CORE JAVA

INTERMEDIATE OOP ASSIGNMENT

1. **Write a singleton class. Confirm that a singleton class cannot be inherited.**
   Code:-

*Singleton.java

```
class Test
{
	private static Test s=null;
	public String str;
	private Test()
	{
		str="Singleton Class";
	}
	public static Test getInstance()
	{
		if(s==null)
		{
			s=new Test();
		}
		return s;
	}
}
class Singleton
{
	public static void main(String args[])
	{
		Test a= Test.getInstance();
		Test b=Test.getInstance();
		a.str=(a.str).toUpperCase();
		System.out.println("String-a:"+a.str);
		System.out.println("String-b:"+b.str);
		System.out.println("Hence,proved.");
	}
}
```

Problems | Javadoc | Declaration | Console | Git Staging

```
<terminated> Singleton [Java Application] C:\Users\MBALKRIS\.p2\pool\pl
String-a:SINGLETON CLASS
String-b:SINGLETON CLASS
Hence,proved.
```

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 subclasses 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.**
   Code:-

Employee.java

Singleton.java | Organization.java | Manager.java | *Employee.java | Labour.java

```
package oop;
public class Employee {
    int id,incentive,overtime;
    String name;
    double base_salary;
    public Employee(int a,String b,double c) {
        this.id=a;
        this.name=b;
        this.base_salary=c;
    }
    public void salary() {
        double sal=base_salary;
        System.out.println("Base salary is:"+sal);
    }
}
```

Manager.java

Singleton.java | Organization.java | Manager.java | *Employee.java | Labour.java

```
package oop;
public class Manager extends Employee{
    double c;
    public Manager(int a, String b, double c) {
        super(a, b, c);
        this.c=c;
    }
    public void salary_calc(int incentive) {
        double sal = c+incentive;
        System.out.println("Manager's Salary is:"+sal);
    }
}
```

Labour.java

```
package oop;
public class Labour extends Employee {
    double c;
    public Labour(int a, String b, double c) {
        super(a, b, c);
        this.c=c;
    }
    public void salary_calc(int overtime) {
        double sal = c+overtime;
        System.out.println("Labour's Salary is : "+sal);
    }
}
```

Organization.java

```
package oop;
public class Organization {
    public static void main(String args[]) {
        Manager m = new Manager(123,"Mrunal",70000);
        m.salary_calc(5000);
        Labour l=new Labour(145,"Abc",10000);
        l.salary_calc(300);
    }
}
```

```
Manager's Salary is:75000.0
Labour's Salary is : 10300.0
```

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.**

Code:-

Account.java

```
package bank;
public class Account
{
    int id;
    double curr=0.00;
    String name;
    double sal;
    public Account(int id,String name)
    {
        this.id=id;
        this.name=name;
    }
}
```

Save.java

```
package bank;
import java.util.*;
public class Save extends Account
{
    double cu1;
    public Save(int id, String name)
    {
        super(id,name);
    }
    public void sav() {
        System.out.println("Enter the amount in savings :");
        Scanner sc= new Scanner(System.in);
        cu1=sc.nextDouble();
        sc.close();
    }
    public void disp() {
        System.out.println("The credentials are : "+id+" "+name);
    }
    public void total(int cu) {
        double tot1=cu1+cu;
        System.out.println("Total assets worth : "+tot1);
    }
}
```

Calculate.java

```
package bank;
public class Calculate
{
    public static void main(String args[])
    {
        Save s = new Save(1234,"Mrunal");
        s.disp();
        s.sav();
        s.total(1000);
    }
}
```

```
The credentials are : 1234 Mrunal
Enter the amount in savings :
12000
Total assets worth : 13000.0
```

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 subclass or declare subclass as abstract.**
    - **Abstract class cannot be private.**
    - **Abstract class cannot be final.**

- **You can declare a class abstract without having an abstract method.**

Code:-

AbstractClass.java

```
Abstractmethod.java  AbstractClass.java ×
 1 package abstract_method;
 2 public abstract class AbstractClass
 3 {
 4 int x = 10;
 5 AbstractClass()
 6 {
 7     System.out.println("Abstract Class ");
 8 }
 9 final void m1()
10 {
11     System.out.println("Final method");
12 }
13 void m2()
14 {
15     System.out.println("Instance method");
16 }
17 static void m3()
18 {
19 System.out.println("Static method");
20 }
21 abstract void msg();
22 }
```

Abstractmethod.java

```
Abstractmethod.java ×  AbstractClass.java
 1 package abstract_method;
 2 public class Abstractmethod extends AbstractClass
 3 {
 4   Abstractmethod()
 5   {
 6     System.out.println("Abstract Class Constructor");
 7   }
 8  void msg()
 9  {
10    System.out.println("Hello Java");
11  }
12 public static void main(String[] args)
13 {
14     Abstractmethod t = new Abstractmethod();
15     t.msg();
16     t.m1();
17     t.m2();
18     m3();
19    System.out.println("x = " +t.x);
20  }
21 }
```

```
Problems  Javadoc  Declaration  Console ×  Git Staging
<terminated> Abstractmethod (1) [Java Application] C:\Users\MBALKRIS\.p2\pool
Abstract Class
Abstract Class Constructor
Hello Java
Final method
Instance method
Static method
x = 10
```

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.**

Code:-

Draw.java

```
Shapes.java  Draw.java ×
 1 public abstract class Draw
 2 {
 3     public abstract void draw();
 4 }
 5 class line extends Draw
 6 {
 7     @Override
 8     public void draw()
 9     {
10         System.out.println("Drawing line");
11     }
12 }
13 class rectangle extends Draw
14 {
15     @Override
16     public void draw()
17     {
18         System.out.println("Drawing a rectangle");
19     }
20 }
21 class cube extends Draw
22 {
23     @Override
24     public void draw() {
25         System.out.println("Drawing a cube");
26     }
27 }
```

Shapes.java

```
Shapes.java ×  Draw.java
 1
 2 public class Shapes {
 3     public static void main(String args[]) {
 4         Draw d = new line();
 5         d.draw();
 6         Draw d1= new rectangle();
 7         d1.draw();
 8         Draw d2=new cube();
 9         d2.draw();
10     }
11 }
12
13
```

```
Problems  Javadoc  Declaration  Console ×  Git Staging
<terminated> Shapes [Java Application] C:\Users\MBALKRIS\.p2\poo
Drawing line
Drawing a rectangle
Drawing a cube
```

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 subclasses. 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.**

Code:-

Per_classes.java

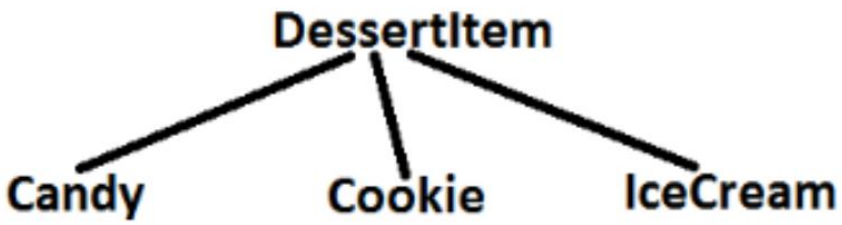
```
Persistence.java  Per_classes.java
1  package persist;
2  public class Per_classes
3  {
4
5  }
6  abstract class persist
7  {
8      abstract void per();
9  }
10 class filepersistence extends persist
11 {
12     @Override
13     void per()
14     {
15         System.out.println("Executing File Persistence.");
16
17     }
18
19 }
20 class databasepersistence extends persist
21 {
22     void per()
23     {
24         System.out.println("Executing Database Persistence.");
25     }
26 }
27
```

Persistence.java

```
Persistence.java  Per_classes.java
1  package persist;
2  public class Persistence
3  {
4  public static void main(String args[])
5  {
6      persist p = new filepersistence();
7      p.per();
8      persist p1=new databasepersistence();
9      p1.per();
10 }
11 }
12
13

Problems  Javadoc  Declaration  Console  Git Stagi
<terminated> Persistence [Java Application] C:\Users\MBALKRIS\.
Executing File Persistence.
Executing Database Persistence.
```

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 and IceCream in Rupees currency. The subclasses 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

Code:-

Candy.java

*Shop.java | Dessertshop.... | Candy.java | Customer.java | Icecream.java | Owner.java

```
1  package shop;
2  public class Candy extends Dessertshop
3  {
4      String dessertName = "Candy";
5      int dessertCost  = 60;
6      @Override
7      public int getCost()
8      {
9
10         return dessertCost;
11     }
12
13 }
14
```

Cookies.java

*Shop.java | Dessertshop.... | Candy.java | *Cookies.java | Customer.java | Owner.java

```
1  package shop;
2  public class Cookies extends Dessertshop
3  {
4      String dessertName = "Cookies";
5      int dessertCost  = 50;
6      @Override
7      public int getCost()
8      {
9
10         return dessertCost;
11     }
12
13 }
```

Icecream.java

*Shop.java | Candy.java | *Cookies.java | Customer.java | Icecream.java | Owner.java

```
1  package shop;
2  public class Icecream extends Dessertshop
3  {
4          String dessertName = "IceCream";
5          int dessertCost  = 70;
6          @Override
7          public int getCost()
8          {
9              return dessertCost;
10         }
11 }
```

Dessertshop.java

*Shop.java | Dessertshop.... | Candy.java

```
1  package shop;
2  public abstract class Dessertshop
3  {
4       int numOfDesert=0;
5       public abstract int getCost();
6  }
```

Customer.java

*Shop.java | Dessertshop.... | Candy.java | *Cookies.java | *Customer.java | Icecream.java

```
1  package shop;
2  import java.util.*;
3  public class Customer
4  {
5  int noOfCandy,noOfCookies,noOfIcecream;
6  public void placeOrder(Dessertshop candy , Dessertshop cookies , Dessertshop icecream)
7  {
8  try (Scanner sc = new Scanner(System.in))
9  {
10 System.out.println("");
11 Boolean isTrue = true;
12 while(isTrue)
13 {
14 System.out.println("Select your order :"+"\n"+ "1. candy(60rs)" +"\n"+"2. cookies(50)"
15 +"\n"+"3. icecream(70)"+"\n"+"Press 1 for candy, 2 for cookies, 3 for icecream"
16 +"\n");
17 int selected = sc.nextInt();
18 switch (selected)
19 {
20 case 1: {
21             System.out.println("Enter number of Candies to update : ");
22             noOfCandy = sc.nextInt();
23             cookies.numOfDesert -=  noOfCandy;
24             break;
25         }
26 case 2: {
27             System.out.println("Enter number of Cookies to update : ");
28             noOfCookies = sc.nextInt();
29             cookies.numOfDesert -=  noOfCookies;
30             break;
31         }
32 case 3: {
```

Customer.java

*Shop.java | Dessertshop.... | Candy.java | *Cookies.java | *Customer.java | Icecream.java

```
29             cookies.numOfDesert -=  noOfCookies;
30             break;
31     }
32 case 3: {
33             System.out.println("Enter number of Icecream to update : ");
34             noOfIcecream = sc.nextInt();
35             icecream.numOfDesert -= noOfIcecream;
36             break;
37         }
38 default:System.out.println("Select correct option");
39 }
40 System.out.println("Enter 1 to continue and 0 to exit : ");
41 int num = sc.nextInt();
42 if (num == 0)
43 {
44 isTrue = false;
45 }
46 }
47 }
48 System.out.println("Added successfully!!!");
49 showOrder(candy.getCost(), cookies.getCost(), icecream.getCost());
50 }
51
52 public void showOrder(int a, int b, int c)
53 {
54 System.out.println("Your order is:" + "\n" + "Desserts    "+"Qty    " +"Amount"
55 +"\n"+"1. candy     "+noOfCandy+"     "+(a*noOfCandy)+"\n"+"2. Cookies   "+noOfCookies+"
56 + " "+ "  "+(b*noOfCookies)+"\n"+"3. Icecream  "+noOfIcecream+"      " +(c*noOfIcecream)+"
57 +"\n"+"Total bill"+" "+((a*noOfCandy)+(b*noOfCookies)+(c*noOfIcecream)));
58 }
59 }
60
```

Owner.java

Dessertshop.... Candy.java *Cookies.java *Customer.java Icecream.java *Owner.java

```java
package shop;
import java.util.*;
public class Owner
{
public void addDessert(Dessertshop candy , Dessertshop cookies ,Dessertshop icec
{
try (Scanner sc = new Scanner(System.in))
{
System.out.println("Add Items ");
Boolean isTrue = true;
while(isTrue)
{
System.out.println("Enter number of Candy to add : ");
candy.numOfDesert +=  sc.nextInt();
System.out.println("Enter number of Cookies to add : ");
cookies.numOfDesert +=  sc.nextInt();
System.out.println("Enter number of Icecream to add : ");
icecream.numOfDesert +=  sc.nextInt();
System.out.println("Enter 1 to update more and 0 if completed : ");
int num = sc.nextInt();
if (num == 0)
{
isTrue = false;
}
System.out.println("Added successfully!!!");
}
}
}
}
```

Shop.java

*Shop.java Dessertshop.... *Cookies.java *Customer.java Icecream.java *Owner.jav

```java
package shop;
import java.util.*;
public class Shop
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
Dessertshop candy = new Candy();
Dessertshop cookies = new Cookies();
Dessertshop iceCream = new Icecream();
Owner owner = new Owner();
Customer customer = new Customer();
Boolean isTrue = true;
while(isTrue)
{
System.out.println("\n\nOwner or Customer");
System.out.println("Press 1 for Owner, Press 2 for Customer and 0 to exit");
int key = sc.nextInt();
if(key == 1)
{
System.out.println("\n"+"Owner:");
owner.addDessert(candy, cookies, iceCream);
}
else if(key == 2)
{
customer.placeOrder(candy, cookies, iceCream);
}
else if(key == 0)
{
int num = sc.nextInt();
if (num == 1)
{
```

Shop.java

*Shop.java Dessertshop.... *Cookies.java *Customer.java

```java
else if(key == 2)
{
customer.placeOrder(candy, cookies, iceCream);
}
else if(key == 0)
{
int num = sc.nextInt();
if (num == 1)
{
isTrue = false;
}
}
}
sc.close();
}
}
```

Problems Javadoc Declaration Console Git Staging

```
<terminated> Shop [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins

Owner or Customer
Press 1 for Owner, Press 2 for Customer and 0 to exit
1

Owner:
Add Items
Enter number of Candy to add :
10
Enter number of Cookies to add :
20
Enter number of Icecream to add :
30
Enter 1 to update more and 0 if completed :
0
Added successfully!!!
```

Shop.java

Problems Javadoc Declaration Console Git Staging

```
<terminated> Shop [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\
Owner or Customer
Press 1 for Owner, Press 2 for Customer and 0 to exit
2

Select your order :
1. candy(60rs)
2. cookies(50)
3. icecream(70)
Press 1 for candy, 2 for cookies, 3 for icecream

1
Enter number of Candies to add :
5
Enter 1 to continue and 0 to exit :
1
Select your order :
1. candy(60rs)
2. cookies(50)
3. icecream(70)
Press 1 for candy, 2 for cookies, 3 for icecream

3
Enter number of Icecream to add :
2
Enter 1 to continue and 0 to exit :
0
Added successfully!!!
```

Shop.java

```
Desserts     Qty    Amount
1. candy      5      300
2. Cookies    0       0
3. Icecream   2      140
-----------------------
Total bill 440
```