Module 1: Introduction to Java and Software Engineering

Ngane Emmanuel

2024-11-06

Module 1: Introduction to Java and Software Engineering

Duration: 2 Hours

Learning Objectives:

- Understand what Java is and how it fits into the software development process.
- Get familiar with the Java development environment (JVM, JDK, JRE).
- Learn the basics of Software Engineering and how Java programming plays a role in it.
- Understand the Software Development Life Cycle (SDLC) and its stages.
- Set up a "Hello World" Java project, compile, and run a Java program.

Lesson Outline:

1. Introduction to Java (30 minutes) What is Java?

- Java is an object-oriented programming language that is widely used for building desktop applications, web applications, and mobile applications.
- It was developed by Sun Microsystems in the mid-1990s and later acquired by Oracle Corporation.
- Java is known for its portability, security features, and robust performance. It follows the "Write Once, Run Anywhere" (WORA) principle, meaning that compiled Java code can run on any machine that has the Java Virtual Machine (JVM) installed, irrespective of the underlying hardware and operating system.

Key Components of Java:

1. **JVM (Java Virtual Machine):**
   - The JVM is the engine that provides runtime environment for Java bytecode. It runs Java programs by interpreting the bytecode.
   - It abstracts the underlying operating system and hardware, enabling Java programs to run on any platform that supports JVM.
2. **JRE (Java Runtime Environment):**
   - The JRE includes the JVM and the standard library required to run Java applications. It provides the environment necessary for executing Java bytecode but does not include development tools like compilers.
3. **JDK (Java Development Kit):**
   - The JDK includes everything in the JRE along with development tools such as the Java compiler (`javac`), debuggers, and other utilities. The JDK is essential for writing and compiling Java programs.

Why Java?

- **Portability:** Java is platform-independent because of the JVM.
- **Robustness:** Java provides strong memory management (automatic garbage collection) and exception handling.
- **Security:** Java has a built-in security manager that restricts the capabilities of running programs.
- **Multithreading Support:** Java supports multithreading, enabling concurrent execution of programs.

Java Ecosystem:

- **Standard Libraries (API):** Java has a rich set of built-in libraries (e.g., Collections API, networking API, I/O classes) that make it easier to develop robust applications.
- **IDE Support:** Integrated Development Environments (IDEs) such as Eclipse, IntelliJ IDEA, or VS Code simplify Java development by providing features like code completion, debugging, and version control integration.

Quick Demo:

- Introduce a simple Java program to demonstrate the structure of a Java class.
- **Code Example:**

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

  - **Explanation:**
    - `public class HelloWorld`: Defines a class named `HelloWorld`.
    - `public static void main(String[] args)`: The entry point of the program, where execution starts.
    - `System.out.println("Hello, World!")`: Prints the message to the console.

Setting Up Java Development Environment: - Walk students through installing Java JDK and configuring the environment variable. - Demonstrate the installation of an IDE (e.g., IntelliJ or Eclipse). - Show how to create and run a simple Java program using the IDE and from the command line.

2. Introduction to Software Engineering (30 minutes) What is Software Engineering?

- Software engineering is the application of engineering principles to the design, development, maintenance, testing, and evaluation of software and systems.
- It is a disciplined, structured approach to software development to ensure high-quality software solutions.

Key Software Engineering Concepts:

- **Software Development Life Cycle (SDLC):** A structured approach to software development involving stages such as planning, design, coding, testing, and deployment.
- **System Design:** The process of defining the architecture, components, interfaces, and data for a system.

- **Version Control:** Managing changes to software code using tools like Git, ensuring collaboration, and tracking changes.

Why is Software Engineering Important?

- It improves the quality and maintainability of software by following best practices.
- It enables teams to manage complex projects and deadlines.
- It ensures the software meets user requirements and is scalable.

Software Engineering Methodologies:

- **Waterfall Model:** A linear and sequential approach where each phase must be completed before moving to the next.
- **Agile Model:** An iterative approach that emphasizes flexibility, collaboration, and customer feedback. Projects are divided into smaller parts called sprints.
- **DevOps:** A collaborative methodology that integrates development and operations, emphasizing automation and continuous delivery.

The Role of Java in Software Engineering:

- Java is one of the most widely used languages in software engineering for building robust, scalable, and maintainable applications.
- Java is used in various stages of SDLC, from initial design to deployment.

3. The Software Development Life Cycle (SDLC) Overview (30 minutes) What is SDLC?

- The Software Development Life Cycle (SDLC) is a systematic process used for planning, creating, testing, and deploying software systems.
- SDLC ensures that software is built on time, within budget, and with the required functionality.

Stages of SDLC:

1. **Requirement Gathering:**
   - Gathering functional and non-functional requirements from stakeholders.
   - Tools: Interviews, surveys, and documentation.
2. **System Design:**
   - Defining the software architecture, choosing technologies, and creating system diagrams (e.g., UML).
   - Java is used to implement this design in real-world applications.
3. **Implementation:**
   - Writing code based on the system design.
   - In Java, this includes creating classes, methods, and using libraries to build functionality.
4. **Testing:**
   - Testing the software to ensure it meets requirements and is free of bugs.
   - Java frameworks like JUnit are used for unit testing.
5. **Deployment:**
   - Deploying the software to the production environment.

6. **Maintenance:**
    - Ongoing updates and bug fixes to ensure the software continues to function correctly over time.

SDLC Models:

- **Waterfall Model:** A rigid approach where each phase is completed before the next starts. Suitable for smaller projects.
- **Agile Model:** A more flexible, iterative approach that allows for regular feedback and adjustments. Ideal for larger, more complex projects.

Java in the SDLC:

- Java is used extensively across all stages: from writing functional code to testing and deployment. It plays a crucial role in ensuring that the software is both robust and scalable.

4. Hands-On Activity: Setting Up and Running a Java Project (30 minutes) **Objective:** Set up a "Hello World" Java project and compile and run it.

1. **Step 1: Installing JDK:**
    - Walk through installing Java JDK from the official Oracle website.
    - Set up Java on the machine and ensure `javac` and `java` commands work from the terminal.
2. **Step 2: Setting Up IDE (Optional):**
    - Install IntelliJ IDEA or Eclipse IDE.
    - Show students how to create a new Java project.
3. **Step 3: Writing the First Program:**
    - Create a simple "Hello World" program using the IDE or a text editor.
    - Discuss basic syntax and structure of Java programs.
4. **Step 4: Compilation and Execution:**
    - Show students how to compile the Java program using the terminal (`javac HelloWorld.java`) and run it (`java HelloWorld`).
    - Demonstrate how this can be done inside the IDE as well.

Conclusion & Homework (10 minutes)

Wrap-Up:

- Review the key points: Java, JDK, JRE, JVM, and SDLC.
- Explain that Java is not just a programming language but a key enabler in many software engineering practices.
- Emphasize the importance of understanding the SDLC in real-world projects.

Homework Assignment:

- Install Java and an IDE
- Write a simple program to print the user's name using `System.out.println()`.
- Read up on SDLC models and prepare a brief summary of Agile and Waterfall.