**Name:-Nilesh Fate\_JH**

**ASSIGNMENT\_01**

**1. Reading Assignment: A Short History of Java**

* **Task**: Read about the history and development of Java.

**History and Development of Java**

**1. Origins:**

* **Developer:** Java was developed by a team of engineers led by James Gosling at Sun Microsystems (which was later acquired by Oracle Corporation).
* **Project Name:** The project initially started in 1991 under the name "Green Project."
* **Original Purpose:** Java was originally intended for interactive television, but the team realized that it was too advanced for the digital cable television industry at the time.

**2. Creation of Java:**

* **Oak:** The original name of Java was "Oak," named after an oak tree that stood outside Gosling’s office. The name was later changed to "Java" due to trademark issues. The name "Java" was chosen after the team consumed a lot of coffee during development, leading them to select Java, inspired by the coffee from the Indonesian island of Java.
* **1995 Public Release:** Java was officially released to the public in 1995 as part of Sun Microsystems’ Java Platform. The motto "Write Once, Run Anywhere" (WORA) became a key selling point, emphasizing the language's platform independence.

**3. Key Versions and Milestones:**

* **JDK 1.0 (1996):** The first version of the Java Development Kit (JDK) was released. It included the basics of Java, such as the core language, libraries, and tools like the Java compiler and Java runtime environment.
* **Java 2 (1998):** This version introduced the Java 2 Platform, Standard Edition (J2SE), which included significant improvements such as Swing (for GUI development), Collections Framework, and Java Naming and Directory Interface (JNDI). Java was also split into three editions: J2SE (Standard Edition), J2EE (Enterprise Edition), and J2ME (Micro Edition).
* **Java 5 (2004):** Java 5 brought major updates like generics, metadata (annotations), enumerated types, and the enhanced for loop.
* **Java 6 (2006):** This version focused on performance improvements, new web services

**2. Reading Assignment: Java Language Features**

* **Task**: Learn about the main features of Java.

**1. Object-Oriented Programming (OOP)**

* **Objects and Classes**: Java is inherently object-oriented, meaning it is based on the concept of "objects" which are instances of "classes." This promotes modular and reusable code.
* **Inheritance**: Java supports inheritance, allowing one class to inherit the properties and behaviors of another.
* **Encapsulation**: Java allows encapsulation, where data and methods are bundled together, and access to them is controlled.
* **Polymorphism**: Java supports polymorphism, enabling objects to be treated as instances of their parent class.
* **Abstraction**: Java allows abstract classes and interfaces, which helps in defining the structure of a system without revealing the implementation details.

**2. Platform Independence**

* **Write Once, Run Anywhere (WORA)**: Java code is compiled into bytecode, which can be run on any device with a Java Virtual Machine (JVM), regardless of the underlying operating system. This makes Java highly portable.

**3. Robust and Secure**

* **Memory Management**: Java has an automatic garbage collection feature that manages memory allocation and deallocation, reducing the chances of memory leaks.
* **Exception Handling**: Java provides a robust exception handling mechanism to catch and manage runtime errors.
* **Security Features**: Java has a secure runtime environment, with features like bytecode verification, secure class loading, and a Security Manager to define access rules.

**4. Multi-threading**

* Java supports multi-threading, allowing multiple threads (smallest units of a process) to run concurrently. This enables the creation of highly responsive and efficient applications.

**5. Rich Standard Library**

* Java comes with a vast standard library (Java Standard Edition - Java SE) that provides pre-built classes and functions for tasks such as data structures, networking, input/output operations, and more. This speeds up the development process.

**6. High Performance**

* Although Java is interpreted, its Just-In-Time (JIT) compiler helps optimize performance by converting bytecode into native machine code at runtime, which makes execution faster.

**3. Reading Assignment: Which Version of JDK Should I Use?**

* **Task**: Find out which JDK version is right for you.

**1. Latest LTS Version: JDK 17**

* **Recommended For:** Most developers, especially those working on long-term projects or enterprise applications.
* **Support:** JDK 17 is the latest Long-Term Support (LTS) version, meaning it will receive updates and support for several years.
* **Stability:** As an LTS version, it is stable and well-tested, making it suitable for production environments.
* **Features:** It includes the latest stable features and improvements in the language, libraries, and JVM.

**2. Most Recent Version: JDK 21**

* **Recommended For:** Developers who want to experiment with the latest Java features or those working on cutting-edge projects.
* **Support:** Non-LTS versions like JDK 21 receive updates for a shorter period, typically six months.
* **Features:** It includes the most recent enhancements and language features but may not be as stable as LTS versions for production use.

**3. Older LTS Versions: JDK 11 and JDK 8**

* **JDK 11 (2018):**
  + **Recommended For:** Projects that started a few years ago and rely on Java 11 features. It's still widely used in many enterprise environments.
  + **Support:** As an LTS version, it continues to receive updates and support.
* **JDK 8 (2014):**
  + **Recommended For:** Legacy projects or systems that were built on Java 8. Many enterprises still use Java 8 due to its stability and long period of support.
  + **Support:** It has been one of the most popular versions and received extended support, but it's becoming outdated for new development.

**4. Considerations for Choosing a JDK Version:**

* **Project Requirements:** If your project or workplace mandates a specific JDK version, follow those guidelines.
* **Tool Compatibility:** Some development tools, libraries, or frameworks may require a specific JDK version.
* **Learning Purpose:** If you're learning Java, starting with the latest LTS version (JDK 17) is a good choice. It provides a balance of modern features and stability.
* **Community and Ecosystem:** The broader Java community and ecosystem typically adopt LTS versions, making it easier to find resources, tutorials, and community support.

**4. Reading Assignment: JDK Installation Directory Structure**

* **Task**: Understand the folder structure and files in the JDK installation.

**JDK Installation Directory Structure**

**1. bin Directory**

* **Purpose:** This directory contains executable files (binaries) for Java development tools.
* **Common Files:**
  + java: The Java runtime environment launcher that runs Java applications.
  + javac: The Java compiler, which compiles .java files into bytecode .class files.
  + javap: The class file disassembler, used to view the structure of compiled classes.
  + javadoc: The tool for generating HTML documentation from Java source files.
  + jar: The tool for creating and managing JAR (Java ARchive) files.
  + jdb: The Java debugger for finding and fixing bugs in Java applications.
* **Usage:** You can run these commands directly from the terminal or command prompt.

**2. lib Directory**

* **Purpose:** Contains essential libraries required by the JDK.
* **Common Files:**
  + tools.jar: Contains classes required for the development tools in the bin directory.
  + dt.jar: Contains the Swing classes for building graphical user interfaces (GUIs).
* **Usage:** These libraries are automatically used by the JDK tools and the JVM, so you typically don’t need to interact with them directly.

**3. jmods Directory (from JDK 9 onwards)**

* **Purpose:** Contains JMOD files, which are modules that package Java platform classes, interfaces, and other resources.
* **Common Files:**
  + Each .jmod file corresponds to a specific module (e.g., java.base.jmod for the base module).
* **Usage:** The modular structure allows for more efficient application packaging and distribution.

**4. include Directory**

* **Purpose:** Contains C header files for JNI (Java Native Interface) programming.
* **Common Files:**
  + jni.h: A key header file for writing native methods in C/C++ that can be called from Java code.
  + jvm.h: Provides access to the JVM functions.
* **Usage:** This directory is mainly used when writing Java applications that interface with native code.

**5. conf Directory (from JDK 9 onwards)**

* **Purpose:** Contains configuration files for the JDK and JVM.
* **Common Files:**
  + net.properties: Configurations related to networking.
  + security: A directory containing security policies and configuration files.
* **Usage:** Modify these files only if you need to change the default behavior of the JDK or JVM.

**6. jre Directory (up to JDK 8)**

* **Purpose:** Contains the Java Runtime Environment, which is used to run Java applications.
* **Common Files/Directories:**
  + bin: Contains binaries necessary for running Java applications (java, keytool, etc.).
  + lib: Contains the runtime libraries required by the JVM.
* **Note:** Starting with JDK 9, the jre directory was removed and merged with the main JDK directory.

**7. src.zip File**

* **Purpose:** Contains the source code for the Java standard library (core Java classes).
* **Usage:** This is useful for developers who want to explore or learn from the implementation of Java's standard library classes.

**8. legal Directory (from JDK 9 onwards)**

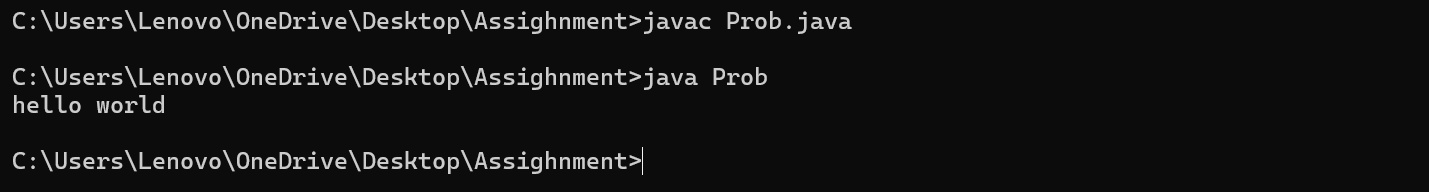
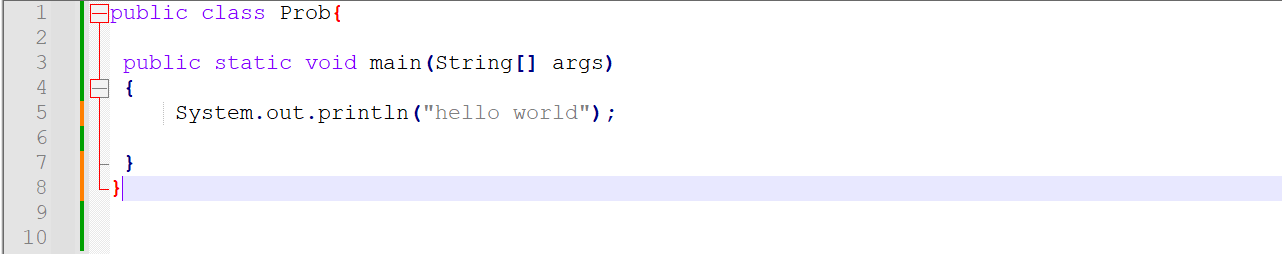
* **Purpose:** Contains legal notices for the various components included in the JDK.
* **Usage:** It's mainly for reference and compliance with software licensing.

**5. Reading Assignment: About Java Technology**

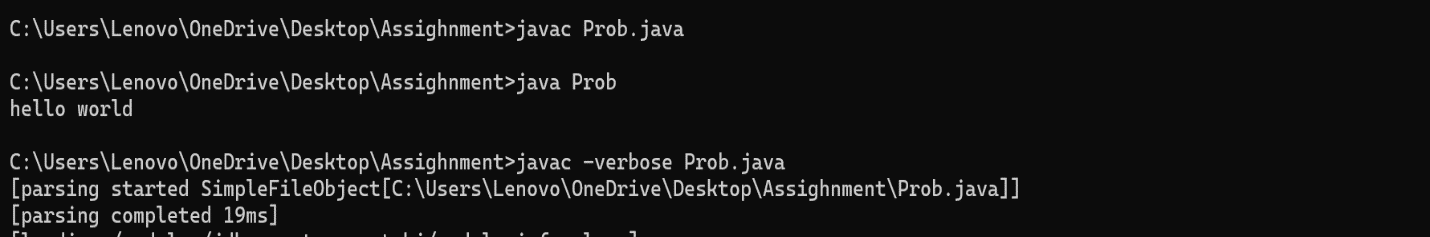
* **Task**: Read about the basics of Java technology and its components.
* **Link**: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

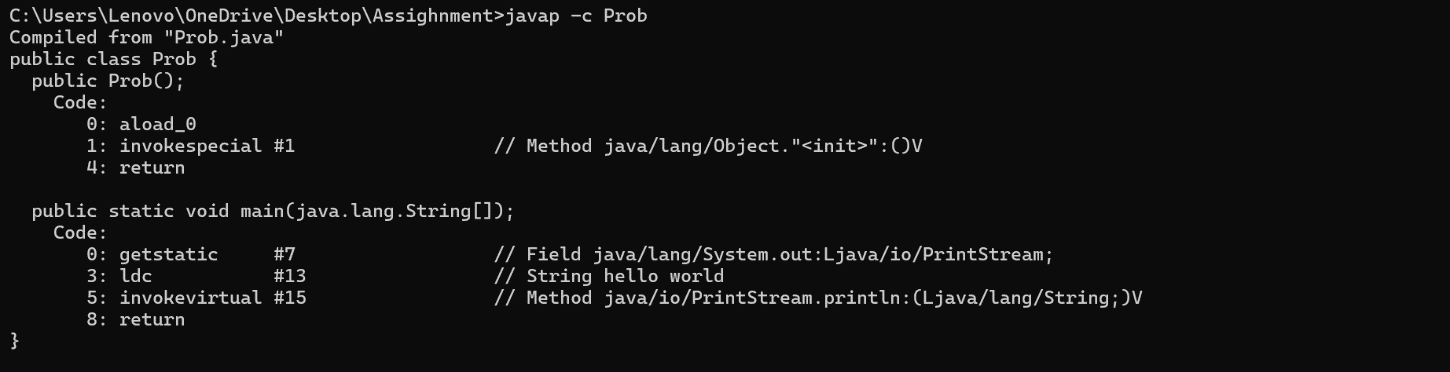
**6. Coding Assignments**

1. **Hello World Program**: Write a Java program that prints "Hello World!!" to the console.



1. **Compile with Verbose Option**: Compile your Java file using the -verbose option with javac. Check the output.



1. **Inspect Bytecode**: Use the javap tool to examine the bytecode of the compiled .class file. Observe the output.