6. Write a Java program to achieve concept of Method Overriding

```
package q11271;

class Bank {
float calculateInterest(float principal, int time) {
return 0.0f;
}
}
class SBI extends Bank {
private static final float INTEREST_RATE = 10.8f;
@Override float calculateInterest(float principal, int time) {
return (principal*INTEREST_RATE*time)/100;
}
}
class ICICI extends Bank {
private static final float INTEREST_RATE = 11.6f;
@Override float calculateInterest(float principal, int time) {
return (principal*INTEREST_RATE*time)/100;
}
}

class AXIS extends Bank {
private static final float INTEREST_RATE = 12.3f;
@Override float calculateInterest(float principal, int time) {
return(principal*INTEREST_RATE*time)/100;
}
}

public class TestOverriding {
public static void main(String[] args) {
Bank sbiBank = new SBI();
Bank iciciBank = new ICICI();
Bank axisBank = new AXIS();
float principal = Float.parseFloat(args[0]);
int time = Integer.parseInt(args[1]);
System.out.println("SBI rate of interest = " +
sbiBank.calculateInterest(principal, time));
System.out.println("ICICI rate of interest = " +
iciciBank.calculateInterest(principal, time));
System.out.println("AXIS rate of interest = " +
axisBank.calculateInterest(principal, time));
}
}
```

7. Write a Java program to implement a Constructor

```
package q11116;

public class Staff
{
```

```
private int id;
private String name;
public Staff(int id,String name)
{
this.id=id;
this.name=name;
}
public void show()
{
System.out.println("Id : "+id);
System.out.println("Name : "+name);
}
public static void main(String[] args)
{
if(args.length!=2)
{
System.out.println("not");
return;
}
try{
int id=Integer.parseInt(args[0]);
String name=args[1];
Staff staffObject=new Staff(id,name);
staffObject.show();
}
catch(NumberFormatException e)
{
System.out.println("no");
}
}
}
```

8. Write a Java program Calculating resistance using two resistor objects

```
import java.util.Scanner;


class Resistor {
double resistance;

void giveData(double resistance) {
this.resistance = resistance;
}

void displayData() {
System.out.println("Resistor-1 Resistance:" + resistance);
}
void displayData2()
{
System.out.println("Resistor-2 Resistance:" +resistance);
```

```
}
}

class SeriesCircuit extends Resistor {
static Resistor calculateSeriesResistance(Resistor resistor1, Resistor resistor2) {
double combinedResistance = resistor1.resistance + resistor2.resistance;
Resistor result = new Resistor();
result.giveData(combinedResistance);
return result;
}
}

class ParallelCircuit extends Resistor {
static Resistor calculateParallelResistance(Resistor resistor1, Resistor resistor2) {
double combinedResistance;
if (resistor1.resistance == 0 || resistor2.resistance == 0) {
combinedResistance = 0.0;
} else {
combinedResistance = 1 / ((1 / resistor1.resistance) + (1 / resistor2.resistance));
}
Resistor result = new Resistor();
result.giveData(combinedResistance);
return result;
}
}

public class ResistorExecute {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

Resistor r1 = new Resistor();
Resistor r2 = new Resistor();

System.out.println("Enter resistance value for Resistor-1:");
r1.giveData(scanner.nextDouble());

System.out.println("Enter resistance value for Resistor-2:");
r2.giveData(scanner.nextDouble());

r1.displayData();
r2.displayData2();
Resistor seriesResult =
SeriesCircuit.calculateSeriesResistance(r1, r2);
Resistor parallelResult =
ParallelCircuit.calculateParallelResistance(r1, r2);

System.out.println("Series Resistance:" +
seriesResult.resistance);
```

```
System.out.println("Parallel Resistance:" +
parallelResult.resistance);
}
}
```

9. Write the code to find area of a rectangle and triangle respectively

```
import java.util.*;

System.out.println("Perimeter of Rectangle:" + perimeter);
}
}

class Circle implements GeometricShape {
double radius;

Circle(double radius) {
this.radius = radius;
}

public void area() {
double area = Math.PI * radius * radius;
System.out.println("Area of Circle:" + area);
}

public void perimeter() {
double perimeter = 2 * Math.PI * radius;
System.out.println("Circumference of Circle:" + perimeter);
}
}

public class ExecuteMain {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Choose a geometric shape:");
System.out.println("1.Triangle");
System.out.println("2.Rectangle");
System.out.println("3.Circle");
System.out.println("Enter your choice:");
int choice = sc.nextInt();

switch (choice) {
case 1:
System.out.println("Enter lengths of the three sides of
Triangle:");
double a = sc.nextDouble();
double b = sc.nextDouble();
double c = sc.nextDouble();
Triangle t = new Triangle(a, b, c);
t.area();
```

```
t.perimeter();
break;
case 2:
System.out.println("Enter length and breadth of Rectangle:");
double length = sc.nextDouble();
double width = sc.nextDouble();
Rectangle r = new Rectangle(length, width);
r.area();
r.perimeter();
break;
case 3:
System.out.println("Enter radius of Circle:");
double radius = sc.nextDouble();
Circle c1 = new Circle(radius);
c1.area();
c1.perimeter();
break;
default:
System.out.println("Invalid choice");
}
}}
```

10. Write a Java program to illustrate Preventing inheritance using final keyword

```
package q29645;

import java.util.*;
final class Figure{}

class Square {
int side;

Square(int side) {
this.side = side;
}

double area() {
return side*side;
}
}

class PreventInherit {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of Square: ");
int side = scanner.nextInt();

Square square = new Square(side);
double area = square.area();
```

```
System.out.println("Inside Area of Square");
System.out.println("Area of Square is " + area);
}
}
```