1. Write a Singleton class. Confirm that singleton class cannot be inherited Singleton.java

```
package single;
single_instance = null;

        public String s;

        private Singleton()
        {
            s = "Hello I am a string part of Singleton
class";
        }
public class Singleton {

        private static Singleton


        public static Singleton getInstance()
        {
            if (single_instance == null)
```

```
                single_instance = new Singleton();

            return single_instance;
        }
    }
```

Main.java

```
package oops;

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

        Singleton x = Singleton.getInstance();
        Singleton y = Singleton.getInstance();
        Singleton z = Singleton.getInstance();
        System.out.println("Hashcode of x is "  +
x.hashCode());
        System.out.println("Hashcode of y is "  +
y.hashCode());
        System.out.println("Hashcode of z is " +
z.hashCode());
        if (x == y && y == z)
        {
            System.out.println("Three objects point to
the same memory location on the heap i.e, to the same
object");

        }
```

```
            else {

            System.out.println("Three objects DO NOT
point to the same memory
location on the heap");

            }
        }
    }
```

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 incentives & labour has over time add the functionality to calculate total salary of all the employees use polymorphism

```
class Employees
{
	public static int base =10000;
	int salary()
	{
		return base;
	}

}
class Manager extends Employees{
	int salary() {
		return base+20000;
	}
}
class Labour extends Employees{
	int salary()
	{
		return base+10000;
	}
}
public class  Employee{
	static void printSalary(Employees e) {
		System.out.println(e.salary());
	}
	public static void main(String args[]) {
		Employees e1=new Manager();
		System.out.println("manager salary");
		printSalary(e1);
```

```
                Employees e2=new Employees();
                System.out.println("labour salary");
                printSalary(e2);
        }
}
```

Output:
manager salary
30000
labour salary
10000

3.write program to consider saving and current bank holder package polymorphism;

```
package polymorphism;
public class poly {
        public static void main(String[] args)
        {
                bank b ;

                b=new saving();
                b.display();
                b=new current();
                b.display();
        }
```

```
}
package polymorphism;

class bank {
	void display()
	{
		System.out.println("account");

	}

}
package polymorphism;

public class saving extends bank{
	void display()
	{
		int number=10000;
		System.out.println("saving account
holder");
		System.out.println("Fixed
deposit:"+number);
	}

}
package polymorphism;

public class current extends bank {
	void display()
```

```
        {
                int number=5000;
                System.out.println("Current account
holder");
                System.out.println("credit cash:"+number);
        }


}
```

Output:

saving account holder

Fixed deposit:10000

Current account holder

credit cash:5000

4.Test any following principles of an abstract class

```
Class Demo{
Void call()
{
System.out println(“calling”);
}
Aabstract void music()
{
System.out.println(“playing  music”);
}
```

```
 Public void main (String args[])
{
Demo d1=new Demo();
D1.call();
}
}
```

Error:Multiple markers at the line
The abstract method playmusic in type Demo can only be defined by an abstract class

4B Q:Abstract Class cannot be initiated

```
Abstract class Demo{
Void  phone()
{
System.out.println(“calling”);
}
Public static void main(String args[])
{
Demo d1=new Demo();
D1.phone();
}

}
```

Output:Exceptionin thread “main” java.lang .Error unresolved compilation problem:
Cannot instantiate the type Demo

At oops.Demo.main.java:11)

4 C Q:When we extend an abstract class ,we must either override all the abstract methods in subclass or declare subclass as abstract.

```
Class Demo
{
Public static void main{String args[])
        {
           Demo2 obj-new Demo2();
           Obj.call();
           Obj.playmusic();
         }
}
Abstract class Demo1
{
Abstract void call();
Abstract void playmusic();
}
Class Demo2 extends Demo1
{
Void call()
{
System.out.println(“phone rings”);
}
Void playmusic()
{
```

System.out.println(“playing music”);
}
}

4 D Q:Abstract class cannot be private

When I made the above abstract class private the output: Exception in thread “main” java.lang. Error :unresolved compilation problems:
Illegal modifier for the class Demo1;only public, abstract& final are permitted

4 E Q:

Output: Exception in thread “main” java .lang.Error: unresolved compilation problems:
The class Demo1 can be either abstract or final,not both
The type Demo2 cannot subclass the Demo1

4 F Q:

you can declare class abstract without having any abstract method
Answer : Yes ,we can have abstract class without abstract methods as both are independent concepts
Declaring aclass abstract means that it can not be instantiated on its own.

5.

```
abstract class Shape {
        abstract void draw();
```

```
}
class Line extends Shape{
        void draw() {
        System.out.println("drawing line");

   }
}
class Rectangle extends Shape{
        void draw() {
                System.out.println("drawing rectangle");
        }
}
class Circle extends Shape{
        void draw() {
                System.out.println("drawing circle");
        }
}
public class Shape{
        public static void main(String args[])
        {
                Shapes s1=new Line();
                s1.draw();
                Rectangle s2=new Rectangle();
                s2=draw();
                Circle s3=new Circle();
                s3.draw();
        }
```

}

6.Develop an application for dessert shop. The application should allow the owner to add items like candy,cookie,ice cream in shop storage. Also customer should be able to place an order.

DessertItem.java

```
package oops;

public abstract class DessertItem {
	protected String name;
	public DessertItem()
	{
		name="";
	}
	public DessertItem(String name1)
	{
		name=name1;
	}
	public String getName()
	{
		return name;
	}
	public void setName(String name1)
	{
		name=name1;
	}
```

```
	public abstract double getCost();

}
```

Candy.java

```
package oops;

public class candy extends DessertItem {
	private double weight;
	private double pricePerPound;
	public candy()
	{
		super();
		weight=0;
		pricePerPound=0;
	}
	public candy(String name,double w,double prc)
	{
		super(name);
		weight=w;
		pricePerPound=prc;
	}
	public double getWeight()
	{
		return weight;
	}
	public void setWeight(double weight)
	{
```

```
		this.weight=weight;
	}
	public double getPricePerPound()
	{
		return pricePerPound;
	}
	public void setPricePerPound(double
pricePerPound)
	{
		this.pricePerPound=pricePerPound;
	}
public double getCost()
{
	double total=weight*pricePerPound;
	total=Math.round(total*100);
	return total;
}
public String toString()
{
	String s=String.format("%-50s$%2f\n\t%.2f
lbs@$.2f",getName(),getCost()/100,weight,pricePerPo
und);
	return s;
}
}
```

7. Cookie.java

```
package oops;
```

```
public class cookie extends DessertItem {
	private int quantity;
	private double pricePerDozen;
	public cookie()
	{
		super();
		quantity=0;
		pricePerDozen=0;
	}
	public cookie(String name,int qty,double prc)
	{
		super(name);
		quantity=qty;
		pricePerDozen=prc;
	}
	public double getQuantity()
	{
		return quantity;
	}
	public double getPricePerDozen()
	{
		return pricePerDozen;
	}
	public void setPricePerDozen(double
pricePerDozen)
	{
		this.pricePerDozen=pricePerDozen;
```

```
	}
	public void setQuantity(int quantity)
	{
		this.quantity=quantity;
	}

public double getCost()
{
	double total=pricePerDozen/12*quantity;
	total=Math.round(total*100);
	return total;
}
public String toString()
{
	String s=String.format("%-50s $.%2f\n\t%d
cookies@ $%.2f per
Dozen",getName(),getCost()/100,quantity,pricePerDoz
en);
return s;
}
}
```

IceCream.java

```
package oops;

public class IceCream extends DessertItem
{
```

```
		private int numberOfScoops;
		private double pricePerScoop;
		private double toppingPrice;
		public IceCream()
		{
			super();
			numberOfScoops=0;
			pricePerScoop=0;
			toppingPrice=0;
		}
		public IceCream(String name,int
scoops,double prcPerScoop,double toppings)
		{
			super(name);
			numberOfScoops=scoops;
			pricePerScoop=prcPerScoop;
			toppingPrice=toppings;

		}
		public int getnumberOfScoops()
		{
			return numberOfScoops;
		}

		public void setnumberOfScoops(int
numberOfScoops)
		{
```

```
	this.numberOfScoops=numberOfScoops;
		}
		public double getPricePerScoop()
		{
			return pricePerScoop;
		}
		public void setPricePerScoop(double
pricePerScoop) {
			this.pricePerScoop=pricePerScoop;
		}

		public double getToppingPrice()
		{
			return toppingPrice;
		}
		public void setToppingPrice(double
toppingPrice)
		{
			this.toppingPrice=toppingPrice;
		}

	public double getCost()
	{
		double
total=(numberOfScoops*pricePerScoop+toppingPrice);
		return Math.round(100*total);
	}
```

```
	public String toString()
	{
		String s=String.format("%-
50s$%.2f\n\t%dscoops@$%.2f/scoop+$%.2f",getName
(),getCost()/100,numberOfScoops,pricePerScoop,toppi
ngPrice);
	return s;
	}
	}
```

DessertShop.java

```
package oops;

public class DessertShop {

public static void main(String arg[])
{
	candy item1=new candy("Peanut Butter
Fudge",2.25,3.99);
	cookie item2=new cookie("Oatmeal Raisin
cookies",4,3.99);
	IceCream item3=new IceCream("VAnilla Ice
Cream",2,1.05,0.45);
	System.out.println(item1);
	System.out.println(item2);
	System.out.println(item3);
}
}
```