C:\Users\valaanus\.p2\poo

```
Rectangle Created
Line created
Cube created
```

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.

```java
package Assignment2.java;

class Rectangle extends Pr {
    public void draw() {
        System.out.println("Rectangle Created");
    }
}
class Line extends Pr {
    public void draw() {
        System.out.println("Line created");
    }
}
class Cube extends Pr {
    public void draw() {
        System.out.println("Cube created");
    }
}
abstract class Pr {
    abstract public void draw();
}
public class Shape {

    public static void main(String args[]) {
        Pr r = new Rectangle();
        r.draw();
        Pr l = new Line();
        l.draw();
        Pr c = new Cube();
        c.draw();
    }

}
```

Console × Problems

terminated> AssignmentQ6 [Java Application] C:\Users\valaanus\.p2\

```
persistence class called
```

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.

```java
package Assignment2.java;



import java.util.ArrayList;



public class Assignment2Q3 {
    public int totalCashInBank(ArrayList<Integer> cash){
        int availBal=0;
        for(Integer i:cash){
            availBal+=i;
        }
        return availBal;
    }
    public int getCash(){
        return 0;
    }
    public static void main(String[] args) {
         Assignment2Q3 assignment2Q3 = new Assignment2Q3();  // TODO 



        CurrentAccount currentAccount = new CurrentAccount();
         int currentBal = currentAccount.getCash();
         System.out.println(currentBal);

         SavingsAccount savingsAccount = new SavingsAccount();
         int savingBal = savingsAccount.getCash();
         System.out.println(savingBal);
         ArrayList<Integer> cash = new ArrayList<>();
         cash.add(currentBal);
         cash.add(savingBal);

         int g= assignment2Q3.totalCashInBank(cash);
         System.out.println(g);

    }
}
class CurrentAccount extends Assignment2Q3 {
    int totalDeposits = 10000;
    int creditLimit = 2000;

    public int getCash() {
        return totalDeposits-creditLimit;
    }
}
class SavingsAccount extends Assignment2Q3 {
    int totalDeposits = 10000;
    int fixedDepositAmount = 5000;
```

```
51      int fixedDepositAmount = 5000;
52
53      public int getCash() {
54          return totalDeposits+fixedDepositAmount;
55      }
56  }
```

Console × Problems

<terminated> Assignment2Q3 [Java Application] C:\Users\valaanus\.p2\pool\plugins\org.ecli

```
8000
15000
23000
```

7. Develop an application for Dessert shop. The application should allow owner to add items like Candy, Cookie or IceCream 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 & IceCream 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.

```java
package Assignment2.java;
import java.util.Scanner;



abstract class DesertItem{
    abstract public int getCost();
}
class Candy extends DesertItem {
    int TotalCandies=0;
    int DollarInRupee=60;
    public int addCandies(int candies) {
        this.TotalCandies+=candies;
        return TotalCandies;
    }
    public int getCost() {
        return TotalCandies*DollarInRupee;
    }
}
class Cookie extends DesertItem {
    int TotalCookies=0;
    int EuroInRupee=70;
    public int addCookies(int cookies) {
        TotalCookies += cookies;
        return TotalCookies;
    }
    public int getCost() {
        return TotalCookies*EuroInRupee;
    }
}
class IceCream extends DesertItem {
    int totalIceCream=0;
    public int addIceCreams(int iceCreams) {
        totalIceCream+=iceCreams;
        return totalIceCream;
    }
```

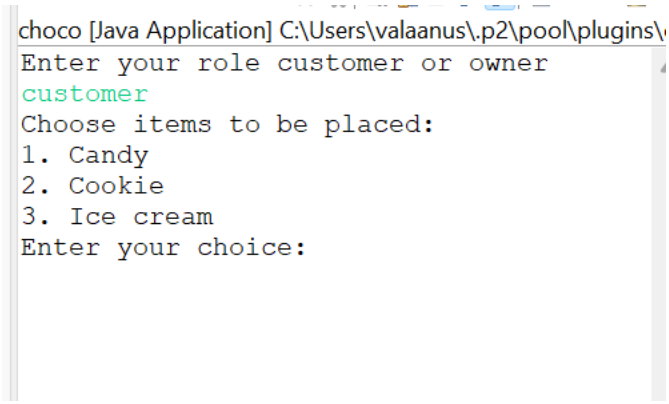
```
    public int getCost() {
        return 0;
    }
}
public class choco {
    Scanner sc = new Scanner(System.in);
    private void selectRoles() {
        System.out.println("Enter your role customer or owner");
        String role;
        role=sc.next();
        roles(role);
    }
    private void roles(String role) {
        if(role.toUpperCase().equals("CUSTOMER")) {
            placeOrder();
        }
        else if(role.toUpperCase().equals("OWNER"))
            addItems();
        else
            System.out.println("Please enter valid details!");
    }
    private void addItems() {
        System.out.println("Choose items to be added :");
        System.out.println("1. Candy\n2. Cookie\n3. Icecream");
        System.out.println("Enter your choice: ");
        int ch;
        ch = sc.nextInt();
        addItemsOperation(ch);
    }
    private void addItemsOperation(int choice) {
        switch(choice) {
        case 1:
            Candy candy = new Candy();
            System.out.println("Enter the amount of candies to be added");
            int amount;
            amount=sc.nextInt();
```

```
            int total = candy.addCandies(amount);
            System.out.println("Candies added succesfully and total available are: "+total);
            break;
        case 2:
            Cookie cookie = new Cookie();
            System.out.println("Enter the amount of cookies to be added");
            amount=sc.nextInt();
            total = cookie.addCookies(amount);
            System.out.println("Cookies added successfully and total available are: "+total);
            break;
        case 3:
            IceCream iceCream = new IceCream();
            System.out.println("Enter the amount of Ice cream to be added");
            amount=sc.nextInt();
            total=iceCream.addIceCreams(amount);
            System.out.println("Ice creams added successfully and total available are: "+total);
        default:
            System.out.println("Please enter valid number");
            break;
        }

    }
    private void placeOrder() {
        System.out.println("Choose items to be placed: ");
        System.out.println("1. Candy\n2. Cookie\n3. Ice cream");
        System.out.println("Enter your choice: ");
        int ch;
        ch = sc.nextInt();
        placeOrderOperation(ch);
    }
    private void placeOrderOperation(int choice) {
        switch(choice) {
        case 1:
             Candy candy = new Candy();
             System.out.println("Enter the amount of candles to be placed: ");
             int total = candy.getCost();
             System.out.println("Candles placed successfully and total cost is: "+total);
             break;
          case 2:
             Cookie cookie = new Cookie();
             System.out.println("Enter the amount of cookies to be placed: ");
             total = cookie.getCost();
             System.out.println("Cookies placed successfully and total cost is: "+total);
             break;
          case 3:
             IceCream iceCream = new IceCream();
             System.out.println("Enter the amount of Icecream to be added: ");
             total = iceCream.getCost();
             System.out.println("Icecreams placed successfully and total cost is: "+total);
          default:
             System.out.println("Please enter valid number!");
             break;
          }
     }
     public static void main(String[] args) {
          choco git = new choco();
          git.selectRoles();
     }



}
```


choco [Java Application] C:\Users\valaanus\.p2\pool\plugins\
Enter your role customer or owner
customer
Choose items to be placed:
1. Candy
2. Cookie
3. Ice cream
Enter your choice: