PRACTICAL-1

Write a java program for converting Pound into Rupees. (Accept Pounds from command line argument and using scanner class also and take 1 Pound = 100 Rupees.)

Program Code:

```
import java.util.Scanner;
class SP_11
{
	public static void main(String[] args)
	{
		double pound=0,rupee=0;
		Scanner input=new Scanner(System.in);
		System.out.print("Enter the Value in Pound : ");
		pound=input.nextDouble();
		rupee=pound*100;
		System.out.println("The value of "+pound+" pound is "+rupee+" rupees");
	}
}
```

Output:

```
C:\Java\JAVA_practicals>javac SP_11.java

C:\Java\JAVA_practicals>java SP_11
Enter the Value in Pound : 15.5
The value of 15.5 pound is 1550.0 rupees
```

PRACTICAL-2

Write a program that defines TriangleArea class with three constructor. The first form accept no arguments. The second accept one double value for radius. The third form accept any two arguments.

Program Code:

```
class TriangleArea
{
		double radius;
		String str=new String();
	public TriangleArea()
	{
		radius=0;
		str=null;
		System.out.println("Constructor has no argument");
	}
	public TriangleArea(double r)
	{
		radius=r;
		System.out.println("Constructor has one argument");
	}
	public TriangleArea(String s,double r)
	{
		radius=r;
		str=s;
		System.out.println("Constructor has two arguments");
		System.out.println("STR: "+str);
	}
}
class SP_12
{
	public static void main(String[] args)
```

```
    {
        TriangleArea obj=new TriangleArea();
        TriangleArea obj1=new TriangleArea(10.22);
        TriangleArea obj2=new TriangleArea("TRIANGLE",10.22);
    }
}
```

Output:

```
C:\Java\JAVA_practicals>javac SP_12.java

C:\Java\JAVA_practicals>java SP_12
Constructor has no argument
Constructor has one argument
Constructor has two arguments
STR: TRIANGLE
```

PRACTICAL-3

Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary(double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named Employee Test that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.

Program Code:

```
class Employee
{
	private String firstName;
	private String lastName;
	private double monthlySalary;
	Employee(String fname,String lname,double ms)
	{
		firstName=fname;
		lastName=lname;
		if(ms<0)
			monthlySalary=0.0;
		else
		monthlySalary=ms;
	}
	void set_firstName(String fname)
	{
		firstName=fname;
	}
	void set_lastName(String lname)
	{
		lastName=lname;
```

```
	}
	void set_monthlySalary(double ms)
	{
		if(ms<0)
			monthlySalary=0.0;
		else
			monthlySalary=ms;
	}
	String get_firstName()
	{
		return firstName;
	}
	String get_lastName()
	{
		return lastName;
	}
	double get_monthlySalary()
	{
		return monthlySalary;
	}
	double yearlySalary()
	{
		return 12*monthlySalary;
	}
	void raise(double raisePercent)
	{
		monthlySalary=(monthlySalary*raisePercent)/100+monthlySalary;
	}
}
class SP_13
```

```
{
	public static void main(String[] args)
	{
		Employee emp1=new Employee("Parth","Patel",550000);
		Employee emp2=new Employee("Jason","Taylor",120000);
		System.out.println("Monthly Salary of "+ emp1.get_firstName() +" is "+ emp1.yearlySalary());
		System.out.println("Monthly Salary of "+ emp2.get_firstName() +" is "+ emp2.yearlySalary());
		emp1.raise(10.0);
		emp2.raise(10.0);
		System.out.println("Monthly Salary of "+emp1.get_firstName()+" is "+emp1.yearlySalary());
		System.out.println("Monthly Salary of "+emp2.get_firstName()+" is "+emp2.yearlySalary());
	}
}
```

Output:

```
C:\Java\JAVA_practicals>javac SP_13.java

C:\Java\JAVA_practicals>java SP_13
Monthly Salary of Parth is 6600000.0
Monthly Salary of Jason is 1440000.0
Monthly Salary of Parth is 7260000.0
Monthly Salary of Jason is 1584000.0
```

PRACTICAL-4

Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date's capabilities.

```
class Date
{
	private int day;
	private int month;
	private int year;
	Date(int d,int m,int y)
	{
		day=d;
		month=m;
		year=y;
	}
	void set_day(int d)
	{
		day=d;
	}
	void set_month(int m)
	{
		month=m;
	}
	void set_year(int y)
	{
		year=y;
	}
	int get_day()
```

```
        {
                return day;
        }
        int get_month()
        {
                return month;
        }
        int get_year()
        {
                return year;
        }
        void displayDate()
        {
                System.out.println(month+"/"+day+"/"+year);
        }

}
class SP_14
{
        public static void main(String[] args)
        {
                Date d1=new Date(06,04,2001);
                d1.displayDate();
        }
}
```

Output:

```
C:\Java\JAVA_practicals>javac SP_14.java

C:\Java\JAVA_practicals>java SP_14
4/6/2001
```

PRACTICAL-5

Complete the code and write main () method to execute program.

Program Code:

```
public class MethodOverloading
{
	/*private void methodOverloaded()
	{
		System.out.println("Private method-1");
	}*/
	private int methodOverloaded(int i)
	{
		System.out.println("private int method\n"+"Entered value="+i);
		return i;
	}
	protected int methodOverloaded(double d)
	{
		System.out.println("protected int method\n"+"Entered value= "+d);
		return 1;
	}
	public void methodOverloaded(int i,double d)
	{
		System.out.println("Public int method\n"+"Entered value="+i+" "+d);
	}
}
class SP_15
{
	public static void main(String[] args)
	{
		MethodOverloading m1=new MethodOverloading();
		//m1.methodOverloaded();
```

```
        System.out.println(m1.methodOverloaded(10));
        System.out.println(m1.methodOverloaded(20.99));
        m1.methodOverloaded(15,21.2);
    }
}
```

Output:

Before Commenting :

```
C:\Java\JAVA_practicals>javac MethodOverloading.java
MethodOverloading.java:27: error: no suitable method found for methodOverloaded(no arguments)
                m1.methodOverloaded();
                  ^
    method MethodOverloading.methodOverloaded(int) is not applicable
      (actual and formal argument lists differ in length)
    method MethodOverloading.methodOverloaded(double) is not applicable
      (actual and formal argument lists differ in length)
    method MethodOverloading.methodOverloaded(int,double) is not applicable
      (actual and formal argument lists differ in length)
1 error
```

After Commenting:

```
C:\Java\JAVA_practicals>javac MethodOverloading.java

C:\Java\JAVA_practicals>java SP_15
protected int method
Entered value= 10.0
1
protected int method
Entered value= 20.99
1
Public int method
Entered value=15 21.2
```