NAME- Harshita Totala ROLL NO- SEITA14 DIV- IT A
SUBJECT- Object Oriented Programming

ASSIGNMENT NO: 4

Problem Statement- Design a base class shape with two double type values and member functions to input the data and compute_area() for calculating area of figure. Derive two classes' triangle and rectangle. Make compute_area() as abstract function and redefine this function in the derived class to suit their requirements. Write a program that accepts dimensions of triangle/rectangle and display calculated area. Implement dynamic binding for given case study.

Aim- To Study Polymorphism & Abstraction using Java.

Objectives- 1) To understand the concept of Polymorphism using Java
2) To implement run time polymorphism

Theory-

Polymorphism in Java
The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Real life example of polymorphism**:** A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.
Polymorphism is considered as one of the important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word "poly" means many and "morphs" means forms, So it means many forms.
In Java polymorphism is mainly divided into two types:- A]Complie
Time Polymorphism
B]Run Time Polymorphism

1. Compile time polymorphism: It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

Method Overloading: When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments.
Example: By using different types of arguments.

```
//Java program for Method overloading
Class MultiplyFun{
//Mehtod with 2 parameter

static int multiply(int a, int b){ return
          a * b;
}
//method with same name but 2 double parameter
static double multiply(double a, double b){ return
a*b;
}
}

class Main {
public static void main(String[] args)
{
System.out.println(MultiplyFun.Multiply(2, 4));
System.out.println(MultiplyFun.Multiply(5.5, 6.3));
}
}
```

Output:

```
8
34.65
```

Operator Overloading: Java also provide option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator '+' when placed between integer operands, adds them and when placed between string operands, concatenates them.

In java, Only "+" operator can be overloaded:

To add integers

To concatenate strings

Example**:** / Java program for Operator overloading

```
class OperatorOVERDDN {
void operator(String str1, String str2)
{
String s = str1 + str2;
System.out.println("Concatinated String - "
+ s);
}
void operator(int a, int b)
{
int c = a + b;
System.out.println("Sum = " + c);
}
}
```

```
class Main {
public static void main(String[] args)
{
OperatorOVERDDN obj = new
OperatorOVERDDN();  obj.operator(2, 3);
obj.operator("joe", "now");
}
}
```

Output:

```
Sum =5
Concatinated String -joenow
```

2]Runtime polymorphism: It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.
Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

Abstract class-
A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).
Abstraction is a process of hiding the implementation details and showing only functionality to the user.
Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.
Abstraction lets you focus on what the object does instead of how it does it.
Ways to achieve Abstraction
There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.
Points to Remember ○ An abstract class must be declared with an abstract keyword.

- ○ It can have abstract and non-abstract methods.
- ○ It cannot be instantiated.
- ○ It can have constructor and static methods also. ○ It can have final methods which will force the subclass not to change the body of the method.

Example of abstract class

1. abstract class A{}

A method which is declared as abstract and does not have implementation is known as an abstract method. Example of abstract method

```
abstract void printStatus();//no method body and abstract
```

Example of Abstract class that has an abstract method
In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike{
abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
Bike obj = new Honda4();
obj.run();
}
}
```

Steps- 1.
Start

2. Create an abstract class named shape that contains two double type numbers and an empty method named compute_area().
3. Provide two classes named rectangle and triangle such that each one of the classes extends the class Shape.
4. Each of the inherited class from shape class should provide the implementation for the method compute_area().
5. Get the input and calculate the area of rectangle and triangle.
6. In the fourth separate class, create the objects for the two inherited classes and invoke the methods and display the area values of the different shapes.
7. Stop.

Input-
length and breadth of rectangle base
and height of triangle
Output-
Area of rectangle
Area of triangle

Implementation- abstract

```
class shape {
abstract public void compute_area();
}
class rectangle extends shape {
public void compute_area() {
---
}
}
class triangle extends shape {
public void compute_area() {
---
}
}
public class Shapeclass {
public static void main(String[] args) {
---
}
```

Test case or Validation-

Different values for length and breadth of rectangle and base and height of triangle.

Program-

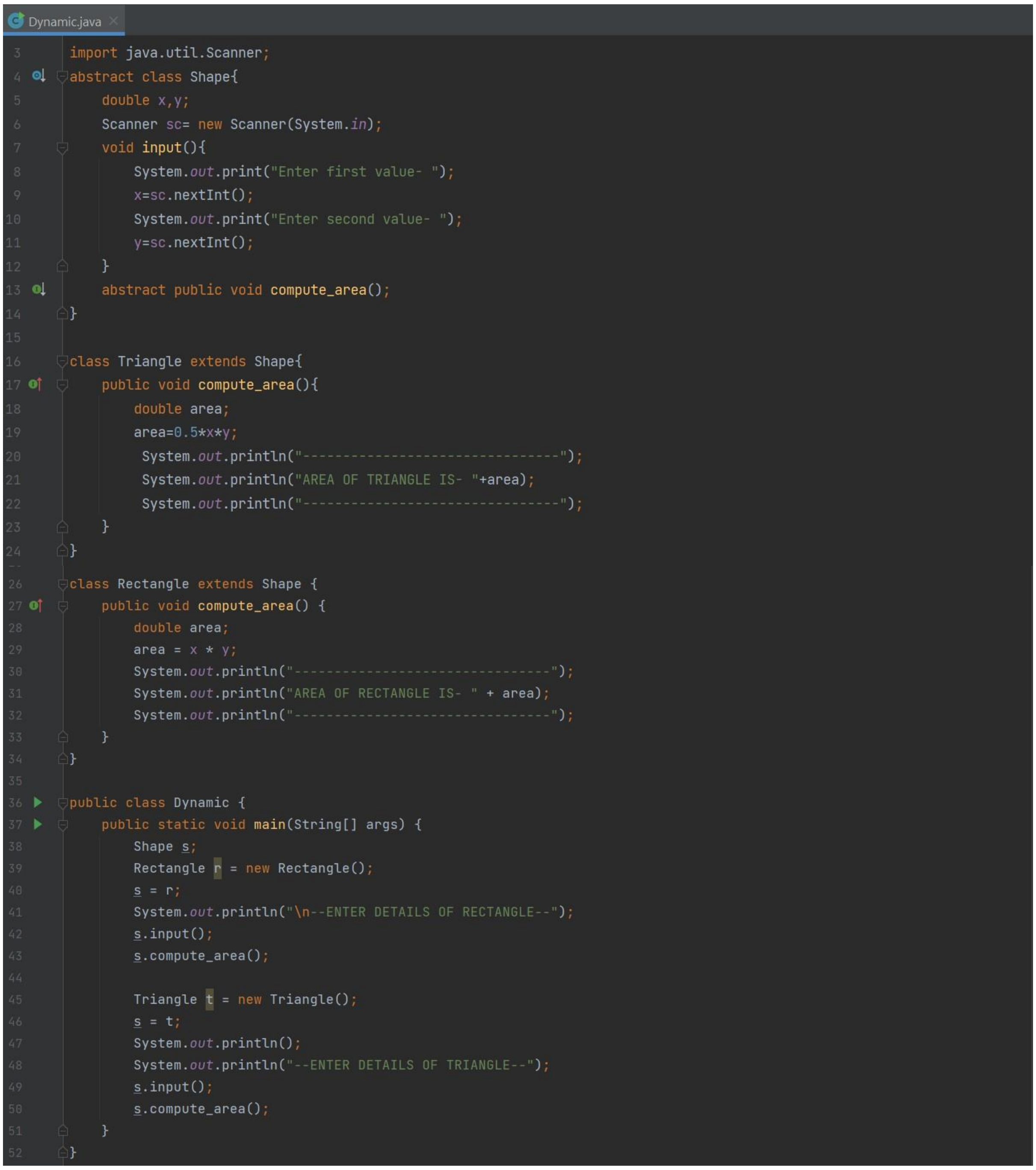
```
import java.util.Scanner; abstract
class Shape{
   double x,y;
   Scanner sc= new Scanner(System.in);
void input(){
      System.out.print("Enter first value- ");
x=sc.nextInt();
      System.out.print("Enter second value- ");
      y=sc.nextInt();
   }
   abstract public void compute_area();
}

class Triangle extends Shape{
public void compute_area(){
      double area;
area=0.5*x*y;
       System.out.println("--------------------------------");
       System.out.println("AREA OF TRIANGLE IS- "+area);
       System.out.println("--------------------------------");
   }
}

class Rectangle extends Shape {
public void compute_area() {
      double area;
area = x * y;
      System.out.println("--------------------------------");
      System.out.println("AREA OF RECTANGLE IS- " + area);
      System.out.println("--------------------------------");
   }
}

public class Dynamic {
   public static void main(String[] args) {
      Shape s;
      Rectangle r = new Rectangle();
      s = r;
      System.out.println("\n--ENTER DETAILS OF RECTANGLE--");
      s.input();
      s.compute_area();

      Triangle t = new Triangle();
s = t;
      System.out.println();
      System.out.println("--ENTER DETAILS OF TRIANGLE--");
      s.input();
      s.compute_area();
   }
}
```


Dynamic.java

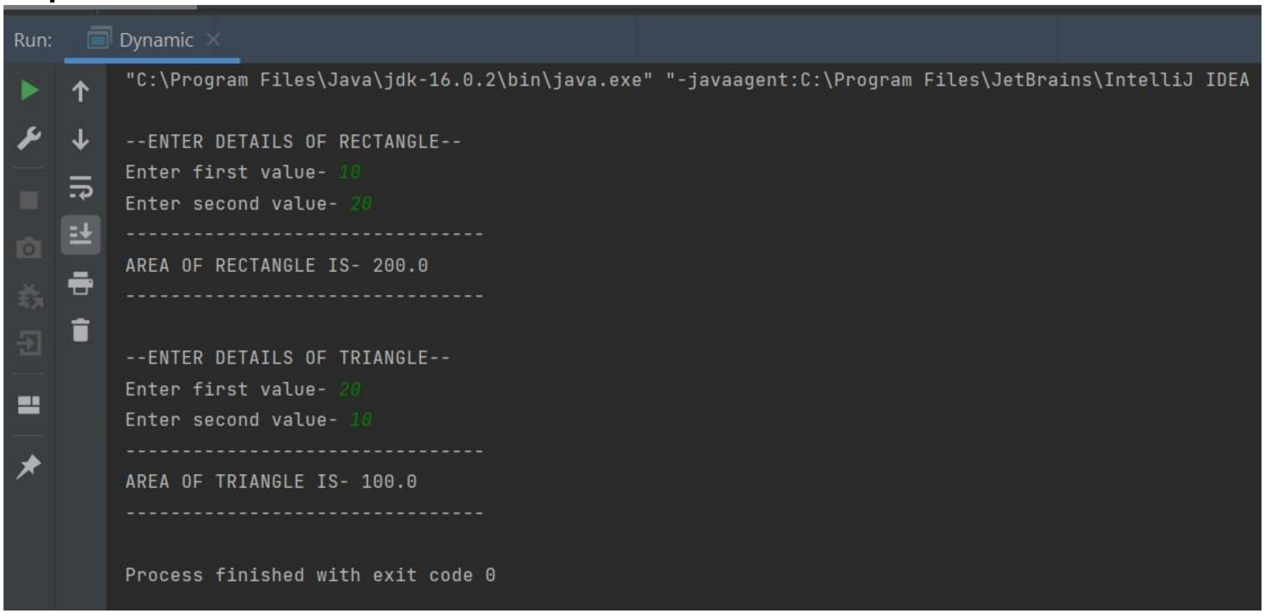
```java
import java.util.Scanner;
abstract class Shape{
    double x,y;
    Scanner sc= new Scanner(System.in);
    void input(){
        System.out.print("Enter first value- ");
        x=sc.nextInt();
        System.out.print("Enter second value- ");
        y=sc.nextInt();
    }
    abstract public void compute_area();
}

class Triangle extends Shape{
    public void compute_area(){
        double area;
        area=0.5*x*y;
         System.out.println("--------------------------------");
         System.out.println("AREA OF TRIANGLE IS- "+area);
         System.out.println("--------------------------------");
    }
}
class Rectangle extends Shape {
    public void compute_area() {
        double area;
        area = x * y;
        System.out.println("--------------------------------");
        System.out.println("AREA OF RECTANGLE IS- " + area);
        System.out.println("--------------------------------");
    }
}

public class Dynamic {
    public static void main(String[] args) {
        Shape s;
        Rectangle r = new Rectangle();
        s = r;
        System.out.println("\n--ENTER DETAILS OF RECTANGLE--");
        s.input();
        s.compute_area();

        Triangle t = new Triangle();
        s = t;
        System.out.println();
        System.out.println("--ENTER DETAILS OF TRIANGLE--");
        s.input();
        s.compute_area();
    }
}
```

Output-



```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA

--ENTER DETAILS OF RECTANGLE--
Enter first value- 10
Enter second value- 20
--------------------------------
AREA OF RECTANGLE IS- 200.0
--------------------------------

--ENTER DETAILS OF TRIANGLE--
Enter first value- 20
Enter second value- 10
--------------------------------
AREA OF TRIANGLE IS- 100.0
--------------------------------

Process finished with exit code 0
```

Conclusion- Thus studied the concept of Polymorphism and Abstraction using java.