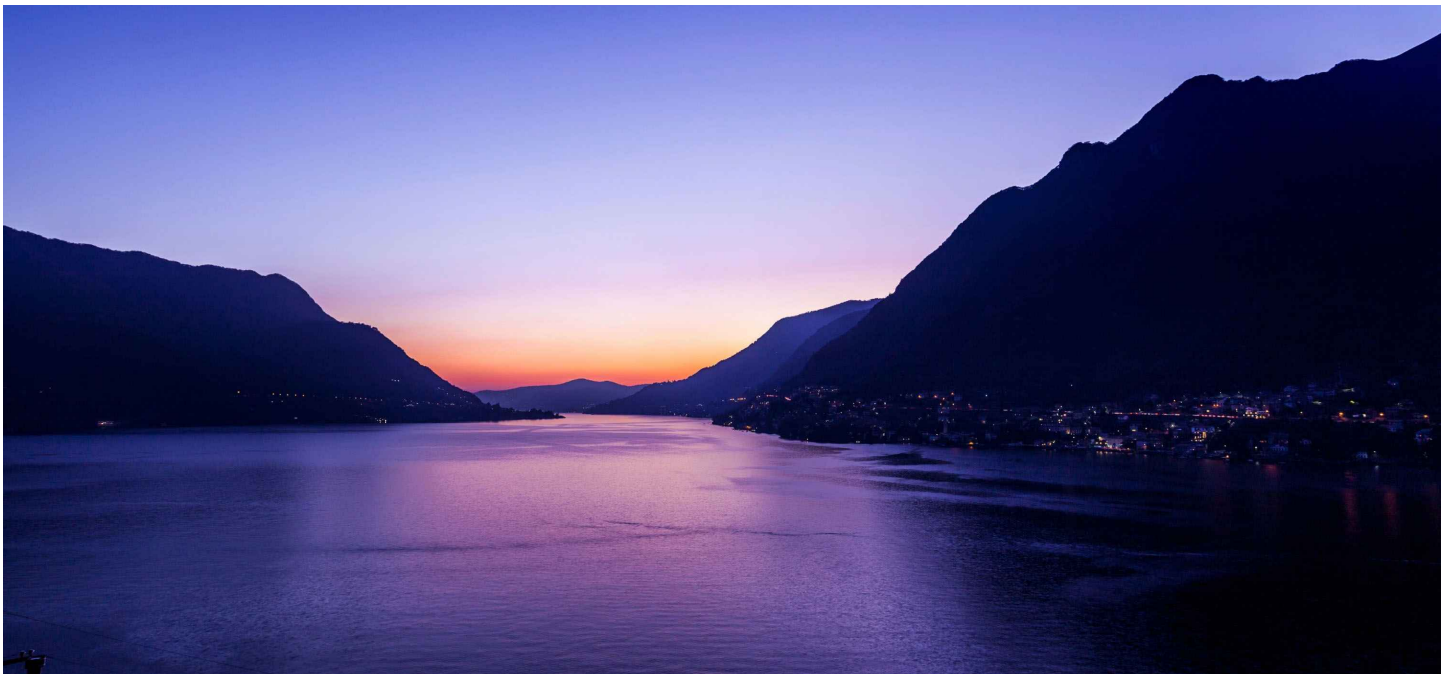


8-Static Methods

Author: Vincent Lau

Learning Objectives

> Understand the structure of declaring a static method
>
> Understand how to invoke a static method
>
> Understand why we need methods

Why do we need method

```
1 int radius = 5;
2 double pi = 3.14159;
3 int area = radius * radius * pi; // sounds alright?
```

How about we have 4 different radius, and we have to calculate all areas for each radius?

```
double pi = 3.14159;
int radius2 = 6;
int area2 = radius2 * radius2 * pi;

int radius3 = 7;
int area3 = radius3 * radius3 * pi;

int radius4 = 8;
int area4 = radius4 * radius4 * pi;
```

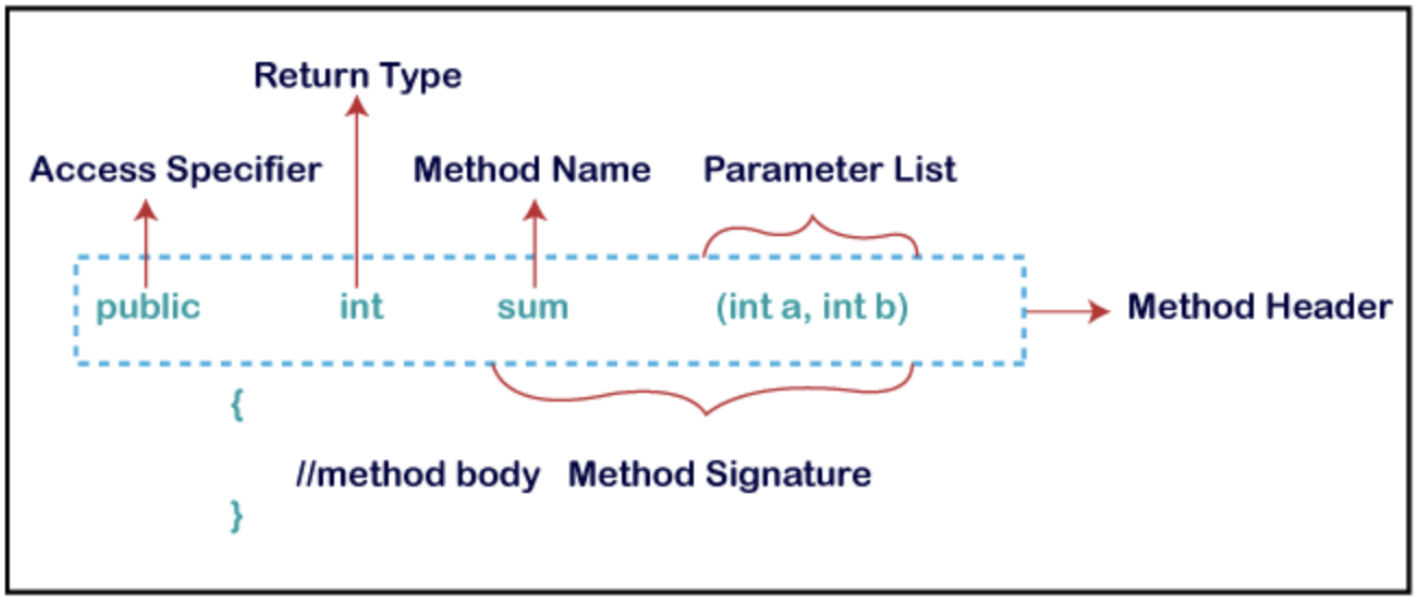
What are the problems here?

- Low readability
- Low reusability
- Repeated code, hard to review each formula to ensure no careless mistake
- If formula changes, it will be expensive to update all codes. And, the testing effort will be higher.

Solution by Methods

```
public class Methods {
    public static double calculateCircleArea(int radius) {
        return radius * radius * 3.14159;
    }
    public static void main(String[] args) {
        double area1 = calculateCircleArea(6);
        double area2 = calculateCircleArea(7);
        double area3 = calculateCircleArea(8);
    }
}
```

Introduction of Methods



In this chapter, we learn some basic rules of methods. In Java, methods are blocks of code that perform specific tasks and can be called to execute that code whenever needed. They encapsulate functionality and promote code reusability. Here's an introduction to methods in Java:

1. Method Name:
    - It is an identifier that represents the name of the method. **It can be duplicated in the class, referring to method signature.**
2. Method Parameter(s):
    - Parameters in methods are variables that are defined within the method signature and **used to pass values into the method**. They allow you to provide inputs to a method so that it can perform specific tasks or calculations based on those inputs.
    - The parameter is optional and specifies the input values required by the method. Multiple parameters can be separated by commas.
3. Method Signature:
    - The **method signature** consists of the **method name** and **its parameter types**.
    - **It helps distinguish methods based on their names** and the types of parameters they accept.
    - **Overloaded methods have the same name** but different parameter lists.
4. Method Body:
    - The method body contains the statements that define the functionality of the method.
    - It is enclosed within curly braces `{}` and executed when the method is called.
    - Statements can include variable declarations, conditional statements, loops, and other Java code.

5. Method Invocation:
    - To execute a method, you need to invoke or call it.
    - Method invocation follows the syntax `methodName(arguments)`.
    - Arguments correspond to the values passed to the method parameters, if any.
6. Return Statement:
    - The `return` statement terminates the method's execution and returns the specified value.
    - The `returnType` specifies the type of value the method will return. Use `void` if the method does not return any value.
7. Access Specifier
    - Will explain in later chapter.

Example 1

```
1  public class Greeting {
2
3      public static void sayHello(String name) {
4          System.out.println("Hello, " + name + "!");
5      }
6
7      public static void main(String[] args) {
8          String person = "John"; // start from here
9          sayHello(person);
10     }
11
12 }
```

In the above code, we have a class called `Greeting` with a static method named `sayHello()`. This method takes a single parameter of type `String` named `name`. Within the method, we concatenate the `name` parameter with a greeting message and print it to the console.

In the `main()` method, we declare a `String` variable called `person` and assign it the value "John". Then, we invoke the `sayHello()` method, passing the `person` variable as an argument. This value is then assigned to the `name` parameter in the `sayHello()` method, and the method is executed, printing "Hello, John!" to the console.

Example 2

```java
public class MathOperation {
    public static int add(int a, int b) {
        return a + b;
    }

    public static int substract(int a, int b) {
        return a - b;
    }

    public static int multiply(int a, int b) {
        return a * b;
    }

    public static int divide(int a, int b) {
        return a / b;
    }

    public static void main(String[] args) {
        int result = add(6, 3); // 9
        int result2 = substract(6, 3); // 3
        int result3 = multiply(6, 3); // 18
        int result4 = divide(6, 3); // 2
        System.out.println("Result: " + result);
        System.out.println("Result2: " + result2);
        System.out.println("Result3: " + result3);
        System.out.println("Result4: " + result4);
    }
}
```

In the code above, the `MathOperation` class contains a static method called `add()`, which takes two integers as parameters and returns their sum. The `add()` method can be called directly by `add(5, 3)`.

In the `main()` method, we invoke the `add()` method and store the result in the `result` variable. Finally, we print the result to the console.