**1. Introduction to Polymorphism**

* **Definition**:
  + Polymorphism in Java refers to the ability of a single entity to take multiple forms.
* **Real-life analogy**:
  + Example: A person can be a parent, an employee, and a friend at the same time.
* **Types of Polymorphism**:
  + **Compile-time Polymorphism (Method Overloading)**
  + **Run-time Polymorphism (Method Overriding)**

2. Compile-time Polymorphism (Method Overloading)

* **Concept**:
  + Same method name, different parameter lists.
* **Syntax & Code Example**:

class Calculator {

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

}

public class Test {

public static void main(String[] args) {

Calculator calc = new Calculator();

System.out.println(calc.add(10, 20)); // Outputs 30

System.out.println(calc.add(10.5, 20.5)); // Outputs 31.0

}

}

* **Key Points**:
  + Overloading is resolved at compile time.
  + Rules for Overloading (e.g., different parameter types, number of parameters).

**3. Run-time Polymorphism (Method Overriding) (30 minutes)**

* **Concept**:
  + A subclass provides a specific implementation of a method that is already defined in its superclass.
* **Syntax & Code Example**:

java

class Animal {

void sound() {

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

}

}

class Dog extends Animal {

void sound() {

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

}

}

public class Test {

public static void main(String[] args) {

Animal obj = new Dog(); // Upcasting

obj.sound(); // Outputs: Dog barks

}

}

* **Key Points**:
  + Overriding is resolved at runtime.
  + The @Override annotation ensures correctness.
  + Demonstrate upcasting.

**4. Differences Between Overloading and Overriding (10 minutes)**

* Create a comparison table:

| **Feature** | **Overloading** | **Overriding** |
| --- | --- | --- |
| Resolution Time | Compile-time | Runtime |
| Method Signature | Must differ | Must be same |
| Return Type | Can differ | Must be same |

**5. Hands-on Activity (15 minutes)**

* **Task 1**: Write a class demonstrating method overloading with a multiply method.
* **Task 2**: Create a superclass Vehicle and a subclass Car to override the move() method.

**6. Summary and Q&A (10 minutes)**

* Recap key points about polymorphism and its types.
* Answer any questions students may have.