College of Engineering, Thiruvananthapuram

Object-Oriented Programming Lab

Anirudh A. V.

S2 CSE R2, Roll No. 11

Department of Computer Science
& Engineering

January 25, 2022

1 Method Overloading

1.1 Aim

To write a Java program to calculate the area of different shapes namely circle, rectangle, and triangle using the concept of method overloading.

1.2 Algorithm

1. Start
2. Create a class Area and initialize all the variables.
3. Declare the overloaded function area and its various forms.
4. Declare the main function and create an object for Area.
5. Declare a Scanner object to accept user input.
6. Display the Menu to the user.
    a. For option 1 go to STEP 7.
    b. For option 1 go to STEP 11.
    c. For option 1 go to STEP 15.
    d. For option 1 go to STEP 19.
7. Accept the length and breadth.
8. Call the area method of the class and pass the length and breadth as the parameter.
9. Print the area.
10. Go to STEP 6.
11. Accept the radius from the user.
12. Call the area method of the class and pass the radius as the parameter.
13. Print the area.
14. Go to STEP 6.
15. Accept the length of the three sides from the user.
16. Call the area method of the class and pass the three lengths as the parameters.
17. Print the area.
18. Go to STEP 6.
19. Stop

1.3 Code

```
/**
 * This program is intended to calculate the are of different shapes
 * using method overloading.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

import java.util.Scanner;

public class Area {

    static double area(double radius) {
        return Math.PI * Math.pow(radius, 2);
    }

    static double area(double length, double breadth) {
        return length * breadth;
    }

    static double area(double a, double b, double c) {
        double s = (a + b + c) / 2;
        double x = s * (s - a) * (s - b) * (s - c);
        return Math.sqrt(x);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int response = 1, choice;
        double radius, a, b, c, length, breadth;
        do {
            System.out.println("\t\nM E N U");
            System.out.println("1. Rectangle\n2. Circle\n3. Triangle\n4.
Exit");
            System.out.printf("\n\t->");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Enter the length and breadth : ");
                    length = sc.nextDouble();
                    breadth = sc.nextDouble();
                    System.out.println("Area : " + area(length, breadth)+" Sq.
unit");
                    break;

                case 2:
                    System.out.println("Enter the radius : ");
                    radius = sc.nextDouble();
```

```
                    System.out.println("Area : " + area(radius)+" Sq. unit");
                    break;

                case 3:
                    System.out.println("Enter the sides a, b, c of the
triangle : ");
                    a = sc.nextDouble();
                    b = sc.nextDouble();
                    c = sc.nextDouble();
                    System.out.println("Area : " + area(a, b, c)+" Sq. unit");
                    break;

                case 4:
                    System.exit(0);
                    break;

                default:
                    break;
            }
        } while (response == 1);
        sc.close();
    }
}
```

1.4 Sample Output

```
M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->1
Enter the length and breadth :
12 10
Area : 120.0 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->2
Enter the radius :
1
Area : 3.141592653589793 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->3
Enter the sides a, b, c of the triangle :
3 4 5
Area : 6.0 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->4
```

2 Inheritance

2.1 Aim

To write a Java program that creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary()' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherit the employee class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign a name, age, phone number, address, and salary to an officer and a manager by making an object of both of these classes and printing the same.

2.2 Algorithm

1. Start
2. Create the class Employee and declare all the member variables and member functions.
3. Create the class Officer that inherits the class Employee.
4. Declare the member variable and all the member functions.
5. Create the class Manager that inherits the class Employee.
6. Declare the member variable and all the member functions.
7. Declare all the main functions.
8. Declare objects of the classes.
9. Call the functions that demonstrate the properties of inheritance
10. Stop

2.3 Code

```
/**
 * This program is intended to understand the concept of Inheritance.
 * It takes input for classes Officer and Employee, which are the child
 * of the class Employee. Then prints the entered data.
 * This program is written by Anirudh A V on Jan 25 2022.
 */
import java.util.Scanner;

class Employee {
    String name;
    int age;
    int ph;
    double salary;
    String adr;

    void printSalary() {
        System.out.println("\nThe salary of the employee is " + salary);
    }
}

class Officer extends Employee {
    String specialization;

    void newOfficer() {
        Scanner sc = new Scanner(System.in);
        System.out.printf("\nPlease enter the name of the officer - ");
        this.name = sc.nextLine();
        System.out.printf("\nPlease enter the age of the officer - ");
        this.age = sc.nextInt();
        System.out.printf("\nPlease enter the phone number of the officer -
");
        this.ph = sc.nextInt();
        System.out.printf("\nPlease enter the salary of the officer - ");
        this.salary = sc.nextDouble();
        sc.nextLine();
        System.out.printf("\nPlease enter the address of the officer - ");
        this.adr = sc.nextLine();
        System.out.printf("\nPlease enter the specialization of the officer -
");
        this.specialization = sc.nextLine();
        sc.close();
    }

    void printData() {
        System.out.println("\nOfficer name is " + name);
        System.out.println("Officer age is " + age);
        System.out.println("Officer phone number is " + ph);
```

```
        printSalary();
        System.out.println("Officer address is " + adr);
        System.out.println("Officer specialization is " + specialization);
    }
}

class Manager extends Employee {
    String department;

    void newManager() {
        Scanner sc = new Scanner(System.in);
        System.out.printf("\nPlease enter the name of the employee - ");
        this.name = sc.nextLine();
        System.out.printf("\nPlease enter the age of the employee - ");
        this.age = sc.nextInt();
        System.out.println("\nPlease enter the phone number of the employee -
");
        this.ph = sc.nextInt();
        System.out.printf("\nPlease enter the salary of the employee - ");
        this.salary = sc.nextDouble();
        sc.nextLine();
        System.out.printf("\nPlease enter the address of the employee - ");
        this.adr = sc.nextLine();
        System.out.printf("\nPlease enter the department of the employee - ");
        this.department = sc.nextLine();
        sc.close();
    }

    void printData() {
        System.out.println("\nManager name is " + name);
        System.out.println("Manager age is " + age);
        System.out.println("Manager phone number is " + ph);
        printSalary();
        System.out.println("Manager address is " + adr);
        System.out.println("Manager department is " + department);
    }
}

public class Office {
    public static void main(String args[]) {
        Manager m1 = new Manager();
        Officer off = new Officer();
        off.newOfficer();
        m1.newManager();
        off.printData();
        m1.printData();
    }
}
```

2.4 Sample Output

```
Please enter the name of the officer - dasnsioam

Please enter the age of the officer - 122

Please enter the phone number of the officer - 231442

Please enter the salary of the officer - 21421

Please enter the address of the officer - kddmkl, fdndas

Please enter the specialization of the officer - dfsnodf

Please enter the name of the employee - Anir

Please enter the age of the employee - 456

Please enter the phone number of the employee -
721974

Please enter the salary of the employee - 347232

Please enter the address of the employee - bdfsb, dfsa

Please enter the department of the employee - dashbuf

Officer name is dasnsioam
Officer age is 122
Officer phone number is 231442

The salary of the employee is 21421.0
Officer address is kddmkl, fdndas
Officer specialization is dfsnodf

Manager name is Anir
Manager age is 456
Manager phone number is 721974

The salary of the employee is 347232.0
Manager address is bdfsb, dfsa
Manager department is dashbuf
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>
```

3 Method Overriding

3.1 Aim

To Write two Java classes, Employee and Engineer. The engineer should inherit from the Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer's salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed).

- display () only prints the name of the class and does not return any value. Ex. "Name of class is Employee."
- calcSalary() in Employee displays "Salary of employee is 10000" and calcSalary() in Engineer displays "Salary of employee is 20000."

3.2 Algorithm

1. Start
2. Create the Employee class and declare all the variables and the member

functions.

3. Create the Engineer class that inherits from Employee.
4. Declare the member variables and the member functions (Redeclare

calcSalary()).

5. Declare the main function and create an object of Engineer.
6. The function overriding takes over and the function in Engineer is called.
7. Stop.

3.3 Code

```
/**
 * This program is intended to demonstrate the concept of method overriding.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

class Employee {
    void display() {
        System.out.println("Inside Employee");
        return;
    }

    void calcSalary() {
        System.out.println("The salary of the employee is 10000");
        return;
    }
}

class Engineer extends Employee {
    void display() {
        System.out.println("Inside Engineer");
        return;
    }

    void calcSalary() {
        System.out.println("The salary of the employee is 20000");
        // Redirects to display of the parent class
        super.display();
        // Redirects to calcSalary() of the parent class
        super.calcSalary();
        return;
    }
}

public class EmployeeEngineer {
    public static void main(String args[]) {
        Engineer eng = new Engineer();
        eng.display();
        eng.calcSalary();
    }
}
```

3.4 Sample Output

```
        at Office.main(Office.java:89)
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>  e:; cd 'e:\Anirudh\Anirudh\CET\SEM 3\O
1\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Java_cycle_2_b4e8e79d\bin' 'EmployeeEngineer'
Inside Engineer
The salary of the employee is 20000
Inside Employee
The salary of the employee is 10000
```

4 Polymorphism

4.1 Aim

To write a java program to create an abstract class named Shape that contains an empty method named numberOfSides(). Provide three classes named Rectangle, Triangle, and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides() that shows the number of sides in the given geometrical structures.

4.2 Algorithm

1. Start
2. Declare an abstract class and create all the variables.
3. Declare the class Rectangle and make it inherit from the abstract class ‘Shape’.
4. Declare the methods in Rectangle.
5. Declare the class Triangle and make it inherit from the abstract class ‘Shape’.
6. Declare the methods in Triangle.
7. Declare the class Hexagon and make it inherit from the abstract class ‘Shape’.
8. Declare the methods in Hexagon.
9. Stop

4.3 Code

```java
/**
 * This program is intended to understand the concept of polymorphism in java.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

abstract class Shape {
    abstract void numberOfSides();
}

class Rectangles extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a rectangle is 4");
    }
}

class Triangles extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a triangle is 3");
    }
}

class Hexagons extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a hexagon is 6");
    }
}

public class Abstract {
    public static void main(String args[]) {
        Rectangles rt = new Rectangles();
        Triangles tr = new Triangles();
        Hexagons hx = new Hexagons();
        rt.numberOfSides();
        tr.numberOfSides();
        hx.numberOfSides();
    }
}
```

4.4 Sample Output

```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>  e:; cd 'e:\Anirudh\Anirudh\
1\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Java_cycle_2_b4e8e79d\bin' 'Abstract'
The number of sides of a rectangle is 4
The number of sides of a triangle is 3
The number of sides of a hexagon is 6
```

5 Interface Inheritance

5.1 Aim

To write a Java program to illustrate Interface inheritance.

5.2 Algorithm

1. Start
2. Initialize the first interface ShapeName.
3. Layout definitions for all the methods in ShapeName.
4. Create another interface in ShapeInfo that inherits ShapeName.
5. Layout definitions for all the methods in ShapeName.
6. Create a class that inherits from ShapeInfo and declare functions inside it.
7. Layout the definitions for the functions in the interfaces in the class definition.
8. Declare the main function and create an object of the class.
9. Call the member function of the class and interface inheritance is observed.
10. Stop

5.3 Code

```
/**
 * This program is intended to demonstrate the concept of interface
inheritance.
 * This program is written by Anirudh A V on Jan 25, 2022.
 */

interface ShapeName {
    public void name();
}

interface ShapeInfo extends ShapeName {
    public void info();
}

class Triangle implements ShapeInfo{
    // Note: If the interface class is public, then this implementation should
also
    //be public
    public void name(){
    System.out.println("\nInside Triangle");
    }
    public void info(){
    System.out.println("A triangle has 3 sides\n");
    }
    }

public class InterfaceInheritance {
    public static void main(String args[]) {
        Triangle tri = new Triangle();
        tri.name();
        tri.info();
    }
}
```

5.4 Sample Output

```
The number of sides of a hexagon is 6
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cyc
1\bin\java.exe' '--enable-preview' '-XX:+ShowCod
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Jav

Inside Triangle
A triangle has 3 sides
```

6 Garbage Collection

6.1 Aim

To write a Java program to demonstrate the use of garbage collectors.

6.2 Algorithm

1. Start
2. Define the garbage_collector class and declare the finalize() function inside it.
3. Print the message “Garbage collector has run”.
4. Declare the main function and create an object of the class in it.
5. Call the display() method in the class.
6. Assign the object to null.
7. The garbage collector then activates and the finalizer() method in the class is called.
8. Stop

6.3 Code

```
/**
 * This program is intended to demonstrate the concept of garbage collection.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

public class garbage_collector {
    // This function is executed whenever garbage collection happens. It has
to
    // be declared public so that it can be called outside the class
    protected void finalize() throws Throwable {
        System.out.println("Garbage collector has run");
    }

    public static void main(String args[]) {
        garbage_collector gb = new garbage_collector();
        gb = null;
        // Calling garbage collector
        System.gc();
    }
}
```

6.4 Sample Output