ASSIGNMENT NO.1

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 originally known as OAK in 1991 James gosling and team introduced java (Sun Microsystems)
-In 1992 The "*7" device was introducing to showcase the technology potential of java, but Time-Warner denied set-top box OS and video-on-demand technology for demo.
-in 1994 Webrunner (a web browser) also known as applet was introduced which ca run the java file.
- 1996 Java 1.0 was released publically
-in 2010 Sun microsystem was acquired by Oracle corporation

---

2. Reading Assignment: Java Language Features

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

Java 8:

- Remove Permanent Generation
- Default Methods in Interfaces
- Effectively Final Variables
- Type Use Annotations
- Repeating Annotations
- Streams (java.util.stream)
- Lambda APIs (java.util.function)
- Date Time (java.time)
- New Opcode

---

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

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

JAVA 8 LTS version

---

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

JDK software is installed at /jdk-1.8, here are some of the most important directories:

- **/jdk-1.8**
  Root directory of the JDK software installation. Contains copyright, license, and README files. Also contains src.zip, the archive of source code for the Java platform.
  /jdk-1.8/bin
  Executables for all the development tools contained in the JDK. The PATH environment variable should contain an entry for this directory.

- /jdk-1.8/lib
  Files used by the development tools. Includes tools.jar, which contains non-core classes for support of the tools and utilities in the JDK. Also includes dt.jar, the DesignTime archive of

BeanInfo files that tell interactive development environments (IDEs) how to display the Java components and how to let the developer customize them for an application.

- /jdk-1.8/jre
  Root directory of the Java Runtime Environment (JRE) used by the JDK development tools. The runtime environment is an implementation of the Java platform. This is the directory referred to by the java.home system property.

- /jdk-1.8/jre/bin
  Executable files for tools and libraries used by the Java platform. The executable files are identical to files in /jdk-1.8/bin. The java launcher tool serves as an application launcher. This directory does not need to be in the PATH environment variable.

- /jdk-1.8/jre/lib
  Code libraries, property settings, and resource files used by the JRE. For example rt.jar contains the bootstrap classes, which are the run time classes that comprise the Java platform core API, and charsets.jar contains the character-conversion classes. Aside from the ext subdirectory, there are several additional resource subdirectories not described here.

- /jdk-1.8/jre/lib/ext
  Default installation directory for extensions to the Java platform. This is where the JavaHelp JAR file goes when it is installed, for example. This directory includes the jfxrt.jar file, which contains the JavaFX runtime libraries and the localedata.jar file, which contains the locale data for the java.text and java.util packages. See The Extension Mechanism.

- /jdk-1.8/jre/lib/security
  Contains files used for security management. These include the security policy java.policy and security properties java.security files.

- /jdk-1.8/jre/lib/applet
  JAR files that contain support classes for applets can be placed in the lib/applet/ directory. This reduces startup time for large applets by allowing applet classes to be preloaded from the local file system by the applet class loader and provides the same protections as though they had been downloaded over the Internet.

- /jdk-1.8/jre/lib/fonts
  Font files used by the platfo

---

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

The Java programming language is a high-level language that can be characterized as follows

ASSIGNMENT NO.1

Simple	Architecture neutral
Object oriented	Portable
Distributed	High performance
Multithreaded	Robust
Dynamic	Secure

---

6. Coding Assignments

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

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

```
D:\Notepad++ files\CODES>java demo
Hello World!!
```

==========

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

```
D:\Notepad++ files\CODES>javac -verbose demo.java
[parsing started SimpleFileObject[D:\Notepad++ files\CODES\demo.java]]
[parsing completed 36ms]
[loading /modules/jdk.crypto.cryptoki/module-info.class]
[loading /modules/jdk.nio.mapmode/module-info.class]
[loading /modules/java.rmi/module-info.class]
[loading /modules/java.xml/module-info.class]
[loading /modules/jdk.jcmd/module-info.class]
[loading /modules/java.logging/module-info.class]
[loading /modules/jdk.accessibility/module-info.class]
[loading /modules/jdk.javadoc/module-info.class]
[loading /modules/jdk.httpserver/module-info.class]
[loading /modules/jdk.internal.vm.compiler/module-info.class]
[loading /modules/jdk.jstatd/module-info.class]
[loading /modules/java.base/module-info.class]
[loading /modules/jdk.internal.opt/module-info.class]
[loading /modules/jdk.jlink/module-info.class]
[loading /modules/jdk.internal.jvmstat/module-info.class]
[loading /modules/jdk.crypto.ec/module-info.class]
[loading /modules/jdk.net/module-info.class]
[loading /modules/jdk.security.jgss/module-info.class]
[loading /modules/java.scripting/module-info.class]
```

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

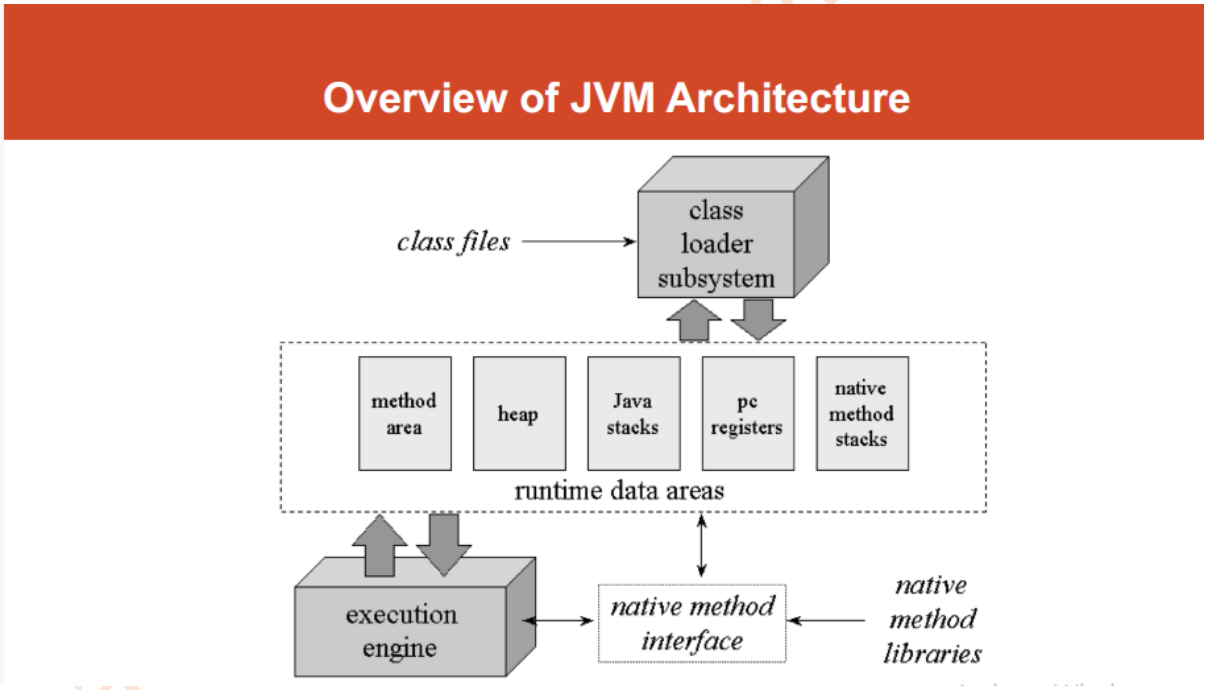
```
PS C:\Users\rahul\OneDrive\Desktop\CDAC\Logic bulding\Day 2> javap -c Demox.class
Compiled from "Demox.java"
public class Demox {
  public Demox();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #13                 // String hello world
       5: invokevirtual #15                 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

==========

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 JVM is divided into three main subsystems:

1. ClassLoader Subsystem
2. Runtime Data Area
3. Execution Engine

1. ClassLoader Subsystem

It loads, links. and initializes the class file when it refers to a class for the first time at runtime, not compile time.

1 Loading

Classes will be loaded by this component. BootStrap ClassLoader, Extension ClassLoader, and Application ClassLoader are the three ClassLoaders that will help in achieving it.

a. BootStrap ClassLoader – Responsible for loading classes from the bootstrap classpath, nothing but rt.jar. Highest priority will be given to this loader.

b. Extension ClassLoader – Responsible for loading classes which are inside the ext folder (jre\lib).

c. Application ClassLoader –Responsible for loading Application Level Classpath, path mentioned Environment Variable.

2 Linking

a. **Verify** – Bytecode verifier will verify whether the generated bytecode is proper or not if verification fails we will get the verification error.
b. **Prepare** – For all static variables memory will be allocated and assigned with default values.
c. **Resolve** – All symbolic memory references are replaced with the original references from Method Area.

3 Initialization

This is the final phase of Class Loading; here, all static variables will be assigned with the original values, and the static block will be executed.

2. Runtime Data Area

The Runtime Data Area is divided into five major components:

1.Method Area – All the class-level data will be stored here, including static variables. There is only one method area per JVM, and it is a shared resource.

2.Heap Area – All the Objects and their corresponding instance variables and arrays will be stored here. There is also one Heap Area per JVM. Since the Method and Heap areas share memory for multiple threads, the data stored is not thread-safe.

3.Stack Area– For every thread, a separate runtime stack will be created. For every method call, one entry will be made in the stack memory which is called Stack Frame. All local variables will be created in the stack memory. The stack area is thread-safe since it is not a shared resource. The Stack Frame is divided into three subentities:

1. **Local Variable Array** – Related to the method how many local variables are involved and the corresponding values will be stored here.
2. **Operand stack** – If any intermediate operation is required to perform, operand stack acts as runtime workspace to perform the operation.
3. **Frame data** – All symbols corresponding to the method is stored here. In the case of any **exception**, the catch block information will be maintained in the frame data.

4.PC Registers – Each thread will have separate PC Registers, to hold the address of current executing instruction once the instruction is executed the PC register will be updated with the next instruction.

5.Native Method stacks – Native Method Stack holds native method information. For every thread, a separate native method stack will be created.

3. Execution Engine

The bytecode, which is assigned to the **Runtime Data Area,** will be executed by the Execution Engine. The Execution Engine reads the bytecode and executes it piece by piece.

1. **Interpreter** – The interpreter interprets the bytecode faster but executes slowly. The disadvantage of the interpreter is that when one method is called multiple times, every time a new interpretation is required.
2. **JIT Compiler**– The JIT Compiler neutralizes the disadvantage of the interpreter. The Execution Engine will be using the help of the interpreter in converting byte code, but when it finds repeated code it uses the JIT compiler, which compiles the entire bytecode and changes it to native code. This native code will be used directly for repeated method calls, which improve the performance of the system.
    1. **Intermediate Code Generator** – Produces intermediate code
    2. **Code Optimizer** – Responsible for optimizing the intermediate code generated above
    3. **Target Code Generator** – Responsible for Generating Machine Code or Native Code
    4. **Profiler** – A special component, responsible for finding hotspots, i.e. whether the method is called multiple times or not.

3. **Garbage Collector**: Collects and removes unreferenced objects. Garbage Collection can be triggered by calling System.gc(), but the execution is not guaranteed. Garbage collection of the JVM collects the objects that are created.

Java Native Interface (JNI): JNI will be interacting with the Native Method Libraries and provides the Native Libraries required for the Execution Engine.

Native Method Libraries: This is a collection of the Native Libraries, which is required for the Execution Engine.

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

Features of java:

1. Java is Object Oriented
2. Architecture Neutral, Portable, and Robust
3. Interpreted and Dynamic
4. Security in Java
5. Multithreading
6. Performance and Comparisons
7. Java Base System and Libraries

ASSIGNMENT NO.1