# OODP workshop 4

1. What are different types of loops in java? Explain each of them with proper syntax.

Ans:

Types of Loops in Java

Java offers three primary loop structures, each with specific use cases and syntax patterns that make them suitable for different programming scenarios2.

**While Loop**

The while loop is a pretest loop that evaluates a condition before entering the loop body. If the condition is true, the statements inside the loop execute; otherwise, the loop is skipped entirely.

**while** (testCondition) {

// statements to execute

// increment/decrement statement to eventually make condition false

}

The while loop is particularly useful when you don't know in advance how many iterations will be needed. It's important to include an increment or decrement statement inside the loop to ensure the condition eventually becomes false; otherwise, you'll create an infinite loop.

**When to use:** When the number of repetitions cannot be determined before execution and zero or more repetitions are possible.

1. Write a program to ask the user to enter a number and print count down from that number to 0.

Ans:

**Do-While Loop**

The do-while loop is a post-test loop that guarantees at least one execution of the loop body before checking the condition. After executing the loop body once, it evaluates the condition and continues if the condition is true.

**do** {

// statements to execute

// increment/decrement statement

} **while** (testCondition);

Do-while loops are useful when you need to ensure the loop body executes at least once regardless of the initial condition state.

**When to use:** When the number of repetitions cannot be determined before execution AND at least one repetition is required.

1. Write a program to ask user to enter the temperature of seven days and then find out the average temperature and print the results.

Ans:

**For Loop**

The for loop is a compact, counter-controlled loop structure that combines initialization, condition checking, and increment/decrement operations in a single statement.

**for** (initialization; testCondition; increment/decrement) {

// statements to execute

}

The execution process follows these steps:

1. Initialization happens once at the beginning
2. Condition is evaluated before each iteration
3. If condition is true, loop body executes
4. After each iteration, the increment/decrement statement executes
5. When condition becomes false, the loop terminates

**When to use:** When the number of repetitions can be determined prior to executing the loop (counter-controlled loops).

1. Write a program to ask user a value and make sure that value is between 3 and 6 inclusive. Keep asking the user until user enters a valid value and display a success message and print error when user enter invalid value.

Ans:

A program that asks the user for a value, ensures it is between 3 and 6 inclusive, and keeps asking until a valid value is entered. It also displays a success message for valid input and an error message for invalid input.

import java.util.Scanner;

public class InputValidationProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int userValue;

// Loop until the user enters a valid value

do {

System.out.print("Enter a value between 3 and 6 (inclusive): ");

userValue = scanner.nextInt();

// Check if the value is valid

if (userValue >= 3 && userValue <= 6) {

System.out.println("Success! " + userValue + " is a valid value.");

} else {

System.out.println("Error: " + userValue + " is not between 3 and 6. Please try again.");

}

} while (userValue < 3 || userValue > 6); // Repeat if the value is invalid

scanner.close();

}

}

1. **Scanner Class**: Used to take input from the user.
2. **do-while Loop**: Ensures the loop body executes at least once, asking for input repeatedly until it meets the condition.
3. **Condition Checking**:
   * If userValue is between 3 and 6 (inclusive), it prints a success message.
   * Otherwise, it prints an error message and prompts the user again.
4. **Loop Termination**: The loop continues until userValue is within the valid range (3 to 6 inclusive).