

## Green University of Bangladesh

**Department of Computer Science and Engineering (CSE) Faculty of Sciences and Engineering**

**Semester: (Spring, 2023), B.Sc. in CSE (Day)**

**LAB REPORT NO: 06**

**Course Title: Object Oriented Programing Lab**

**Course Code: CSE 202 Section: DE**

**Student Details**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Md. Moshiur Rahman | 221902324 |

Lab Date : 26/04/2023

Submission Date : 30/04/2023

**Course Teacher’s Name** : **Dr. Muhammad Aminur Rahaman**

**[For Teachers use only: Don’t Write Anything inside this box]**

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature:..................... Comments:.............................................. Date:..............................** |

##### 1. TITLE OF THE LAB EXPERIMENT:

➤ Write code to demonstrate both method overriding and overloading.

**2. OBJECTIVES**

➤ Understanding Polymorphism in Abstract classes

➤ Method overloading, Method overriding

##### 3. ALGORITHM

Step-1 : Start

Step-2 : Define a Shape class with two methods: draw and draw(String color)

Step-3 : Implement the draw method to print "Drawing a shape" and the draw(String color) method to print "Drawing a {color} shape"

Step-4 : Define a Circle class that extends Shape

Step-5 : Override the draw method in Circle to print "Drawing a circle"

Step-6 : Define a new method in Circle called draw(int diameter) that prints "Drawing a circle with diameter {diameter}"

Step-7 : In the main method, create an instance of Shape and an instance of Circle

Step-8 : Call the draw and draw(String color) methods on the Shape instance

Step-9 : Call the draw and draw(int diameter) methods on the Circle instance

Step-10 : End.

**4. IMPLEMENTATION**

class Shape {

public void draw() {

System.out.println("Drawing a shape");

}

public void draw(String color) {

System.out.println("Drawing a " + color + " shape");

}

}

class Circle extends Shape {

@Override

public void draw() {

System.out.println("Drawing a circle");

}

public void draw(int diameter) {

System.out.println("Drawing a circle with diameter " + diameter);

}

}

public class Main {

public static void main(String[] args) {

Shape shape = new Shape();

shape.draw();

shape.draw("red");

Circle circle = new Circle();

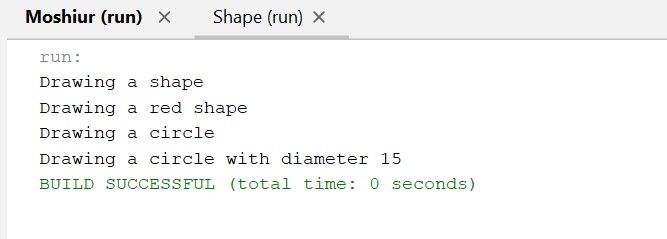
circle.draw();

circle.draw(15);

}

}

**5. TEST RESULT**



**6. ANALYSIS & DISCUSSION**

The Java code I provided demonstrates the concepts of method overriding and overloading in object-oriented programming.

Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. In the code, the Circle class overrides the draw method of the Shape class to provide a specific implementation that draws a circle.

Method overloading occurs when a class has multiple methods with the same name, but different parameters. In the code, the Circle class has two methods named draw, one with no parameters and one with an integer parameter, which demonstrates method overloading.

In the main method, I created an instance of the Shape class and called its draw and draw(String color) methods. Then, I created an instance of the Circle class and called its draw and draw(int diameter) methods. The output shows the different messages printed by each method call.

This code demonstrates the power and flexibility of object-oriented programming, allowing us to define classes and methods that can be easily reused and extended. By using inheritance, subclasses can provide their own specific implementations of methods, while method overloading allows classes to provide multiple methods with the same name, but different functionality.

**7. SUMMARY**

In summary, the given Java code demonstrates the concepts of method overriding and overloading in object-oriented programming. The Circle class overrides the draw method of the Shape class to provide a specific implementation that draws a circle, as well as introducing overloading methods with a new method draw(int diameter) that takes an integer parameter. In the main method, instances of both shapes and circles are created and different methods are called to display different outputs. This code shows the power and flexibility of object-oriented programming in creating reusable and extensible code.