# PRACTICAL 8

**Aim**: To understand concept of overriding and implement different type of Association relationship

**Prerequisite:**

* Knowledge of basic constructs in Java programming language.

**Outcome:** After successful completion of this experiment students will be able to,

* Invoke the superclass’s constructors and methods using the super keyword and also to override instance methods in the subclass
* Design and use abstract classes
* Specify common behaviour for objects using interfaces
* Define interfaces and define classes that implement interfaces
* To explore the similarities and differences between an abstract class and an interface

**Theory:**

**Method Overriding:**

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

**Usage Method Overriding**

* Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
* Method overriding is used for runtime polymorphism

**Rules for Java Method Overriding**

* The method must have the same name as in the parent class
* The method must have the same parameter as in the parent class.
* There must be an IS-A relationship (inheritance).

**Super Keyword**

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

**Usage of super Keyword**

* super can be used to refer immediate parent class instance variable.
* super can be used to invoke immediate parent class method.
* super() can be used to invoke immediate parent class constructor.

**Association Relationships:**

Association establishes relationship between two separate classes through their objects. The relationship can be one to one, One to many, many to one and many to many. Composition and aggregation are two types of association.

[Aggregation](https://beginnersbook.com/2013/05/aggregation/) is a special form of association which is a unidirectional one way relationship between classes (or entities), for e.g. Wallet and Money classes. Wallet has Money but money doesn’t need to have Wallet necessarily so its a one directional relationship. In this relationship both the entries can survive if other one ends. In our example if Wallet class is not present, it does not mean that the Money class cannot exist.

Composition is a restricted form of Aggregation in which two entities (or you can say classes) are highly dependent on each other. For e.g. Human and Heart. A human needs heart to live and a heart needs a Human body to survive. In other words when the classes (entities) are dependent on each other and their life span are same (if one dies then another one too) then its a composition. Heart class has no sense if Human class is not present.

**Example of Composition:**

//Car must have Engine

public class Car {

   //engine is a mandatory part of the car

   private final Engine engine;

   public Car () {

      engine = new Engine();

   }

}

//Engine Object

class Engine {}

**Example of Aggregation**

//Team

public class Team {

   //players can be 0 or more

   private List players;

   public Car () {

      players = new ArrayList();

   }

}

//Player Object

class Player {}

(TO BE COMPLETED BY STUDENTS)

|  |  |
| --- | --- |
| Roll No. K005 | Name: Jal Bafana |
| Class: Btech. Cyber Security (Sem-4) | Batch: K1 |
| Date of Practical: 24.03.2025 | Date of Submission: 24.03.2025 |
| Grade: |  |

1. Design and implement classes as shown in the Class diagram below. Write a test program to test the classes created.



package k005\_lab8;

class Shape

{

private String color = "red";

private boolean filled = true;

public Shape()

{

}

public Shape(String color, boolean filled)

{

this.color = color;

this.filled = filled;

}

public String getColor()

{

return color;

}

public void setColor(String color)

{

this.color = color;

}

public boolean isFilled()

{

return filled;

}

public void setFilled(boolean filled)

{

this.filled = filled;

}

@Override

public String toString()

{

return "Shape[color=" + color + ", filled=" + filled + "]";

}

}

class Circle extends Shape

{

private double radius = 1.0;

public Circle()

{

}

public Circle(double radius)

{

this.radius = radius;

}

public Circle(double radius, String color, boolean filled)

{

super(color, filled);

this.radius = radius;

}

public double getRadius()

{

return radius;

}

public void setRadius(double radius)

{

this.radius = radius;

}

public double getArea()

{

return Math.PI \* radius \* radius;

}

public double getPerimeter()

{

return 2 \* Math.PI \* radius;

}

@Override

public String toString()

{

return "Circle[radius=" + radius + ", " + super.toString() + "]";

}

}

class Rectangle extends Shape

{

private double width = 1.0;

private double length = 1.0;

public Rectangle()

{

}

public Rectangle(double width, double length)

{

this.width = width;

this.length = length;

}

public Rectangle(double width, double length, String color, boolean filled)

{

super(color, filled);

this.width = width;

this.length = length;

}

public double getWidth()

{

return width;

}

public void setWidth(double width)

{

this.width = width;

}

public double getLength()

{

return length;

}

public void setLength(double length)

{

this.length = length;

}

public double getArea()

{

return width \* length;

}

public double getPerimeter()

{

return 2 \* (width + length);

}

@Override

public String toString()

{

return "Rectangle[width=" + width + ", length=" + length + ", " + super.toString() + "]";

}

}

class Square extends Rectangle

{

public Square()

{

}

public Square(double side)

{

super(side, side);

}

public Square(double side, String color, boolean filled)

{

super(side, side, color, filled);

}

public double getSide()

{

return getWidth();

}

public void setSide(double side)

{

setWidth(side);

setLength(side);

}

@Override

public void setWidth(double side)

{

super.setWidth(side);

super.setLength(side);

}

@Override

public void setLength(double side)

{

super.setWidth(side);

super.setLength(side);

}

@Override

public String toString()

{

return "Square[side=" + getWidth() + ", " + super.toString() + "]";

}

}

public class Pgm1

{

public static void main(String[] args)

{

Shape shape = new Shape("blue", false);

System.out.println(shape);

Circle circle = new Circle(2.5, "green", true);

System.out.println(circle);

System.out.println("Circle Area: " + circle.getArea());

System.out.println("Circle Perimeter: " + circle.getPerimeter());

Rectangle rectangle = new Rectangle(4.0, 5.0, "yellow", false);

System.out.println(rectangle);

System.out.println("Rectangle Area: " + rectangle.getArea());

System.out.println("Rectangle Perimeter: " + rectangle.getPerimeter());

Square square = new Square(3.0, "purple", true);

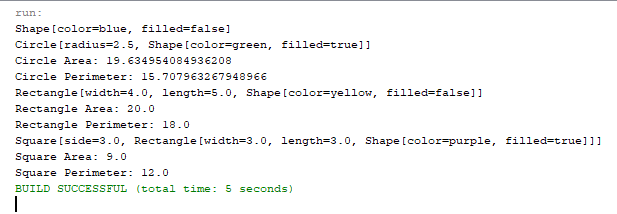
System.out.println(square);

System.out.println("Square Area: " + square.getArea());

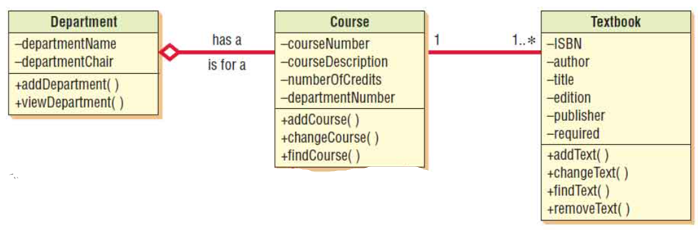
System.out.println("Square Perimeter: " + square.getPerimeter());

}

}



1. Implement the class diagram given below.



package k005\_lab8;

import java.util.ArrayList;

import java.util.List;

class Department

{

private String departmentName;

private String departmentChair;

public Department(String departmentName, String departmentChair)

{

this.departmentName = departmentName;

this.departmentChair = departmentChair;

}

public void addDepartment(String name, String chair)

{

this.departmentName = name;

this.departmentChair = chair;

}

public void viewDepartment() {

System.out.println("Department Name: " + departmentName);

System.out.println("Department Chair: " + departmentChair);

}

}

class Course {

private String courseNumber;

private String courseDescription;

private int numberOfCredits;

private String departmentNumber;

private List<Textbook> textbooks;

public Course(String courseNumber, String courseDescription, int numberOfCredits, String departmentNumber)

{

this.courseNumber = courseNumber;

this.courseDescription = courseDescription;

this.numberOfCredits = numberOfCredits;

this.departmentNumber = departmentNumber;

this.textbooks = new ArrayList<>();

}

public void addCourse(String number, String description, int credits, String department)

{

this.courseNumber = number;

this.courseDescription = description;

this.numberOfCredits = credits;

this.departmentNumber = department;

}

public void changeCourse(String newDescription, int newCredits)

{

this.courseDescription = newDescription;

this.numberOfCredits = newCredits;

}

public void findCourse()

{

System.out.println("Course Number: " + courseNumber);

System.out.println("Course Description: " + courseDescription);

System.out.println("Credits: " + numberOfCredits);

System.out.println("Department Number: " + departmentNumber);

}

public void addTextbook(Textbook textbook)

{

textbooks.add(textbook);

}

public void listTextbooks()

{

for (Textbook textbook : textbooks)

{

System.out.println(textbook);

}

}

}

class Textbook

{

private String ISBN;

private String author;

private String title;

private String edition;

private String publisher;

private boolean required;

public Textbook(String ISBN, String author, String title, String edition, String publisher, boolean required)

{

this.ISBN = ISBN;

this.author = author;

this.title = title;

this.edition = edition;

this.publisher = publisher;

this.required = required;

}

public void addText(String title, String author, String edition, String publisher, boolean required)

{

this.title = title;

this.author = author;

this.edition = edition;

this.publisher = publisher;

this.required = required;

}

public void changeText(String newTitle, String newEdition)

{

this.title = newTitle;

this.edition = newEdition;

}

public void findText()

{

System.out.println("Title: " + title);

System.out.println("Author: " + author);

System.out.println("Edition: " + edition);

System.out.println("Publisher: " + publisher);

System.out.println("Required: " + required);

}

public void removeText()

{

this.ISBN = "";

this.author = "";

this.title = "";

this.edition = "";

this.publisher = "";

this.required = false;

}

@Override

public String toString()

{

return "Textbook[ISBN=" + ISBN + ", Author=" + author + ", Title=" + title + ", Edition=" + edition + ", Publisher=" + publisher + ", Required=" + required + "]";

}

}

public class Pgm2

{

public static void main(String[] args)

{

Department department = new Department("Computer Science", "Dr. Smith");

department.viewDepartment();

Course course = new Course("CS101", "Introduction to Programming", 3, "CS");

course.findCourse();

Textbook textbook1 = new Textbook("123456", "John Doe", "Java Programming", "1st", "TechBooks", true);

Textbook textbook2 = new Textbook("789012", "Jane Doe", "Data Structures", "2nd", "TechBooks", false);

course.addTextbook(textbook1);

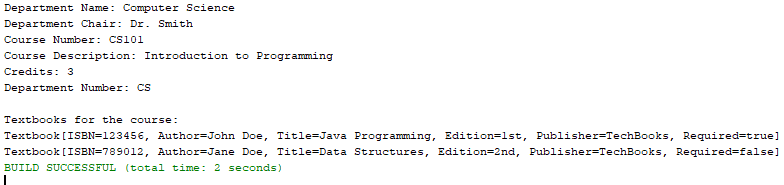
course.addTextbook(textbook2);

System.out.println("\nTextbooks for the course:");

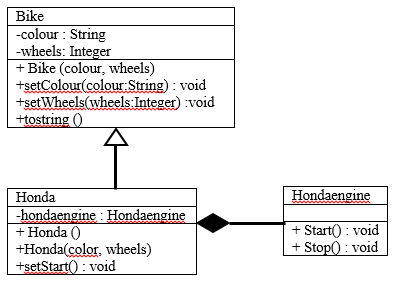
course.listTextbooks();

}

}



1. Implement the class diagram given below.



package k005\_lab8;

class HondaEngine

{

public void Start()

{

System.out.println("Honda engine starting...");

}

public void Stop()

{

System.out.println("Honda engine stopping...");

}

}

class Bike

{

private String colour;

private int wheels;

public Bike(String colour, int wheels)

{

this.colour = colour;

this.wheels = wheels;

}

public void setColour(String colour)

{

this.colour = colour;

}

public void setWheels(int wheels)

{

this.wheels = wheels;

}

@Override

public String toString()

{

return "Bike[colour=" + colour + ", wheels=" + wheels + "]";

}

}

class Honda extends Bike

{

private HondaEngine hondaEngine;

public Honda()

{

super("Default Colour", 2);

this.hondaEngine = new HondaEngine();

}

public Honda(String colour, int wheels)

{

super(colour, wheels);

this.hondaEngine = new HondaEngine();

}

public void setStart()

{

hondaEngine.Start();

}

}

public class Pgm3

{

public static void main(String[] args)

{

Honda myHonda = new Honda("Red", 2);

System.out.println(myHonda);

myHonda.setStart();

}

}

