Java Programming - I: Session 1 - Introduction to Java

Course Overview

Welcome to **Java Programming - I**! This course introduces you to the fundamentals of Java, a versatile and widely-used object-oriented programming language for building robust applications. By the end of this 36-hour course, delivered over 18 two-hour sessions, you will be able to:

- Develop and declare Java classes.
- Work with variables, data types, operators, and control structures.
- Use arrays, strings, packages, and access specifiers.
- Understand inheritance, polymorphism, and error handling.
- Explore modern Java features, including the Date and Time API, functional programming, and JDK 20 enhancements.

You'll engage with theoretical concepts, hands-on coding, and "Try It Yourself" exercises from *Java Programming - The Complete Guide for Beginners*. Resources like eBooks, practice tests, and lab assignments are available on OnlineVarsity.

Session 1 Objectives

In this session, we will:

- Understand structured and object-oriented programming paradigms.
- Explore Java's features as an object-oriented programming (OOP) language.
- Learn about the Java platform and its components.
- Identify different editions of Java.
- Trace the evolution of Java Standard Edition (Java SE).
- Set up the Java Development Kit (JDK) and Visual Studio Code (VS Code) for Java development.

1. Introduction to Programming Paradigms

Structured Programming

- **Definition**: A programming approach using a linear, step-by-step process with control structures like sequences, conditionals (if-then-else), and loops (while, for).
- **Characteristics**:
  - Code organized into procedures or functions.
  - Focus on breaking tasks into manageable subroutines.
  - Example: A program to sum numbers using a loop.
- **Limitations**: Can become complex for large systems, lacking modularity.

Object-Oriented Programming (OOP)

- **Definition**: A paradigm organizing code into objects, instances of classes that combine data (attributes) and behavior (methods).
- **Key Principles**:
  - **Encapsulation**: Bundling data and methods, restricting access for data integrity.
  - **Inheritance**: Allowing a class to inherit properties/methods from another.
  - **Polymorphism**: Treating objects as instances of their parent class with specific behavior.
  - **Abstraction**: Hiding complex details, exposing only necessary features.
- **Advantages**: Modular, reusable, and scalable code.
- **Example**: A "Car" class with attributes (color, speed) and methods (drive, stop).

2. Features of Java as an OOP Language

Java is a robust, platform-independent OOP language with:

- **Simple**: Familiar C/C++-like syntax, with automatic memory management (garbage collection).
- **Object-Oriented**: Supports encapsulation, inheritance, polymorphism, and abstraction.
- **Platform-Independent**: Compiles to bytecode, runnable on any device with a Java Virtual Machine (JVM).
- **Secure**: Sandboxing and bytecode verification ensure safe execution.
- **Multithreaded**: Supports concurrent task execution.
- **Robust**: Strong type-checking and exception handling.
- **Portable**: Write Once, Run Anywhere (WORA) via JVM.
- **High Performance**: Just-In-Time (JIT) compilation optimizes bytecode.

3. Java Platform and Its Components

The Java platform includes:

- **Java Virtual Machine (JVM)**: Executes bytecode, ensuring platform independence.
- **Java Runtime Environment (JRE)**: Includes JVM and libraries to run Java applications.
- **Java Development Kit (JDK)**: Includes JRE, compilers (javac), and tools (javadoc, jar) for development.
- **Java APIs**: Libraries for tasks like file I/O, networking, and GUI development.

4. Editions of Java

Java has multiple editions:

- **Java SE (Standard Edition)**: For desktop and standalone applications.
- **Java EE (Enterprise Edition)**: For large-scale, distributed enterprise applications.
- **Java ME (Micro Edition)**: For resource-constrained devices (e.g., mobile phones).
- **JavaFX**: For rich, cross-platform GUI applications.

5. Evolution of Java Standard Edition (Java SE)

Key milestones:

- **JDK 1.0 (1996)**: Basic features introduced.
- **JDK 1.2 (1998)**: Added Swing, Collections Framework.
- **Java SE 5 (2004)**: Generics, enums, annotations.
- **Java SE 8 (2014)**: Lambda expressions, Stream API.
- **Java SE 11 (2018)**: Long-term support (LTS), modular system (JPMS).
- **Java SE 17 (2021)**: LTS, sealed classes, pattern matching.
- **Java SE 20 (2023)**: Virtual threads, vector API (covered in Session 17).

6. Setting Up the Java Development Environment

To code in Java, you'll need the JDK and Visual Studio Code (VS Code) with Java extensions. Follow these steps:

Step 1: Download and Install JDK

1. **Visit the Oracle JDK Website**:
   - Go to https://www.oracle.com/java/technologies/javase-downloads.html.
   - Download JDK 20 (or the latest version recommended for the course).
2. **Choose the Installer**:
   - Select the installer for your OS (Windows, macOS, Linux).
3. **Install JDK**:
   - Run the installer and follow prompts.
   - Note the installation path (e.g., `C:\Program Files\Java\jdk-20` on Windows).
4. **Verify Installation**:
   - Open a terminal (Command Prompt on Windows, Terminal on macOS/Linux).
   - Run:

     ```
     java -version
     javac -version
     ```

   - Expected output:

     ```
     java version "20.0.1" 2023-04-18
     javac 20.0.1
     ```

Step 2: Set Up Environment Variables

Configure `JAVA_HOME` and `PATH` to run Java commands globally.

On Windows:

1. **Set JAVA_HOME**:
   - Right-click "This PC" > Properties > Advanced system settings > Environment Variables.
   - Under "System Variables," click "New":
     - Variable name: `JAVA_HOME`
     - Variable value: JDK path (e.g., `C:\Program Files\Java\jdk-20`).
2. **Update PATH**:
   - In "System Variables," edit `Path` .
   - Add: `%JAVA_HOME%\bin` .
3. **Verify**:
   - Open a new Command Prompt and run `java -version` and `javac -version` .

On macOS/Linux:

1. **Set JAVA_HOME**:
   - Edit the shell configuration file (e.g., `~/.bashrc`, `~/.zshrc`):

     ```
     export JAVA_HOME=$(/usr/libexec/java_home)
     ```

     Or specify manually, e.g.,

     `export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home`.
2. **Update PATH**:
   - Add:

     ```
     export PATH=$JAVA_HOME/bin:$PATH
     ```

3. **Apply Changes**:
   - Run `source ~/.bashrc` (or `~/.zshrc`).
4. **Verify**:
   - Run `java -version` and `javac -version`.

Step 3: Install and Configure VS Code

VS Code is a lightweight, customizable IDE perfect for Java development.

1. **Download VS Code**:
   - Visit https://code.visualstudio.com/ and download for your OS.
   - Install and launch VS Code.
2. **Install Java Extensions**:
   - Open the Extensions view (`Ctrl+Shift+X` or `Cmd+Shift+X` on macOS).
   - Install the **Java Extension Pack** by Microsoft, which includes:
     - Language Support for Java by Red Hat.
     - Debugger for Java.
     - Java Test Runner.
     - Maven for Java.
     - Project Manager for Java.
3. **Verify Java Support**:
   - VS Code should detect the JDK automatically. If prompted, select the JDK 20 path.
   - Create a new folder for your Java project and open it in VS Code (`File > Open Folder`).

Step 4: Write and Run Your First Java Program

1. **Create a Java File**:
   - In VS Code, create a new file named `HelloWorld.java` in your project folder.
   - Add:

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

2. **Compile and Run**:
   - In VS Code, right-click the file and select **Run Java** (Java Extension Pack enables this).
   - Alternatively, open the terminal in VS Code (`Ctrl+``) and run:

```
javac HelloWorld.java
java HelloWorld
```

   - Output: `Hello, World!`
3. **Debugging**:
   - Set a breakpoint by clicking to the left of a line number.
   - Press `F5` or select **Debug Java** to start debugging.

7. Getting Started with OnlineVarsity

- **Access**: Log in to OnlineVarsity for course resources.
- **Resources**:
  - **eBook**: Download *Java Programming - The Complete Guide for Beginners*.
  - **Practice 4 Me**: Test your knowledge with quizzes.
  - **Work Assignments**: Complete hands-on lab assignments.
  - **References**: Access additional reading materials.

8. Next Steps

- Review Session 1 in *Java Programming - The Complete Guide for Beginners*.
- Complete "Try It Yourself" questions for Session 1 (covered in Session 3).
- Prepare for Session 2 on variables, data types, and operators.

Additional Resources

- **Books**:
  - *Head First Java* by Kathy Sierra & Bert Bates.
  - *Java: A Beginner's Guide* by Herbert Schildt.
  - *Java for Dummies* by Barry A. Burd.
- **Online**: Oracle Java Tutorials (https://docs.oracle.com/javase/tutorial/).
- **VS Code Java Guide**: https://code.visualstudio.com/docs/languages/java.