Lesson 2: Conditional Statements and Loops

Marcin Kurzyna

Lecture Goals

This lecture introduces the control structures used to make decisions and repeat actions in Java: **conditional statements** and **loops**. By the end of this lecture, students should be able to:

- Use `if` and `switch` statements for decision making.
- Use `for` and `while` loops to repeat operations.
- Understand basic program flow control.

1 Introduction to Control Flow

So far, programs have executed statements sequentially — one after another. However, real-world problems often require:

- Making **decisions** based on conditions (e.g., "if the number is positive, print it").
- **Repeating** certain actions (e.g., "print numbers from 1 to 10").

Java provides special statements to control the flow of execution.

2 Conditional Statements

Conditional statements let a program choose between different actions depending on whether a condition is true or false.

2.1 The `if` Statement

The simplest form:

```
if (condition) {
    // statements to execute if condition is true
}
```

Example:

```
int number = 10;
if (number > 0) {
    System.out.println("The number is positive.");
}
```

2.2 `if-else` and `else if`

We can add alternatives:

```
if (number > 0) {
    System.out.println("Positive");
} else if (number < 0) {
    System.out.println("Negative");
} else {
    System.out.println("Zero");
}
```

Only one branch is executed depending on which condition is true.

2.3 Nested `if` Statements

You can place one `if` inside another:

```
int age = 20;
if (age >= 18) {
    if (age >= 65) {
        System.out.println("Senior citizen");
    } else {
        System.out.println("Adult");
    }
}
```

2.4 Comparison and Logical Operators

Conditions are expressions that evaluate to a boolean value (`true` or `false`). Common operators:

Operator	Meaning
`==`	equal to
`!=`	not equal to
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`&&`	logical AND
`||`	logical OR
`!`	logical NOT

Example:

```
if (x > 0 && x < 10) {
    System.out.println("x is between 0 and 10");
}
```

3 The `switch` Statement

The `switch` statement is used to select one of many possible actions based on a value.

```
int day = 3;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Invalid day");
}
```

Key points:

- Each `case` compares the value of `day` to a constant.
- `break` prevents execution from "falling through" to the next case.
- `default` runs if no case matches.

4 Loops

Loops allow you to execute a block of code multiple times.

4.1 The `while` Loop

Executes as long as a condition remains true:

```
int count = 1;
while (count <= 5) {
    System.out.println("Count: " + count);
    count++;
}
```

Note: Make sure the loop condition eventually becomes false; otherwise, the loop will run forever.

4.2 The `do-while` Loop

This version checks the condition **after** executing the body at least once.

```
int number = 1;
do {
    System.out.println("Number: " + number);
    number++;
} while (number <= 3);
```

4.3 The `for` Loop

The `for` loop is often used when you know in advance how many times to iterate.

```
for (int i = 1; i <= 5; i++) {
    System.out.println("i = " + i);
}
```

Parts of a `for` loop:

- **Initialization:** executed once before the loop starts.
- **Condition:** checked before each iteration.
- **Update:** executed after each iteration.

4.4 Nested Loops

Loops can be placed inside other loops.

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
        System.out.println("i=" + i + ", j=" + j);
    }
}
```

5 Example: Summing Numbers

This program computes the sum of numbers from 1 to 10.

```
class SumNumbers {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 10; i++) {
            sum = sum + i;
        }
        System.out.println("The sum is " + sum);
    }
}
```

Summary

This lecture introduced:

- Conditional statements: `if`, `if-else`, and `switch`.
- Loops: `while`, `do-while`, and `for`.
- Basic control flow concepts.

6 Exercises

1. Write a program that checks whether a given number is positive, negative, or zero.
2. Modify the above program to also check whether the number is even or odd.
3. Use a `switch` statement to print the name of a month given its number (1–12).
4. Write a program that prints all numbers from 1 to 10 using a `while` loop.
5. Write a program that prints even numbers between 1 and 20 using a `for` loop.
6. Create a program that calculates the factorial of a number (e.g., 5! = 120) using a `for` loop.
7. Modify the factorial program to use a `while` loop instead.
8. Write a program that asks for a password and repeats until the user enters the correct one.
9. Use nested loops to print a simple rectangle of asterisks, for example:

   ```
   ****
   ****
   ****
   ```

10. Challenge: Write a program that prints the multiplication table (1–10) using nested `for` loops.