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

* **Task**: Read about the history and development of Java.
* **Link**: <http://sunsite.uakom.sk/sunworldonline/swol-07-1995/swol-07-java.html>

Java was developed by James Gosling at Sun Microsystems in the early 1990s, initially named Oak. It was publicly released in 1995 as Java, promoting the "Write Once, Run Anywhere" (WORA) capability, allowing programs to run on any device with the Java Virtual Machine (JVM). Java quickly became popular for web development and enterprise applications. In 2007, it was open-sourced, and in 2010, Oracle acquired Sun Microsystems, taking over Java. Java has continued to evolve, with modern features like lambda expressions and records, maintaining its status as a leading programming language for various applications.

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

* **Task**: Learn about the main features of Java.
* **Link**: <https://javaalmanac.io/features/>

Java offers key features like **platform independence** through the Java Virtual Machine (JVM), enabling "Write Once, Run Anywhere" (WORA). It supports **object-oriented programming** with classes, inheritance, and polymorphism. Java provides **automatic memory management** via garbage collection, enhancing efficiency. **Multithreading** allows concurrent execution of tasks. Features like **generics** and **lambda expressions** improve code reusability and functional programming capabilities. **Security** is a priority, with a robust security manager and bytecode verification. The language also includes a rich **standard library** and is continually updated with modern features, such as records and enhanced switch expressions.

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

* **Task**: Find out which JDK version is right for you.
* **Link**: <https://whichjdk.com/>

Use the latest LTS version of the Java Development Kit (JDK) for stability; currently, it's Java 17, ideal for production with extended support. For cutting-edge features, consider the latest non-LTS version, like Java 21, but note the shorter support cycle. Always check your project's framework requirements to ensure compatibility. If maintaining an older project, you might need an older JDK, such as Java 8 or 11. For most new projects, Java 17 is recommended, balancing modern features and long-term support.

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

* **Task**: Understand the folder structure and files in the JDK installation.
* **Link**: <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/jdkfiles.html>

The JDK installation directory includes several key subdirectories:

* **bin/**: Executables like java and javac.
* **conf/**: Configuration files.
* **include/**: C/C++ headers for JNI.
* **lib/**: Core libraries and support files.
* **jmods/**: Java modules introduced in Java 9.
* **legal/**: Legal notices.
* **src.zip**: Source code for standard Java libraries.

Older versions may also include a **jre/** directory for the Java Runtime Environment, but in modern JDK versions (Java 11+), the JRE is integrated. This structure organizes all components needed for Java development efficiently.

jdk-17/

│

├── bin/

├── conf/

├── include/

├── jmods/

├── legal/

├── lib/

├── src.zip

└── man/

**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>

Java technology is a comprehensive platform composed of several key components:

- Java Language: High-level, object-oriented, and platform-independent.

- JDK: Development kit with a compiler, libraries, and tools.

- JRE: Runtime environment with the JVM and core libraries.

- JVM: Executes Java bytecode on any platform.

-Java SE: Core libraries and APIs for general-purpose development.

- Java EE: Enterprise APIs for large-scale applications.

- Java ME: Optimized for mobile and embedded systems.

- JavaFX: Platform for rich internet applications.

- Java Card: For secure applets on smart cards.

These components support a wide range of applications.

**6. Coding Assignments**

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

class Hello

{

public **static** void main(String args[]) {

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

}

}

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

**javac -verbose Hello.java**

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

**javap -c Hello**

**7. Reading Assignment: The JVM Architecture Explained**

* **Task**: Learn about how the Java Virtual Machine (JVM) works.
* **Link**: <https://dzone.com/articles/jvm-architecture-explained>

The Java Virtual Machine (JVM) enables Java bytecode execution across platforms. Key components include:

- Class Loader Subsystem: Loads and links classes.

- Runtime Data Areas: Includes Method Area (class data), Heap (objects), Stack (method calls), PC Register (instruction address), and Native Method Stack (native calls).

- Execution Engine: Consists of the Interpreter (executes bytecode), JIT Compiler (optimizes bytecode to native code), and Garbage Collector (manages memory).

- Native Method Interface (JNI): Allows Java to interact with native code.

- Native Method Libraries: Provides platform-specific libraries needed for native method execution.

This architecture supports Java’s portability and performance.

+-------------------------------------------+

| JVM |

| |

| +--------------+ +-------------------+ |

| | Class | | Native Method | |

| | Loader | | Interface (JNI) | |

| +--------------+ +-------------------+ |

| |

| +-------------------------------------+ |

| | Runtime Data Areas | |

| | +---------------------------------+ | |

| | | Method Area | | |

| | +---------------------------------+ | |

| | +---------------------------------+ | |

| | | Heap | | |

| | +---------------------------------+ | |

| | +---------------------------------+ | |

| | | Stack | | |

| | +---------------------------------+ | |

| | +---------------------------------+ | |

| | | Program Counter (PC) Register | | |

| | +---------------------------------+ | |

| | +---------------------------------+ | |

| | | Native Method Stack | | |

| | +---------------------------------+ | |

| +-------------------------------------+ |

| |

| +--------------+ +-------------------+ |

| | Execution | | Native Method | |

| | Engine | | Libraries | |

| | (Interpreter,| | | |

| | JIT, GC) | | | |

| +--------------+ +-------------------+ |

| |

+-------------------------------------------+

**8. Reading Assignment: The Java Language Environment: Contents**

* **Task**: Explore the content and features of the Java language environment.
* **Link**: <https://www.oracle.com/java/technologies/language-environment.html>

The Java language environment includes:

- Java Programming Language: Object-oriented, platform-independent, with features like automatic memory management and multithreading.

- JDK: Tools for development, including the compiler (`javac`), interpreter (`java`), debugger (`jdb`), and documentation generator (`javadoc`).

- JRE: Provides the JVM for executing bytecode and core libraries for essential functions.

- Java SE: Core APIs and utilities for general-purpose development.

- Java EE: Enterprise features like Servlets, JSP, and EJBs for scalable applications.

- Java ME: Optimized for mobile and embedded systems.

- JavaFX: For rich UI applications.

- Java Card: For smart cards and secure elements.