OOP in Java

Lecture 04

Agenda Points

- Introduction to Objects as Function Arguments
- Why Pass Objects as Arguments?
- How to Pass Objects as Arguments
- Example 1: Passing Objects to Methods
- Example 2: Modifying Object Attributes via Method
- Key Points to Remember
- Use Cases for Passing Objects as Arguments

Introduction to Objects as Function Arguments

- Overview of why objects are used as arguments in methods.
- Passing objects to methods enhances flexibility and allows for complex operations on data.

Why Pass Objects as Arguments?

- **Flexibility**: Handle complex data structures efficiently.
- **Modifiability**: Methods can modify object attributes directly.
- **Reusability**: Methods can operate on various object instances, increasing code reusability.

Passing Objects to Methods

- Objects are passed by reference, not by value.
- Changes made to the object within the method affect the original object.

```
class MyClass {
// Method accepting object as an argument
void myMethod (MyObject obj) {
// Perform operations on obj
    }
}
```

Example 1: Passing Objects to Methods

Scenario: Displaying the area of a rectangle by passing the Rectangle object to a method.

```
class Rectangle {
    int length, width;

    Rectangle(int l, int w) {
        length = l;
        width = w;
    }

    int area() {
        return length * width;
    }
}

public class Main {
    static void displayArea(Rectangle rect) {
        System.out.println("Area: " + rect.area());
    }

    public static void main(String[] args) {
        Rectangle myRect = new Rectangle(10, 5);
        displayArea(myRect); // Output: Area: 50
    }
}
```

Example 2: Modifying Object Attributes via Method

Scenario: Changing the dimensions of a rectangle by passing the Rectangle object to a method.

```
class Rectangle {
    int length, width;

    Rectangle(int l, int w) {
        length = l;
        width = w;
    }

    int area() {
        return length * width;
    }
}

public class Main {

    static void modifyDimensions(Rectangle rect, int newLength, int newWidth) {
        rect.length = newLength;
        rect.width = newWidth;
    }

    public static void main(String[] args) {
        Rectangle myRect = new Rectangle(10, 5);
        System.out.println("Original Area: " + myRect.area()); // Output: 50
        modifyDimensions(myRect, 20, 10);
        System.out.println("Modified Area: " + myRect.area()); // Output: 200
    }
}
```

Key Points to Remember

- Objects are passed by reference, not by value.
- Modifications inside the method persist outside the method.
- Ideal for complex operations on objects.

Use Cases for Passing Objects as Arguments

- **Data Processing:** Passing large or complex data structures like lists, arrays, or custom objects.
- **Object Modification:** Methods that need to update or change object states.
- **Interoperability:** Allowing different methods to work on the same object instance.

Conclusion

- Passing objects as function arguments is a powerful technique in Java.
- It enables dynamic and efficient operations on complex data types.
- Enhances flexibility, modifiability, and reusability of code.

Thank you for your attention

Any question please…