**1. Introduction to Polymorphism:**

* **Definition:**  
  Polymorphism in Java allows one entity (a method or object) to take many forms. It enables a single action to behave differently based on the context (i.e., different objects or inputs).
* **Example in real life:**  
  A person can be a teacher at work, a parent at home, and a friend socially. Similarly, in programming, objects and methods can behave differently in different contexts.

**2. Types of Polymorphism:**

**a. Compile-time Polymorphism (Method Overloading):**

* This type of polymorphism is resolved during compilation.
* Method overloading allows multiple methods in a class to have the same name but different parameters (either in number or type).

**Example:**

java

class Calculator {

// Method with two parameters

int add(int a, int b) {

return a + b;

}

// Overloaded method with three parameters

int add(int a, int b, int c) {

return a + b + c;

}

}

public class Main {

public static void main(String[] args) {

Calculator calc = new Calculator();

System.out.println(calc.add(5, 10)); // Calls method with two parameters

System.out.println(calc.add(5, 10, 15)); // Calls method with three parameters

}

}

**Key Points:**

* Method names are the same, but the parameter lists are different.
* This is determined at compile time.

**b. Runtime Polymorphism (Method Overriding):**

* This occurs during program execution.
* Method overriding allows a subclass to provide a specific implementation of a method that is already defined in its superclass.

**Example:**

java

class Animal {

void sound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

@Override

void sound() {

System.out.println("Dog barks");

}

}

public class Main {

public static void main(String[] args) {

Animal myAnimal = new Animal();

Animal myDog = new Dog(); // Upcasting

myAnimal.sound(); // Calls method from Animal class

myDog.sound(); // Calls method from Dog class (method overriding)

}

}

**Key Points:**

* The method in the parent class is overridden in the child class.
* The method to be executed is determined at runtime based on the object type.

**3. Benefits of Polymorphism:**

* **Code Reusability:** Common methods can be written in a parent class, while specific behaviors can be defined in child classes.
* **Flexibility:** It allows for building extensible systems, where new functionalities can be added without altering existing code.

**4. Practical Exercise:**

* **Task 1:** Create a superclass Shape with a method draw(). Then, create subclasses Circle, Square, and Triangle that override the draw() method.
* **Task 2:** Create a class Printer with overloaded methods print(int) and print(String) to understand compile-time polymorphism.

**5. Summary:**

* **Polymorphism** allows one action to be performed in different ways.
* **Compile-time polymorphism (method overloading)** resolves at compile time and is based on the method signature.
* **Runtime polymorphism (method overriding)** resolves at runtime based on the object being referred to.