

# LAB\_5

## Oops

1)a)

Create an interface named Turner, with a single method named turn(). Create a class named Leaf that implements turn() to display Changing colors. Create a class named Page that implements turn() to display Going to the next page. Create a class named Pancake that implements turn() to display Flipping. Write an application named DemoTurners that creates one object of each of these class types and demonstrates the turn() method for each class. Save the files as Turner.java, Leaf.java, Page.java, Pancake.java, and DemoTurners.java.

Code:

```
interface Turner
{
    abstract void turn();
}

class Leaf implements Turner
{
    public void turn()
    {
        System.out.println("Changing Colors");
    }
}

class PageAttributes implements Turner
{
    public void turn()
    {
        System.out.println("Going to Next page");
    }
}
```

class Pancake implements Turner

```
{  
    public void turn()  
    {  
        System.out.println("Flipping");  
    }  
}
```

public class DemoTurners

```
{  
    public static void main(String args[])  
    {  
        Leaf ob1=new Leaf();  
        PageAttributes ob2=new PageAttributes();  
        Pancake ob3=new Pancake();  
        ob1.turn();  
        ob2.turn();  
        ob3.turn();  
    }  
}
```

Output:

```
C:\Users\MAJJIGA JASWANTH>cd desktop  
C:\Users\MAJJIGA JASWANTH\Desktop>cd java  
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Leaf.java  
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac PageAttributes.java  
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac PanCake.java  
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac DemoTurners.java  
C:\Users\MAJJIGA JASWANTH\Desktop\java>java DemoTurners  
Changing Colors  
Going to Next page  
Flipping  
C:\Users\MAJJIGA JASWANTH\Desktop\java>_
```

1)b)

Think of two more objects that use turn(), create classes for them, and then add objects to the DemoTurners application, renaming it DemoTurners2.java. Save the files, using the names of new objects that use turn().

Code:

```
class Free implements Turner
{
    public void turn()
    {
        System.out.println("Copies are not free");
    }
}

class Quality implements Turner
{
    public void turn()
    {
        System.out.println("Qualities are good");
    }
}

public class DemoTurners2
{
    public static void main(String args[])
    {
        Leaf ob1=new Leaf();
        PageAttributes ob2=new PageAttributes();
        Pancake ob3=new Pancake();
        Free ob4=new Free();
        Quality ob5=new Quality();
        ob1.turn();
        ob2.turn();
        ob4.turn();
    }
}
```

```

        ob3.turn();

        ob5.turn();
    }
}

```

Output:

```

C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Free.java

C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Quality.java

C:\Users\MAJJIGA JASWANTH\Desktop\java>javac DemoTurners2.java

C:\Users\MAJJIGA JASWANTH\Desktop\java>java DemoTurners2
Changing Colors
Going to Next page
Copies are not free
Flipping
Qualities are good

C:\Users\MAJJIGA JASWANTH\Desktop\java>

```

1)c)

Apply Dynamic methoddispatch to show the power of it and name the class as DemoTurners3.

Code:

```

public class DemoTurners3
{
    public static void main(String args[])
    {
        Leaf ob1=new Leaf();

        PageAttributes ob2=new PageAttributes();

        Pancake ob3=new Pancake();

        Free ob4=new Free();

        Quality ob5=new Quality();

        Turner ref;

        ref = ob1;

        ref.turn();
    }
}

```

```

        ref=ob2;
        ref.turn();

        ref=ob3;
        ref.turn();

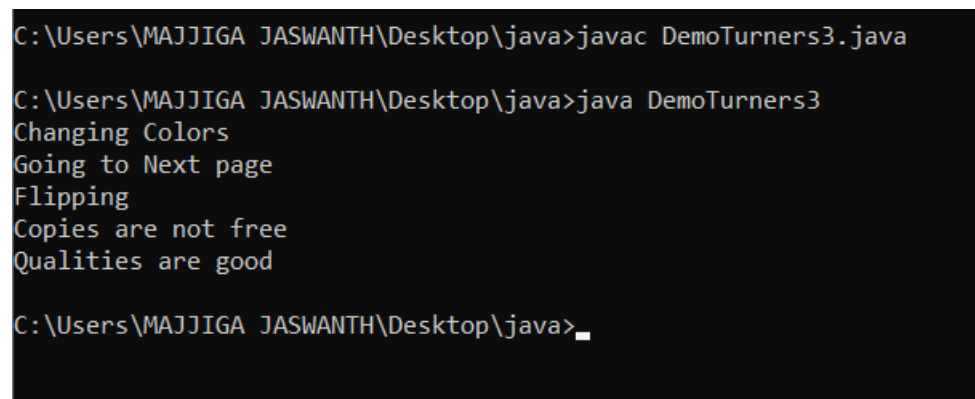
        ref=ob4;
        ref.turn();

        ref=ob5;
        ref.turn();

    }
}

```

Output:



```

C:\Users\MAJJIGA JASWANTH\Desktop\java>javac DemoTurners3.java

C:\Users\MAJJIGA JASWANTH\Desktop\java>java DemoTurners3
Changing Colors
Going to Next page
Flipping
Copies are not free
Qualities are good

C:\Users\MAJJIGA JASWANTH\Desktop\java>_

```

2) Modify question (3) of Lab exercise 4, adding an interface called SidedObject that contains a method called displaySides(); this method displays the number of sides the object possesses. Modify the GeometricFigure subclasses to include the use of the interface to display the number of sides of the figure. Create an application that demonstrates the use of both subclasses. Save the files as GeometricFigure2.java, Square2.java, Triangle2.java, SidedObject.java, and UseGeometric2.java.

Code:

```
import java.util.*;
```

```

interface SidedObject
{
    public void displaySides();
}

abstract class GeometricFigure2 implements SidedObject
{
    String figuretype; int height;
    int width;

    GeometricFigure2(String type,int hght,int width)
    {
        figuretype=type; height=hght; this.width=width;
    }

    abstract void area();
}

class Square2 extends GeometricFigure2
{
    Square2(String type,int side)
    {
        super(type,side,side);
    }

    void area()
    {
        System.out.println("The area of "+figuretype+" is
"+((double)(height)*(double)(height)));
    }

    public void displaySides()
    {
        System.out.println("No of sides : 4");
    }
}

class Triangle2 extends GeometricFigure2

```

```

{
    Triangle2(String type,int hght,int width)
    {
        super(type,hght,width);
    }
    void area()
    {
        System.out.println("The area of "+figuretype+" is
"+(0.5*(double)(height)*(double)(width)));
    }
    public void displaySides()
    {
        System.out.println("No of sides : 3");
    }
}

public class UseGeometric2
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the side of the square");
        int s=sc.nextInt();
        Square2 s2=new Square2("Square",s);
        s2.area();
        s2.displaySides();
        System.out.println("Enter the height and width of the triangle");
        int hght=sc.nextInt();
        int width=sc.nextInt();
        Triangle2 t=new Triangle2("Triangle",hght,width);
        t.area();
        t.displaySides();
    }
}

```

}

Output:

```
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac GeometricFigure2.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Square2.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Traingle2.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac UseGeometric2.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>java UseGeometric2
Enter the side of the square
4
The area of Square is 16.0
No of sides : 4
Enter the height and width of the triangle
8
28
The area of Triangle is 112.0
No of sides : 3
C:\Users\MAJJIGA JASWANTH\Desktop\java>
```

3) Sanchez Construction Loan Co. makes loans of up to \$100,000 for construction projects. There are two categories of Loans—those to businesses and those to individual applicants. Write an application that tracks all new construction loans. The application also must calculate the total amount owed at the due date (original loan amount + loan fee). The application should include the following classes:

- Loan—A public abstract class that implements the LoanConstants interface. A Loan includes a loan number, customer last name, amount of loan, interest rate, and term. The constructor requires data for each of the fields except interest rate. Do not allow loan amounts greater than \$100,000. Force any loan term that is not one of the three defined in the LoanConstants class to a short-term, 1-year loan. Create a toString() method that displays all the loan data.
- LoanConstants—A public interface class. LoanConstants includes constant values for short-term (1 year), medium-term (3 years), and long-term (5 years) loans. It also contains constants for the company name and the maximum loan amount.

- BusinessLoan—A public class that extends Loan. The BusinessLoan constructor sets the interest rate to 1% more than the current prime interest rate.
- PersonalLoan—A public class that extends Loan. The PersonalLoan constructor sets the interest rate to 2% more than the current prime interest rate.
- CreateLoans—An application that creates an array of five Loans. Prompt the user for the current prime interest rate. Then, in a loop, prompt the user for a loan type and all relevant information for that loan.



Store the created Loan objects in the array. When data entry is complete, display all the loans. Save the files as Loan.java, LoanConstants.java, BusinessLoan.java, PersonalLoan.java, and CreateLoans.java

Code:

```
import java.util.*;

public abstract class Loan implements LoanConstants
```

```
{
    long num;
    String name, terms;
    int amt;
    double rate;

    Loan(long a, String b, int c, String d)
    {
        num=a;
        name=b;
        amt=c;
        terms=d;
    }

    public String toString()
    {
        return num + " " + name + " " + amt + " "
            + terms + " " + rate;
    }
}
```

```
interface LoanConstants
```

```
{
```

```
        int short_terms=15000;

        int mid_term=50000;

        int long_term=100000;

        String name="XYZ";

        int max_amt=100000;

    }
```

```
class BusinessLoan extends Loan
{
    BusinessLoan(long a,String b,int c,String d)
    {
        super(a,b,c,d);
        rate=1.0;
    }
}
```

```
class PersonalLoan extends Loan
{
    PersonalLoan(long a,String b,int c,String d)
    {
        super(a,b,c,d);
        rate=2;
    }
}
```

```
import java.util.*;

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

```

{

Scanner ac=new Scanner(System.in);

Loan ob[]=new Loan[5];

long n=0;

String na="",t="";

int a=0;

a:for(int i=0;i<5;i++)
{

    System.out.println("Enter B/P/Q");

    switch(ac.next())
    {

        case "B":

            System.out.println("Enter Loan Number ");

            n=ac.nextLong();

            System.out.println("Enter Name " );

            na=ac.next();

            System.out.println("Enter Amount " );

            a=ac.nextInt();

            System.out.println("Enter Terms " );

            t=ac.next();

            if(a>100000)
            {

                System.out.println("Amount is to much");

            }

            else

            {

                BusinessLoan obj=new BusinessLoan(n,na,a,t);

                ob[i]=obj;

            }

            break;

```

```

case "P":
    System.out.println("Enter Loan Number ");
    n=ac.nextLong();
    System.out.println("Enter Name " );
    na=ac.next();
    System.out.println("Enter Amount " );
    a=ac.nextInt();
    System.out.println("Enter Terms " );
    t=ac.next();
    if(a>100000)
    {
        System.out.println("Amount is to much");
    }
    else
    {
        PersonalLoan obj=new PersonalLoan(n,na,a,t);
        ob[i]=obj;
    }
    break;

case "Q":
    for(int j=0;j<i;j++)
    {
        System.out.println(ob[j].toString());
    }
    break a;

default:
    i--;

```

```
System.out.println("Wrong Choise");
```

```
}
```

```
}
```

```
}
```

```
}
```

## OUTPUT:

```
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac Loan.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac BusinessLoan.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac PersonalLoan.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>javac CreateLoan.java
C:\Users\MAJJIGA JASWANTH\Desktop\java>java CreateLoan
Enter B/P/Q
B
Enter Loan Number
87458745
Enter Name
JASSU
Enter Amount
878
Enter Terms
YES
Enter B/P/Q
P
Enter Loan Number
9999785
Enter Name
PANDU
Enter Amount
74575
Enter Terms
NEVER
Enter B/P/Q
```

B  
Enter Loan Number  
858585  
Enter Name  
IQPERFE  
Enter Amount  
74584  
Enter Terms  
NO  
Enter B/P/Q  
P  
Enter Loan Number  
74125896  
Enter Name  
MJ  
Enter Amount  
96354  
Enter Terms  
YES  
Enter B/P/Q  
P  
Enter Loan Number  
8526937  
Enter Name  
LEON  
Enter Amount  
8524  
Enter Terms  
NO