Java Programming 1 - Week 5 Notes

Hia Al Saleh

October 3rd, 2024

Contents

1 Introduction to Functions

A function is a block of code used to complete a specific task. Functions help to:

- Simplify tasks and code.
- Remove duplicate code.
- Minimize the chance for errors.

In Java, a function is composed of the following elements:

- Access Modifier (e.g., `public`, `private`).
- Non-Access Modifier (optional, e.g., `static`, `final`).
- Return Type (e.g., `int`, `boolean`, `void`).
- Function Name.
- Formal Parameters.

2 Access Modifiers

Access modifiers determine the visibility of functions. The common modifiers include:

- `public`: The function can be accessed from anywhere.
- `private`: The function can only be accessed within the class.
- `protected`: The function can be accessed within the package and by subclasses.

Example:

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

3 Non-Access Modifiers

Non-access modifiers add special features to functions:

- `static`: Allows the function to be called without creating an object.
- `final`: Prevents overriding of the function.
- `abstract`: Creates a template for future functions (used in object-oriented programming).
- `synchronized`: Ensures only one thread uses the function at a time.

4 Return Types

The return type specifies the data type of the value returned by the function. Examples of return types include:

- `int`, `double`, `boolean`, `String`, etc.
- `void`: Indicates that no value is returned.

Example:

```
public double findSquareRoot(double number) {
    return Math.sqrt(number);
}
```

5 Naming Functions

When naming functions:

- Use camelCase for naming (e.g., `calculateTotal`).
- Choose meaningful names for clarity.
- Avoid starting names with _ or $.

6 Formal Parameters

Formal parameters allow you to pass data to a function when it is called. For example:

```
public int addNumbers(int num1, int num2) {
    return num1 + num2;
}
```

Here, `num1` and `num2` are formal parameters that the function uses.

7 Code Audit

We can improve code by modularizing repeated tasks into functions. For example, a `min()` function to return the smallest of two numbers:

```
public static double min(double num1, double num2) {
    return (num1 < num2) ? num1 : num2;
}
```

Similarly, we can create a `max()` function to find the larger number, and an `equals()` function to check equality:

```java
public static double max(double num1, double num2) {
    return (num1 > num2) ? num1 : num2;
}

public static boolean equals(double num1, double num2) {
    return num1 == num2;
}
```

8 Building a Program (Example: HighLow)

In this exercise, we will build a program that compares numbers entered by the user and keeps track of the highest and lowest numbers.

Steps:

1. Create a new project in IntelliJ named `HighLow`.
2. Create a new Java class file named `HighLow`.
3. Write the `main()` method, which takes user input and stores it in a `Scanner` object.
4. Use the `min()` and `max()` functions to update the highest and lowest numbers.

Example:

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    double num, highest, lowest;

    System.out.print("Enter a number: ");
    num = input.nextDouble();
    highest = lowest = num;

    // Continue accepting numbers until the user enters -1
    while (num != -1) {
        highest = max(highest, num);
        lowest = min(lowest, num);
        System.out.print("Enter a number: ");
        num = input.nextDouble();
    }

    System.out.println("Highest number: " + highest);
    System.out.println("Lowest number: " + lowest);
}
```

9 Homework

- Read Chapter 6 of the textbook.
- Review the `min()`, `max()`, and `equals()` functions and ensure you understand their structure.