Java Basics Course Study Materials

1. Introduction to Java

Overview:

Java is a versatile, high-level programming language designed for developing a wide range of applications. Known for its platform independence, Java allows code to run on any device equipped with a Java Virtual Machine (JVM). Java is used in various domains, including web development, mobile applications, enterprise software, and scientific computing.

History:

Java was developed by Sun Microsystems and was released in 1995. It was initially called Oak, named after an oak tree that stood outside the developer's office. Later, the name was changed to Java, inspired by Java coffee. Java's main goal was to create a language that could run on various devices and be platform-independent. In 2010, Oracle Corporation acquired Sun Microsystems and took over the development of Java.

Key Features:

Object-Oriented: Java is an object-oriented programming language that supports OOP concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction. This allows for modular, scalable, and maintainable code.

Platform-Independent: Java's motto, "Write Once, Run Anywhere," is achieved through the use of the Java Virtual Machine (JVM). Java programs are compiled

into bytecode, which can be executed on any device with a JVM, regardless of the underlying hardware or operating system.

Secure: Java includes a security manager that defines the access rules for Java applications. It provides a secure environment for running code, especially important for web applications and applets.

Robust: Java has strong memory management, automatic garbage collection, and exception handling capabilities, which contribute to the reliability and robustness of applications.

Multithreaded: Java supports multithreading, allowing multiple threads to run concurrently within a single program. This is beneficial for applications that require concurrent processing, such as games and real-time simulations.

Portable: Java programs are platform-independent at both the source and binary levels. The Java compiler converts the source code into bytecode, which can be executed on any platform with a JVM.

Setting Up Java Development Kit (JDK):

To develop Java applications, you need to install the Java Development Kit (JDK). The JDK includes the Java Runtime Environment (JRE), the Java compiler, and other development tools.

1.1 Download JDK:

- Visit Oracle's official website or OpenJDK website.
- Download the appropriate version of the JDK for your operating system.
- Install JDK:
- Run the installer and follow the on-screen instructions.
- Set up environment variables:
- PATH: Add the JDK's bin directory to the PATH environment variable to enable running Java commands from the command line.

- JAVA_HOME: Set the JAVA_HOME environment variable to point to the JDK installation directory.
- Setting Up Integrated Development Environment (IDE):
- An IDE simplifies the development process by providing tools for writing, compiling, and debugging Java code. Popular IDEs for Java include Eclipse, IntelliJ IDEA, and NetBeans.

1.2 Download and Install IDE:

- Download the IDE installer from the official website.
- Install the IDE by following the on-screen instructions.
- Create a New Project:
- Open the IDE and create a new Java project.
- Set up the project structure, including source folders and libraries.
- Write Java Code:
- Use the IDE's code editor to write Java programs.
- Take advantage of features like syntax highlighting, code completion, and error checking.
- Compile and Run:
- Use the IDE's built-in tools to compile and run your Java programs.
- Debug code using the IDE's debugging tools, such as breakpoints and variable inspection.
- Example: Hello World Program

Example:

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

Results:

- Class Declaration: public class Main
- Declares a public class named Main.
- Main Method: public static void main(String[] args)
- Defines the main method, which is the entry point of the program.
- Print Statement: System.out.println("Hello, World!");
- Prints "Hello, World!" to the console.