HYDERABAD INSTITUTE OF ARTS, SCIENCE, AND TECHNOLOGY

Object Oriented Programming lab-11

Instructor: Miss Ayesha Eman

Date: 06/10/2025

Title:

Inheritance and polymorphism

Objectives

- To understand the concept of inheritance in Java.
- To explore different types of inheritance (single, multilevel, hierarchical).
- To use the 'extends' and 'super' keywords effectively.
- To demonstrate method overriding and polymorphism.

Definition

Inheritance is an important feature of Object-Oriented Programming that allows one class to acquire the properties and behaviors of another class. The class that inherits is known as the subclass (child class), and the class being inherited from is known as the superclass (parent class). It promotes code reusability and improves program structure.

The 'extends' keyword in Java is used to establish inheritance between two classes. The 'super' keyword allows a subclass to call its parent class constructor or access parent class methods and variables.

Polymorphism

Polymorphism is one of the most powerful features of Object-Oriented Programming. It allows a single interface to represent different data types or objects. In Java, polymorphism enables methods to perform different tasks based on the object that calls them. The behavior is determined at runtime, making the code more flexible and scalable.

There are two main types of polymorphism in Java:
1. Compile-time Polymorphism (Method Overloading)
2. Runtime Polymorphism (Method Overriding)

Example of Runtime Polymorphism:

```
Shape s;
s = new Circle();
s.draw(); // Calls Circle’s draw()

s = new Rectangle();
s.draw(); // Calls Rectangle’s draw()
```

Even though the reference variable 's' is of type Shape, the actual method that gets executed depends on the object type at runtime. This is known as Dynamic Method Dispatch.

Advantages of Polymorphism:
- Improves code reusability and flexibility.
- Supports extensibility (new classes can be added easily).
- Simplifies maintenance and promotes cleaner, modular design.

Example Code (Java)

Example 1: Multilevel Inheritance

```
class Person {
    String name;
    int age;

    Person(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}

class Employee extends Person {
    double salary;

    Employee(String n, int a, double s) {
        super(n, a);
        salary = s;
    }

    void display() {
        super.display();
        System.out.println("Salary: " + salary);
    }
}

class Manager extends Employee {
    String department;

    Manager(String n, int a, double s, String d) {
        super(n, a, s);
        department = d;
    }

    void display() {
        super.display();
        System.out.println("Department: " + department);
    }

    public static void main(String[] args) {
        Manager m1 = new Manager("Laraib", 26, 120000, "AI & Research");
        m1.display();
    }
}
```

Example 2: Method Overriding & Polymorphism

```
class Shape {
    void draw() {
        System.out.println("Drawing a shape...");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle.");
    }
}

class Rectangle extends Shape {
    void draw() {
        System.out.println("Drawing a rectangle.");
    }
}

public class TestPolymorphism {
  public static void main(String[] args) {
    Shape s1 = new Circle();
    Shape s2 = new Rectangle();
    s1.draw();
    s2.draw();
  }
}
```

Output

```
Name: Laraib, Age: 26
Salary: 120000.0
Department: AI & Research

Drawing a circle.
Drawing a rectangle.
```

Lab Tasks

1. Task 1: Create a base class Person with attributes name and age. Create a subclass Student that adds rollNo and department.
   Display student details using inheritance.
3. Task 2: Write a program demonstrating method overriding and use of super keyword.
4. Task 3: Develop a program to show polymorphism by referencing subclass objects through superclass references.

Discussion Questions

1. What is inheritance, and how does it promote code reusability?
2. How is the 'super' keyword used in constructor chaining?
3. What is method overriding, and why is it useful?
4. What is polymorphism, and how does it work in inheritance?
5. Can constructors be inherited? Explain.