Java Programming Tutorial for Linux Terminal

Prerequisites

Before starting, ensure you have Java installed on your Linux system:

```bash
# Check if Java is installed
java -version
javac -version

# If not installed, install OpenJDK (Ubuntu/Debian):
sudo apt update
sudo apt install default-jdk

# For Fedora/RHEL:
sudo dnf install java-latest-openjdk-devel
```

Course Outline

Module 1: Getting Started (Week 1)
1. Understanding Java basics
2. Your first Java program
3. Compilation and execution workflow
4. Basic syntax and structure
5. Comments and code organization

Module 2: Data Types and Variables (Week 2)
1. Primitive data types (int, double, boolean, char, etc.)
2. Variables and constants
3. Type casting and conversion
4. Operators (arithmetic, relational, logical)
5. String basics

Module 3: Control Flow (Week 3)
1. If-else statements
2. Switch statements
3. While and do-while loops
4. For loops
5. Break and continue

Module 4: Methods (Week 4)
1. Method declaration and calling
2. Parameters and return types
3. Method overloading
4. Scope and lifetime of variables
5. Recursion basics

Module 5: Arrays (Week 5)

1. Single-dimensional arrays
2. Multi-dimensional arrays
3. Array manipulation
4. Enhanced for loop
5. Common array operations

Module 6: Object-Oriented Programming Basics (Week 6-7)
1. Classes and objects
2. Constructors
3. Instance variables and methods
4. Encapsulation and access modifiers
5. The `this` keyword

Module 7: OOP Advanced Concepts (Week 8-9)
1. Inheritance
2. Polymorphism
3. Abstract classes
4. Interfaces
5. The `super` keyword

Module 8: Exception Handling (Week 10)
1. Try-catch blocks
2. Multiple catch blocks
3. Finally block
4. Throwing exceptions
5. Custom exceptions

Module 9: Working with Collections (Week 11)
1. ArrayList
2. LinkedList
3. HashMap
4. HashSet
5. Iterators

Module 10: File I/O (Week 12)
1. Reading from files
2. Writing to files
3. BufferedReader and BufferedWriter
4. File class operations
5. Exception handling with files

Lesson 1: Your First Java Program

The Classic "Hello, World!"

Create a new file called `HelloWorld.java`:

```java
public class HelloWorld {
    public static void main(String[] args) {
```

```
        System.out.println("Hello, World!");
    }
}
```

Understanding the Code

- `public class HelloWorld`: Declares a public class named HelloWorld
  - The filename MUST match the class name exactly (case-sensitive)
- `public static void main(String[] args)`: The entry point of your program
  - `public`: Accessible from anywhere
  - `static`: Can be called without creating an object
  - `void`: Doesn't return a value
  - `String[] args`: Command-line arguments
- `System.out.println()`: Prints text to the console with a new line

Compiling and Running in Terminal

```bash
# 1. Compile the program
javac HelloWorld.java

# This creates HelloWorld.class (bytecode)

# 2. Run the program
java HelloWorld

# Output: Hello, World!
```

Important Notes:
- Use `javac` to compile (creates .class files)
- Use `java` to run (don't include .class extension)
- The class name in the java command must match the class name in your code

---

Lesson 2: Variables and Data Types

Primitive Data Types

Create `DataTypes.java`:

```java
public class DataTypes {
    public static void main(String[] args) {
        // Integer types
        byte smallNumber = 127;           // -128 to 127
        short mediumNumber = 32000;       // -32,768 to 32,767
        int number = 2147483647;          // -2^31 to 2^31-1
        long bigNumber = 9223372036854775807L; // -2^63 to 2^63-1
```

```
        // Floating-point types
        float decimal = 3.14f;           // 32-bit floating point
        double preciseDecimal = 3.14159265359; // 64-bit floating point

        // Character and boolean
        char letter = 'A';               // Single character
        boolean isTrue = true;           // true or false

        // Printing variables
        System.out.println("Integer: " + number);
        System.out.println("Double: " + preciseDecimal);
        System.out.println("Character: " + letter);
        System.out.println("Boolean: " + isTrue);
    }
}
```

Compile and run:
```bash
javac DataTypes.java
java DataTypes
```

Working with Strings

Create `StringExample.java`:

```java
public class StringExample {
    public static void main(String[] args) {
        String greeting = "Hello";
        String name = "Java Programmer";

        // String concatenation
        String message = greeting + ", " + name + "!";
        System.out.println(message);

        // String methods
        System.out.println("Length: " + message.length());
        System.out.println("Uppercase: " + message.toUpperCase());
        System.out.println("Lowercase: " + message.toLowerCase());
        System.out.println("Contains 'Java': " + message.contains("Java"));
    }
}
```

Lesson 3: User Input

Reading from the Console

Create `UserInput.java`:

```java
import java.util.Scanner;

public class UserInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        System.out.println("Hello, " + name + "!");
        System.out.println("You are " + age + " years old.");

        scanner.close();
    }
}
```

Scanner Methods:
- `nextLine()` - reads a line of text
- `nextInt()` - reads an integer
- `nextDouble()` - reads a double
- `nextBoolean()` - reads a boolean

Lesson 4: Control Flow - If Statements

Create `IfExample.java`:

```java
import java.util.Scanner;

public class IfExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        if (age >= 18) {
            System.out.println("You are an adult.");
        } else if (age >= 13) {
            System.out.println("You are a teenager.");
        } else {
            System.out.println("You are a child.");
```

```
        }

        scanner.close();
    }
}
```

Operators

Comparison Operators:
- `==` equal to
- `!=` not equal to
- `>` greater than
- `<` less than
- `>=` greater than or equal to
- `<=` less than or equal to

Logical Operators:
- `&&` AND
- `||` OR
- `!` NOT

Lesson 5: Loops

For Loop

Create `ForLoop.java`:

```java
public class ForLoop {
    public static void main(String[] args) {
        // Print numbers 1 to 10
        for (int i = 1; i <= 10; i++) {
            System.out.println("Number: " + i);
        }

        // Print even numbers from 0 to 20
        for (int i = 0; i <= 20; i += 2) {
            System.out.println("Even number: " + i);
        }
    }
}
```

While Loop

Create `WhileLoop.java`:

```java
import java.util.Scanner;
```

```
public class WhileLoop {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = "";

        while (!input.equals("quit")) {
            System.out.print("Enter a command (type 'quit' to exit): ");
            input = scanner.nextLine();
            System.out.println("You entered: " + input);
        }

        System.out.println("Goodbye!");
        scanner.close();
    }
}
```

Lesson 6: Methods

Create `MethodsExample.java`:

```java
public class MethodsExample {

    // Method that returns a value
    public static int add(int a, int b) {
        return a + b;
    }

    // Method that doesn't return a value
    public static void greet(String name) {
        System.out.println("Hello, " + name + "!");
    }

    // Method with multiple parameters
    public static double calculateArea(double length, double width) {
        return length * width;
    }

    public static void main(String[] args) {
        int sum = add(5, 3);
        System.out.println("Sum: " + sum);

        greet("Alice");

        double area = calculateArea(5.0, 3.0);
        System.out.println("Area: " + area);
    }
}
```

```

---

## Lesson 7: Arrays

Create `ArrayExample.java`:

```java
public class ArrayExample {
    public static void main(String[] args) {
        // Declaring and initializing an array
        int[] numbers = {10, 20, 30, 40, 50};

        // Accessing elements
        System.out.println("First element: " + numbers[0]);
        System.out.println("Last element: " + numbers[4]);

        // Array length
        System.out.println("Array length: " + numbers.length);

        // Iterating through an array
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("Element at index " + i + ": " + numbers[i]);
        }

        // Enhanced for loop
        System.out.println("\nUsing enhanced for loop:");
        for (int num : numbers) {
            System.out.println(num);
        }

        // Creating an empty array
        String[] names = new String[3];
        names[0] = "Alice";
        names[1] = "Bob";
        names[2] = "Charlie";

        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Lesson 8: Classes and Objects

Create `Person.java`:

```java
```
```

```
public class Person {
    // Instance variables (attributes)
    private String name;
    private int age;

    // Constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Methods (behaviors)
    public void introduce() {
        System.out.println("Hi, I'm " + name + " and I'm " + age + " years old.");
    }

    // Getter methods
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    // Setter methods
    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        }
    }
}
```

Create `PersonTest.java`:

```java
public class PersonTest {
    public static void main(String[] args) {
        // Creating objects
        Person person1 = new Person("Alice", 25);
        Person person2 = new Person("Bob", 30);

        // Calling methods
        person1.introduce();
        person2.introduce();

        // Using getters and setters
```

```
        System.out.println(person1.getName() + " is " + person1.getAge() + "
years old.");

        person1.setAge(26);
        person1.introduce();
    }
}
```

Compiling multiple files:
```bash
# Compile all .java files in the directory
javac *.java

# Or compile specific files
javac Person.java PersonTest.java

# Run the main class
java PersonTest
```

Terminal Workflow Tips

Organizing Your Code

```bash
# Create a directory structure
mkdir -p java-learning/lesson1
cd java-learning/lesson1

# Create your Java file
nano HelloWorld.java  # or use vim, gedit, etc.

# Compile
javac HelloWorld.java

# Run
java HelloWorld

# Clean up .class files
rm *.class
```

Common Compilation Errors

1. **File name doesn't match class name**
   - Error: `class HelloWorld is public, should be declared in a file named
HelloWorld.java`
   - Solution: Rename file to match class name exactly

2. **javac: command not found**
   - Solution: Install JDK or add Java to PATH

3. **Error: Could not find or load main class**
   - Solution: Make sure you're using `java ClassName` (not `java ClassName.class`)

Practice Exercises

Exercise 1: Calculator
Create a simple calculator that takes two numbers and an operator from the user and performs the calculation.

Exercise 2: Grade Calculator
Write a program that takes a numerical grade (0-100) and outputs the letter grade (A, B, C, D, F).

Exercise 3: Number Guessing Game
Create a game where the computer picks a random number and the user tries to guess it.

Exercise 4: Array Statistics
Write a program that calculates the average, maximum, and minimum of an array of numbers.

Exercise 5: Bank Account Class
Create a `BankAccount` class with deposit, withdraw, and check balance methods.

Next Steps

After completing this tutorial:

1. Learn about inheritance and polymorphism
2. Study exception handling in depth
3. Explore Java Collections Framework
4. Learn about file I/O operations
5. Understand multithreading basics
6. Study design patterns

Additional Resources

- Oracle Java Documentation: https://docs.oracle.com/en/java/
- Java Tutorials: https://docs.oracle.com/javase/tutorial/
- Practice coding: LeetCode, HackerRank, Codewars

Good luck with your Java learning journey!