Lesson 6: Constructors and Method Overloading

Marcin Kurzyna

Lecture Goals

By the end of this lesson, students should be able to:

- Understand what a **constructor** is and what it is used for.
- Create custom constructors.
- Explain and use **constructor overloading**.
- Understand and apply **method overloading**.

1 What Is a Constructor?

A **constructor** is a special method that is called when an object is created.
It is used to:

- Initialize object fields.
- Prepare an object for use.

Important characteristics:

- A constructor has the **same name as the class**.
- It has **no return type**, not even `void`.

2 Basic Constructor Example

```java
class Car {
    private String brand;
    private int year;

    // Constructor
    Car(String brand, int year) {
        this.brand = brand;
        this.year = year;
    }
}
```

The constructor is automatically called when we write:

```java
Car c = new Car("Toyota", 2020);
```

3 Default Constructor

If you do not write any constructor, Java provides a **default constructor**:

```
class Person {
    private String name;
}
// Java automatically adds:
// Person() {}
```

However, if **you define any constructor**, the default one disappears.

4 Constructor Overloading

You can have multiple constructors with different parameter lists.

```
class Point {
    private int x;
    private int y;

    // Constructor 1
    Point() {
        this.x = 0;
        this.y = 0;
    }

    // Constructor 2
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // Constructor 3
    Point(int v) {
        this.x = v;
        this.y = v;
    }
}
```

This allows flexible object creation.

5 What Is Method Overloading?

Method overloading means having multiple methods with the **same name** but different:

- number of parameters,
- types of parameters,
- order of parameters.

```
class MathUtils {
    public int add(int a, int b) {
        return a + b;
    }

    public double add(double a, double b) {
        return a + b;
    }

    public int add(int a, int b, int c) {
        return a + b + c;
    }
}
```

Overloading improves readability and avoids creating many similar method names.

6 Using `this()` to Call Constructors

You can call one constructor from another using `this()`.

```
class Rectangle {
    private int width;
    private int height;

    Rectangle() {
        this(1, 1); // Calls the second constructor
    }

    Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }
}
```

Rules:

- `this()` must be the **first** line in the constructor.

Summary

In this lecture you learned:

- Constructors initialize new objects.
- Java creates a default constructor only if you do not define any.
- Overloading allows multiple constructors or methods with the same name.
- `this()` is used to call one constructor from another.

7 Exercises

1. Create a class `Book` with fields: `title`, `author`, `year`. Provide at least **three constructors**.
2. Create a class `Circle` with constructors:
   - without parameters (radius = 1),
   - with radius parameter,
   - with diameter parameter.
3. Overload a method `multiply` so it works with:
   - two integers,
   - two doubles,
   - three integers.
4. Create a class `Player` with fields `name` and `score`. Add constructors that allow setting:
   - only the name (score = 0),
   - name and score,
   - no data (name = "Unknown", score = 0).
5. Challenge: Create a class `Vector2D` that supports:
   - constructor from two coordinates,
   - constructor copying another vector,
   - constructor creating a unit vector along X or Y (argument: ‘’x‘‘ or ‘’y‘‘).