Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.

OOPM / Odd Sem 2021-22 / Experiment 5.

YASH SARANG.
47 / D6AD.

EXPERIMENT - 5.

AIM: a. Consider a class Figure and overload the function called area () to display the area of figures like square, triangle, rectangle, and circle.

b. Write a Program for Complex Number addition using Constructor.

THEORY: Explain following concepts in detail

1. Constructors
2. Access
3. Specifiers
4. Abstract Classes
5. Wrapper Classes
6. Inheritance
7. Polymorphism
8. Method Overriding
9. Keyword-Static, final, Super and this.

Aim: a. Consider a class figure and overload the function called area to display area of figures like square, triangle, circle, rectangle.

b. Write a program for Complex No. add^{x} using Constructor.

Theory: • Constructor: is a special type of method that is used to initialize the object. It is invoked at the time of object creation. It constructs the value, i.e provides data for object. These are of 2 types.

1. Default: provides default values like 0, null.
2. Parameterized: provides different values to distinct objects.

• Access specifiers: 1. Public: Class, methods, variables and constructors can be accessed from other class.

2. Protected: methods, etc are declared protected in a superclass can be accessed only by subclasses.
3. Private: can only be accessed within the declared class.
4. Default: No modifier required. Access class, method, variables in same package but not from outside.

Abstract Class: Class declared as abstract. It needs to be extended and its abstract method implemented. It cannot be initiated, cannot have non abstract methods.

Syntax - abstract class <class name>{}

- Wrapper Class : Provides mechanism to convert primitive into object and vice-versa. Uses are: Change the value in method, serialization, synchronization.

- Inheritence : Inheritence is a mechanism in ~~the~~ which one object acquires all properties and behaviors of parent object. It represents the IS-A relationship.

  Uses - Method overriding, Code Reusability.
  Syntax - class Subclass-name extends Superclass-name
  { //methods & fields }

- Polymorphism: Concept by which we can perform a single action in different ways. Types are compile-time & run-time. If you overload static method in Java, it is an example of compile-time.

- Method Overriding: If subclass provides specific implementation of method, ie already used by parent class. Used for runtime polymorphism, must have same name and parameter as in parent class.

- Keyword 1. Static : used in Java mainly for memory management. We may apply it with variables, methods, blocks, nested class.

2. This - can be used to refer current class instance variable.
  - this () can be used to invoke current class constructor.
  - can be passed as an argument in method call.

3. Super - The super is a reference variable that is used to refer immediate present class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly, i.e referred by the superclass reference variable.

4. Final: Used to restrict the user. Can be used in many contexts.
  final can be : Variable / Method / Class.

Conclusion : We have studied and learned about constructors, different types of class, different keywords - static, super, etc. We also have learnt method overriding, inheritence, implementation and use of super and sub-classes, access specifiers. We have implemented programs using constructor and function overloading.

Program 1: Consider a class Figure and overload the function called area () to display the area of figures like square, triangle, rectangle, and circle.

```
import java.util.Scanner;
class Figure
{
   public int area(int s)
   {
      // Area of square
      return s*s;
   }

   public double area(double h, double b)
   {
      // Area of triangle
      return 0.5*h*b;
   }

   public int area(int l,int b)
   {
      // Area of Rectangle
      return l*b;
   }

   public double area(double r)
   {
      // Area of Circle
      return 3.14*r*r;
   }

   public static void main(String[] args)
   {

     Figure a = new Figure();

     System.out.println("Area of Square: "+a.area(4));
     System.out.println("Area of Circle: "+a.area(5.0));
     System.out.println("Area of Triangle: "+a.area(3.0,4.0));
     System.out.println("Area of Rectangle: "+a.area(4,5));
   }
}
```

The output of program 1:

```
C:\Users\admin\Desktop\Java>java Figure.java
Area of Square: 16
Area of Circle: 78.5
Area of Triangle: 6.0
Area of Rectangle: 20
```

Program 2: Write a Program for Complex Number addition using Constructor.

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

class Complex
{
	double real,imaginary;

	Complex(double real,double imaginary)
	{
		this.real = real;
		this.imaginary = imaginary;
	}

	void display()
	{
		System.out.println(real + " + " + imaginary + " i");
	}

	static Complex sum(Complex c1,Complex c2)
	{
		Complex addition = new Complex(0,0);
		addition.real = c1.real + c2.real;
		addition.imaginary = c1.imaginary + c2.imaginary;
		return addition;
	}

	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter Real & Img. part of 1st no. : ");
		double a = sc.nextInt();
		double b = sc.nextInt();
		Complex c1 = new Complex(a,b);
		System.out.print("Enter Real & Img. part of 2nd no. : ");
		double c = sc.nextInt();
		double d =sc.nextInt();
		Complex c2 = new Complex(c,d);
		c1.display();
		c2.display();
```

```
        Complex addition = sum(c1,c2);
        System.out.println("The sum is: "+ addition.real+" + "+ addition.imaginary+"i");
    }
}
```

The output of program 2:

```
C:\Users\admin\Desktop\Java>java Complex.java
Enter Real & Img. part of 1st no. : 1 5
Enter Real & Img. part of 2nd no. : 2 7
1.0 + 5.0 i
2.0 + 7.0 i
The sum is: 3.0 + 12.0i
```
