Assignment 1 - Sandeep Sir

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

Java was created at Sun Microsystems, Inc., where James Gosling led a team of researchers in an effort to create a new language that would allow consumer electronic devices to communicate with each other. Work on the language began in 1991, and before long the team's focus changed to a new niche, the World Wide Web.

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

the main features of Java, a popular programming language used for coding web applications and more:

Platform independence: Java programs can be run on any operating system or machine without modification, as long as they were compiled on one.

Object-oriented: Java is a fully object-oriented language, which means it emphasizes the use of classes and objects.

Simple and easy to learn: Java's syntax is designed to be easy to understand.

Secure and robust: Java's design prioritizes reliability and robustness.

Multithreading: Java supports multithreading, which allows multiple threads to execute concurrently within a program.

Automatic memory management: Java automatically manages memory.

Rich API: Java has a rich API.

High performance: Java is a high performance programming language.

Compiled and interpreted: Java offers both compilation and interpretation of programs.

Dynamic and extensible: Java is dynamic and extensible

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

Non-trivial, stable applications in production should use an LTS release.

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

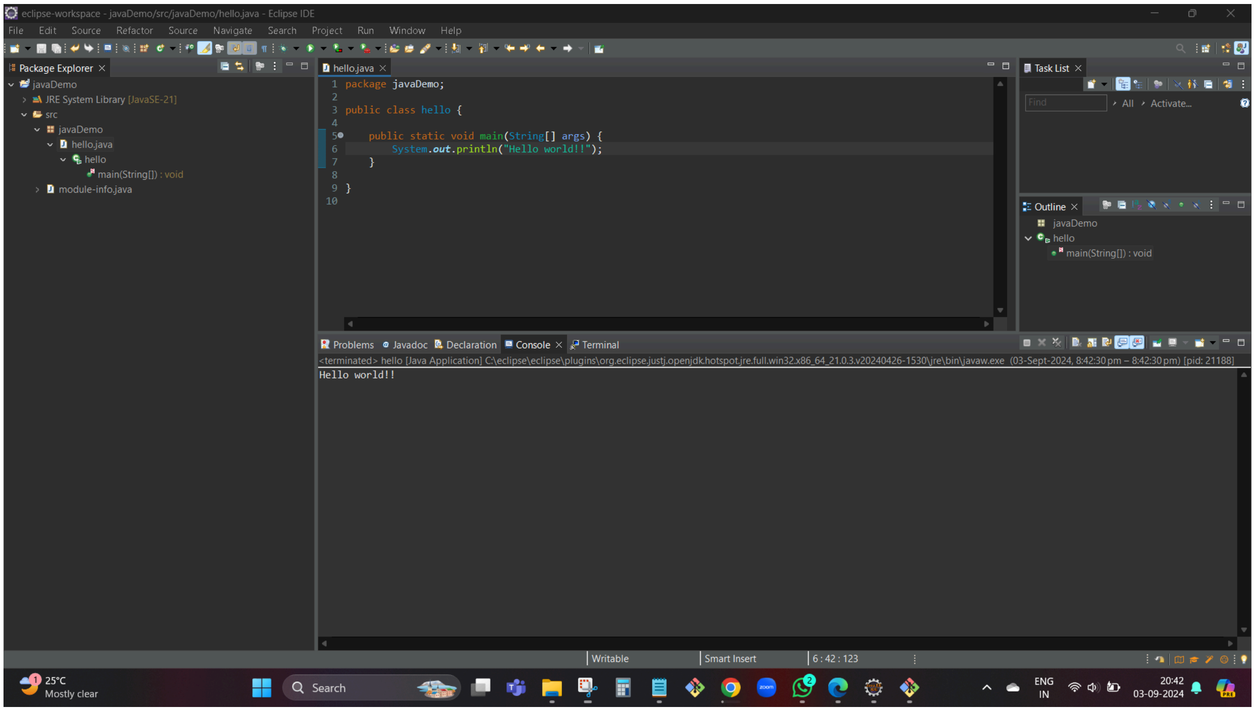
The locations of these folders vary depending on your system, but in all versions of Windows, the JDK root folder is in the path Program Files\Java on your boot drive. The name of the JDK root folder also varies, depending on the Java version you've installed. For version 1.7, the root folder is jdk1. 7.0 .

5. Task: Read about the basics of Java technology and its components.

There are three main components of the Java programming language, including: JVM: The Java Virtual Machine or JVM is a platform-independent Java component that provides an environment for executing Java programs. So, JVM loads the code, validates the code, executes the code, and provides a runtime environment.

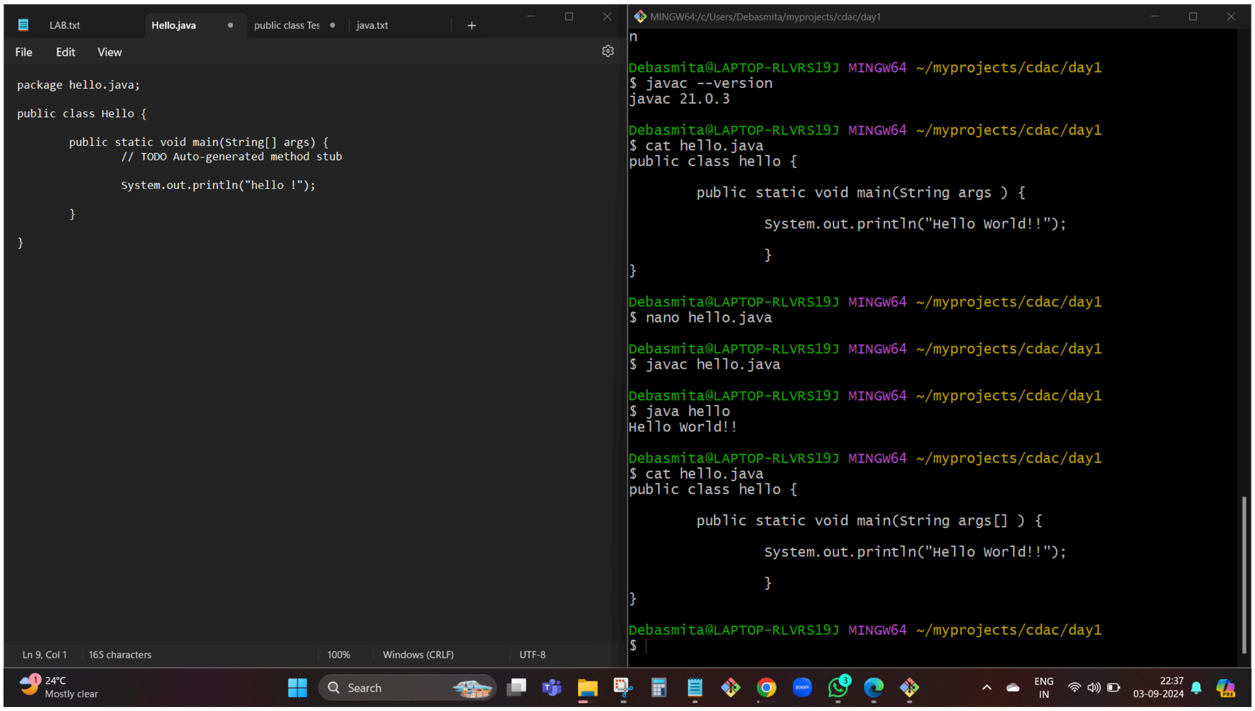
6. Coding Assignments

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


eclipse-workspace - javaDemo/src/javaDemo/hello.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer
javaDemo
JRE System Library [JavaSE-21]
src
javaDemo
hello.java
hello
main(String[]) : void
module-info.java
hello.java
Task List
Outline
Problems Javadoc Declaration Console Terminal
Writable Smart Insert 6 : 42 : 123
25°C Mostly clear
Search
ENG IN
20:42 03-09-2024

```java
package javaDemo;

public class hello {

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

}
```

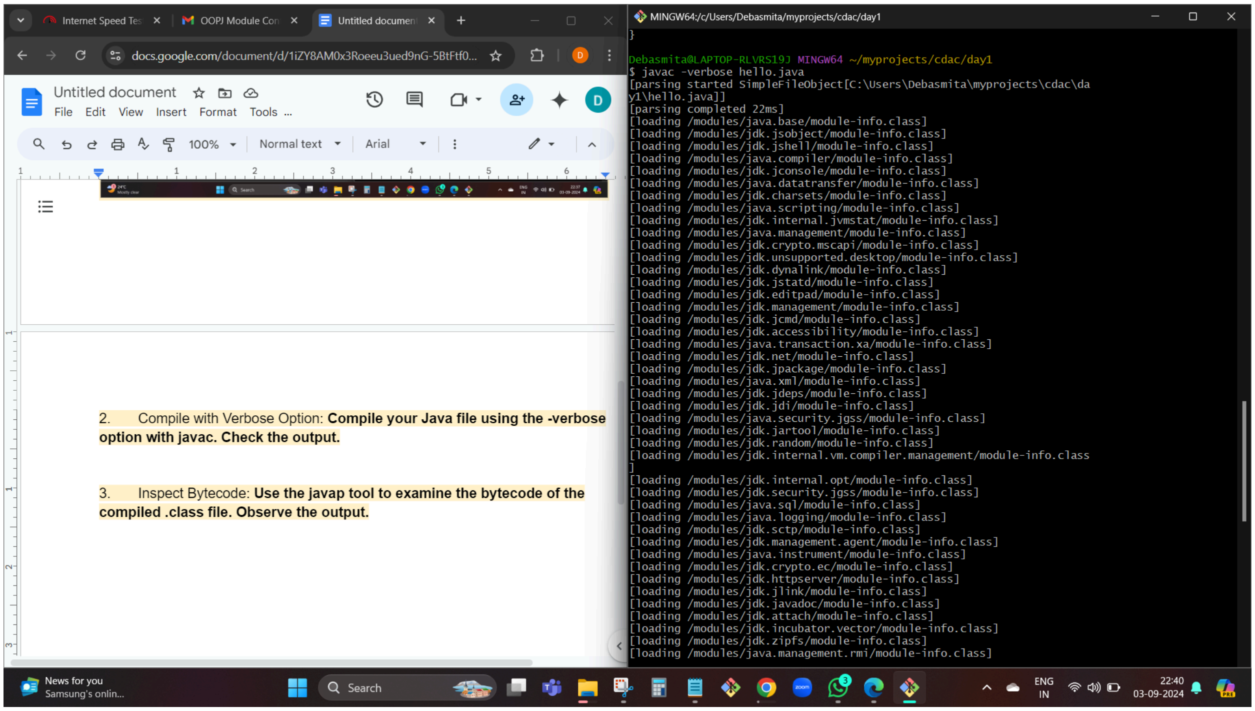
```
<terminated> hello [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (03-Sept-2024, 8:42:30 pm – 8:42:30 pm) [pid: 21188]
Hello world!!
```



```java
package hello.java;

public class Hello {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                System.out.println("hello !");

        }

}
```

```
n

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javac --version
javac 21.0.3

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ cat hello.java
public class hello {

        public static void main(String args ) {

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

                }
}

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ nano hello.java

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javac hello.java

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ java hello
Hello World!!

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ cat hello.java
public class hello {

        public static void main(String args[] ) {

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

                }
}

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$
```

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



Untitled document
docs.google.com/document/d/1iZY8AM0x3Roeeu3ued9nG-5BtFtf0...
MINGW64:/c/Users/Debasmita/myprojects/cdac/day1

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

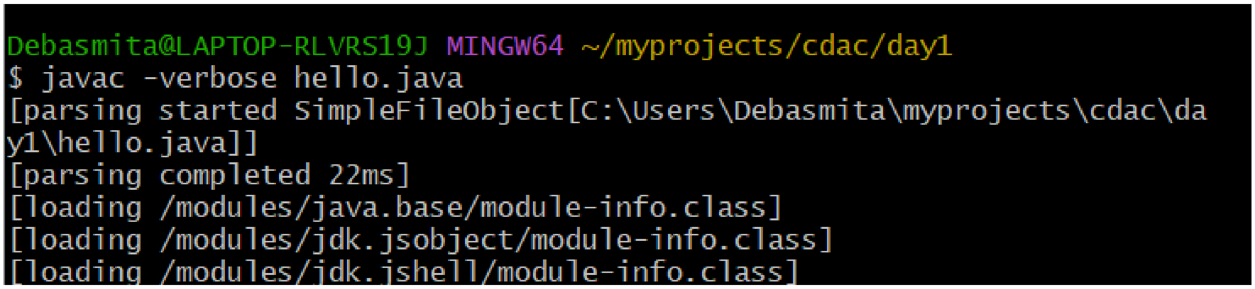
3. Inspect Bytecode: **Use the javap tool to examine the bytecode of the compiled .class file. Observe the output.**

```
}

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javac -verbose hello.java
[parsing started SimpleFileObject[C:\Users\Debasmita\myprojects\cdac\da
y1\hello.java]]
[parsing completed 22ms]
[loading /modules/java.base/module-info.class]
[loading /modules/jdk.jsobject/module-info.class]
[loading /modules/jdk.jshell/module-info.class]
[loading /modules/java.compiler/module-info.class]
[loading /modules/jdk.jconsole/module-info.class]
[loading /modules/java.datatransfer/module-info.class]
[loading /modules/jdk.charsets/module-info.class]
[loading /modules/java.scripting/module-info.class]
[loading /modules/jdk.internal.jvmstat/module-info.class]
[loading /modules/java.management/module-info.class]
[loading /modules/jdk.crypto.mscapi/module-info.class]
[loading /modules/jdk.unsupported.desktop/module-info.class]
[loading /modules/jdk.dynalink/module-info.class]
[loading /modules/jdk.jstatd/module-info.class]
[loading /modules/jdk.editpad/module-info.class]
[loading /modules/jdk.management/module-info.class]
[loading /modules/jdk.jcmd/module-info.class]
[loading /modules/jdk.accessibility/module-info.class]
[loading /modules/java.transaction.xa/module-info.class]
[loading /modules/jdk.net/module-info.class]
[loading /modules/jdk.jpackage/module-info.class]
[loading /modules/java.xml/module-info.class]
[loading /modules/jdk.jdeps/module-info.class]
[loading /modules/jdk.jdi/module-info.class]
[loading /modules/java.security.jgss/module-info.class]
[loading /modules/jdk.jartool/module-info.class]
[loading /modules/jdk.random/module-info.class]
[loading /modules/jdk.internal.vm.compiler.management/module-info.class
]
[loading /modules/jdk.internal.opt/module-info.class]
[loading /modules/jdk.security.jgss/module-info.class]
[loading /modules/java.sql/module-info.class]
[loading /modules/java.logging/module-info.class]
[loading /modules/jdk.sctp/module-info.class]
[loading /modules/jdk.management.agent/module-info.class]
[loading /modules/java.instrument/module-info.class]
[loading /modules/jdk.crypto.ec/module-info.class]
[loading /modules/jdk.httpserver/module-info.class]
[loading /modules/jdk.jlink/module-info.class]
[loading /modules/jdk.javadoc/module-info.class]
[loading /modules/jdk.attach/module-info.class]
[loading /modules/jdk.incubator.vector/module-info.class]
[loading /modules/jdk.zipfs/module-info.class]
[loading /modules/java.management.rmi/module-info.class]
```

The -verbose option in javac is used to provide detailed output during the compilation process. It lists all the classes and resources that are being loaded by the compiler.

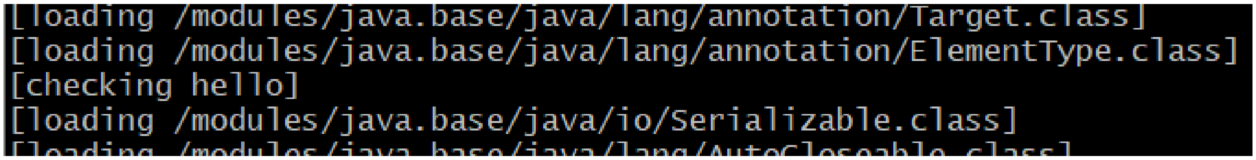
Key Parts of the Output:

- **[loading]: Shows the loading of classes and resources.**



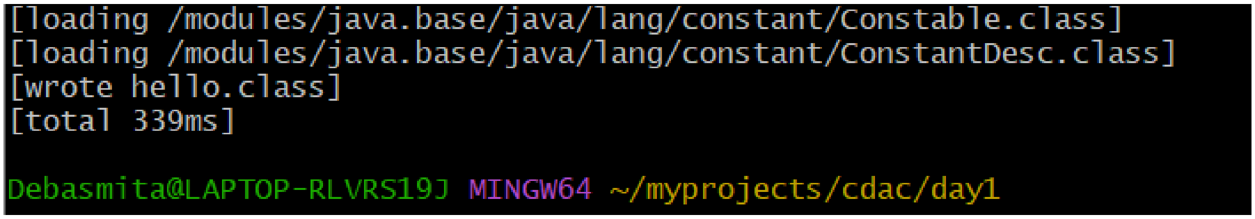
```
Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javac -verbose hello.java
[parsing started SimpleFileObject[C:\Users\Debasmita\myprojects\cdac\da
y1\hello.java]]
[parsing completed 22ms]
[loading /modules/java.base/module-info.class]
[loading /modules/jdk.jsobject/module-info.class]
[loading /modules/jdk.jshell/module-info.class]
```

- **[parsing started/completed]: Indicates the start and completion of parsing your source code.**
- **[checking]: Refers to the checking of different types and symbols during compilation.**



```
[loading /modules/java.base/java/lang/annotation/Target.class]
[loading /modules/java.base/java/lang/annotation/ElementType.class]
[checking hello]
[loading /modules/java.base/java/io/Serializable.class]
[loading /modules/java.base/java/lang/AutoCloseable.class]
```

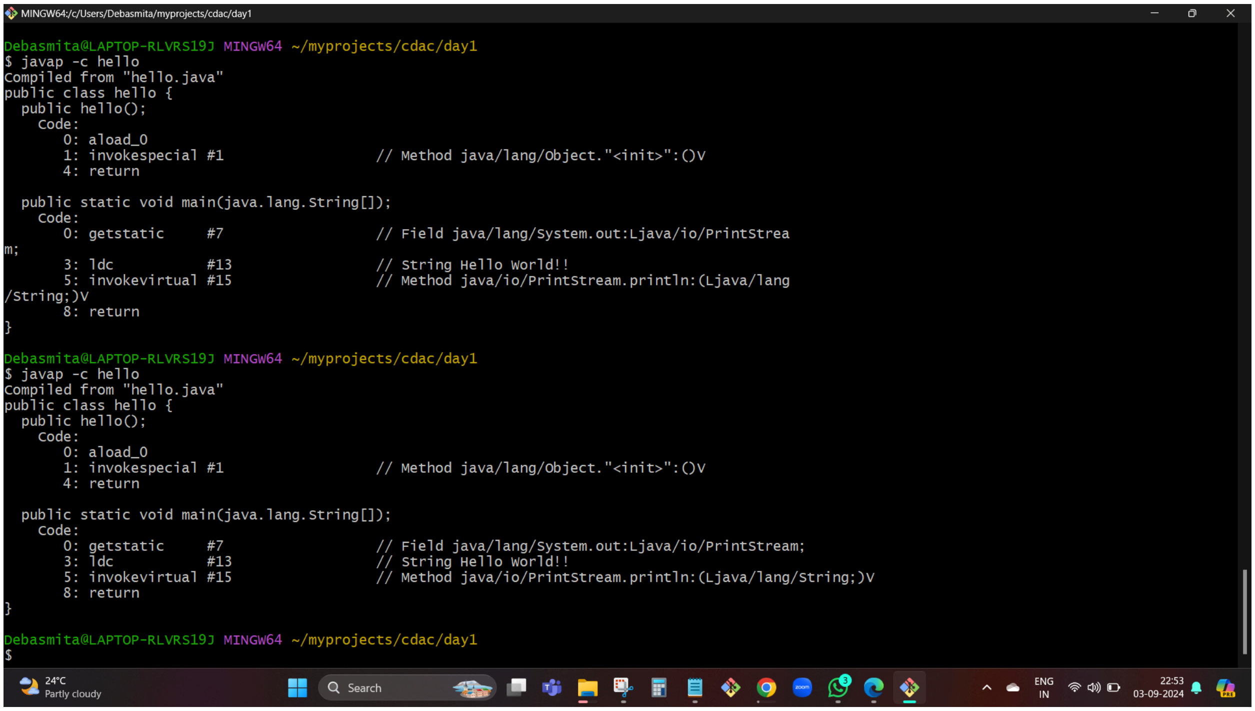
- **[wrote]: Confirms that the class file has been written to the disk.**



```
[loading /modules/java.base/java/lang/constant/Constable.class]
[loading /modules/java.base/java/lang/constant/ConstantDesc.class]
[wrote hello.class]
[total 339ms]

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
```

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

The javap tool is used to disassemble class files and examine the bytecode that the Java compiler generates.



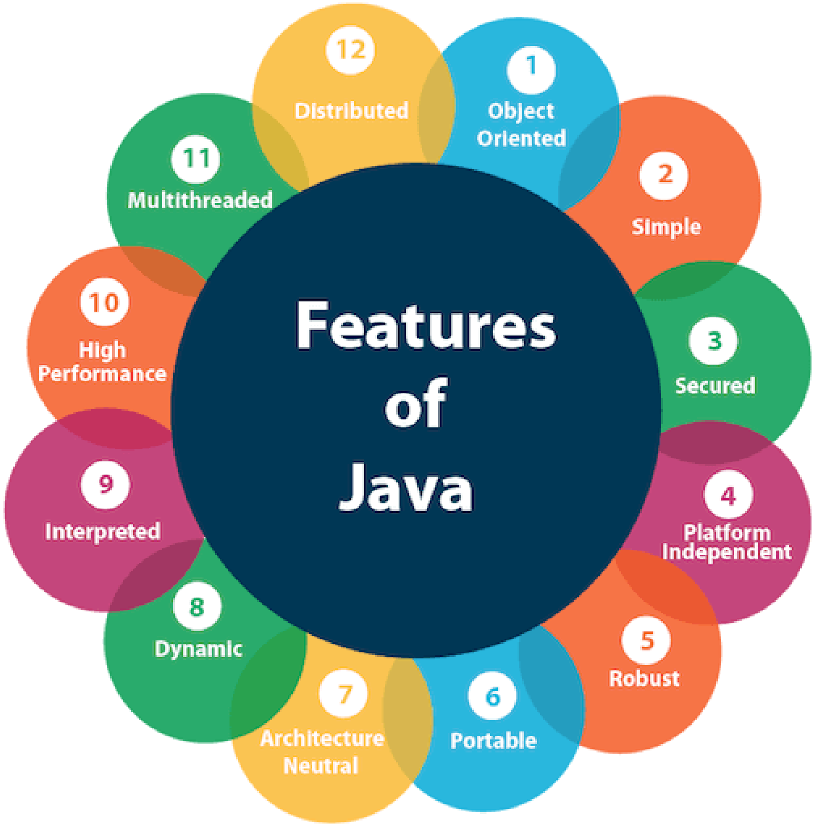
```
Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javap -c hello
Compiled from "hello.java"
public class hello {
  public hello();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

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

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$ javap -c hello
Compiled from "hello.java"
public class hello {
  public hello();
    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
}

Debasmita@LAPTOP-RLVRS19J MINGW64 ~/myprojects/cdac/day1
$
```

7. Task: Learn about how the Java Virtual Machine (JVM) works.

A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode. The JVM is detailed by a specification that formally describes what is required in a JVM implementation.

8. Task: Explore the content and features of the Java language environment.

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed in Java development.



1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic