//

IJSE

Institute of Software Engin

Graduate Diploma in Software Engineering

Batch - GDSE69

Module - Programming Fundamentals

Name : Shimara Appuhami

Nic : 200362310434

//
/

Assignment 01

1. Sarah is a computer science student who is working on her Java programming assignment. She
needs to create a Java program. What are the basic steps that Sarah should follow to create and run
her Java program.

- ➢ Install Java Development Kit (JDK)
- ➢ Use notepad and to write source code
- ➢ should create a new Java source file with a `java` extension
- ➢ public class Example {

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

- ➢ save the Java source file
- ➢ use the ‘`javac`’ command and compile
- ➢ Use ‘Java’ command and run the program

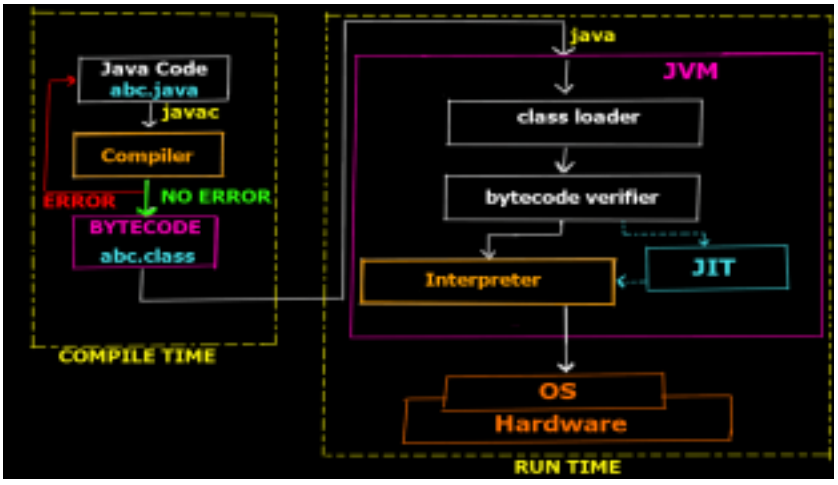
//

2. You are working in a Linux environment and have a Java program named "MyProgram.java" that
you want to compile and run. What are the specific commands you would use to achieve this.

- ➢ Open terminal and open your working directory
- ➢ Open gedit
- ➢ Create a source file named MyPrograme.java
- ➢ Write your java program and save it
- ➢ Compile the Java Program>>>>>>>>>>>javac MyProgram.java
- ➢ Run the Java Program>>>>>>>>>>>>java MyProgram

//

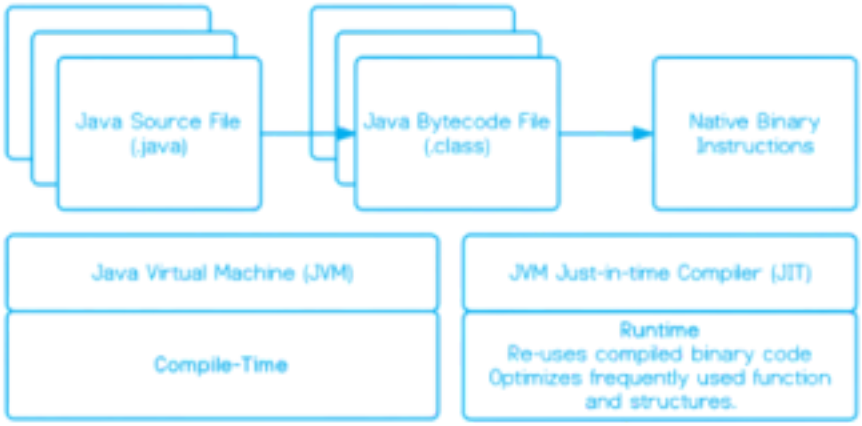
3. How does a Java program work? Explain with an appropriate diagram.



//

4. What are the key differences between Java source files and compiled Java files?

Compilation is the process of converting the Java source code into an executable form, known as bytecode.	**Compilation Interpretation Interpretation is the process of executing the Java bytecode directly by the JVM.**
The Java compiler takes the source code and produces bytecode, which is then executed by the Java Virtual Machine (JVM).	The bytecode is loaded into the JVM, and the JVM interprets the bytecode and executes the program.
The code cannot be compiled because of compilation errors.	Debugging is mainly done in run-time.
Compared to interpreted compiled programs operate faster.	When an interpreted program executes, changes can be made.



5. Compare and contrast the main differences between Java, C, and Python in terms of execution.

C++

History: C++ was developed by Bjarne Stroustrup along with Dennis Ritchie (creator of C) in Bell Laboratory during the 1970s. C++ was created as an extension of C, adding object-oriented

features.
Popularity-wise, C++ is the second oldest programming language and ranked 4th in the TIOBE programming language ranking.

Features

- C++ is a fast and compiled programming language. Because it's a compiled programming language, C++ is **Platform dependent**.
- Using C++, the programmer can gain **full control over the hardware** as it has many libraries for directly handling the hardware-level tasks.
- C++ can quickly adapt to take advantage of **Hardware changes**, so its also gaining popularity along with the sudden rise of containerisation, GPUs, and Cloud computing.
- Because it is **super-fast**, C++ is heavily used in performance-critical and resource-constrained systems.

Use-Cases

C++ is widely used among programmers as it supports Object-oriented programming as well as gives access to hardware. Using C++, many high-level end-user applications can be developed:

- **GUI Based Applications:** Adobe Photoshop. Illustrator and WinAmp Media Player are developed using C++.
- **Operating Systems**: Apple OS has some parts written in C++. Most software from Microsoft, like Visual Studio IDE and Internet Explorer, are also developed using C++.
- **Browsers**: Because of the faster execution time of C++, C++ is widely used in browsers for rendering purposes.Examples Google File System and Chrome Browser and Mozilla Thunderbird
- **Cloud/Distributed Systems**: C++ is a good choice for the implementation of Cloud Systems as it is close to hardware and also provide multithreading support.

Java

History

Java was developed in the early 1990s by James Gosling as an object-oriented language. The principles for creating Java were, "Simple, Robust, Portable, Platform-Independent, Secured, High-Performance, Multithreaded, Architecture Neutral and Dynamic".

Popularity wise, Java is the third most popular programming language right after five years of its

release till now (even after two decades).
Features

- Java is platform-independent, it is the first programming language that achieved the, “Write Once, Run anywhere” title
- Java supports automatic garbage collection and memory management, which manages the object life cycle.
- Java supports multithreading which allows writing programs that can execute many tasks simultaneously.
- Java enables high performance using Just-In-Time java online compilers, also Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to different objects at run-time.

Use-Cases

- **Mobile App Development**: Most android applications are built using Java, even the most popular android app development IDE, ‘Android Studio also uses Java to develop Android applications.
- **Web-Based Applications:** Java provides vast support for web development using Servlets, Struts, and JSP.
- **Big Data Technology**: Hadoop HDFS platform, which is used for processing and storing big data applications, is written in Java. Java is also used in Apache Camel and Apache Kafka
- **Game Development**: Java provides the support of an open-source 3D engine, so Java is extensively used in Game development. Many popular games, including Minecraft and Mission Impossible III, are written using Java.

You can also read about the topic of Java Destructor and Hashcode Method in Java.

Python

History:Python was designed by Guido van Rossum in the 1990s as a side project and developed by Python Software Foundation. It was named after the BBC’s TV show – “Monty Python’s Flying Circus”.

Python focussed on developer experience and is highly productive, simple, and yet very powerful and is incredibly popular. TIOBE has ranked Python as the third most popular programming language.
Features

- Python is an expressive language, meaning it can perform many complex tasks using very few lines of code.
- Python is an interpreted language. It makes debugging a lot easier.
- Python has a lot of libraries and modules, many complex tasks can be executed very simply using these libraries and built-in functions. **As of now, there are more than 137,000 python libraries**.
- Python is extensively used in Data Science, Machine Learning, Natural Language Processing, and deep learning.

Use-Cases

- **Web Scraping:** Web scraping involves the scraping of a massive amount of data from the web. Such data is useful for corporations and is extensively used in a Lead generation for Marketing. Python's Selenium, PythonRequest, and Mechanical Soup are used to build web scraping applications.
- **Data Science:** Python has numerous powerful libraries like scikit-learn and TensorFlow, making it incredibly popular for advanced data work, including data extraction, data mining, and data visualisation.
- **Machine learning:** Nowadays, most e-commerce websites offer enhanced user experience and improved search functionalities using Machine Learning and Artificial Intelligence using Python.

Read about, Fibonacci Series in Python and Swap Function in Java

//

6. How does Java achieve platform independence?

➢ ava is a platform-independent language, meaning we run the same code on multiple platforms. Java achieves this **using JVM and Byte Code**. Java compiler converts the programming code into byte code. Byte code is platform-independent and can be run on any processor or system

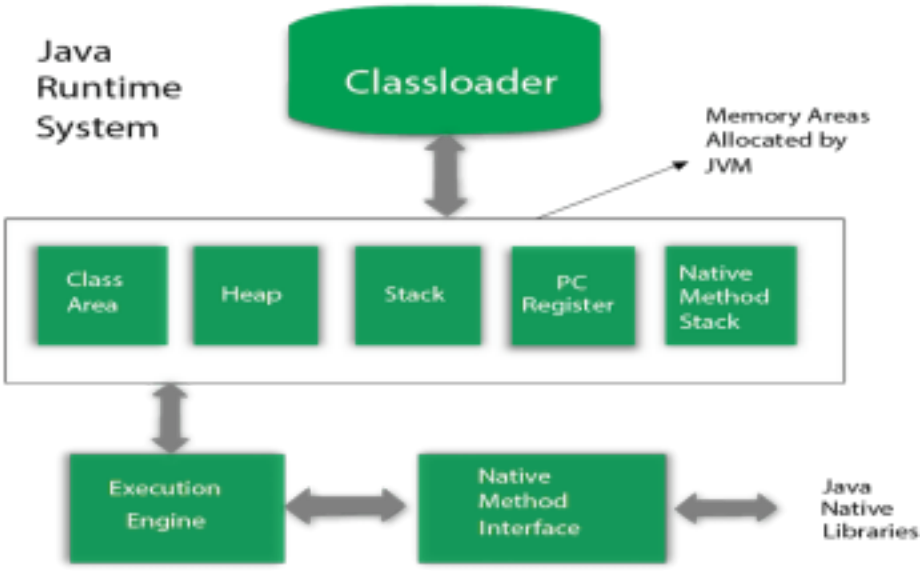
//

7. What is the role of the Java Virtual Machine (JVM)?

- ➢ JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.
- ➢ JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

JVM Architecture

- ➢ Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



8. Distinguish between a compiler and an interpreter. How do they process source code differently?

Parameter	Compiler	Interpreter
Program scanning	Compilers scan the entire program in one go.	The program is interpreted/translated one line at a time.
Error detection	As and when scanning is performed, all the errors are shown in the end together, not line by line.	
Object code	Compilers convert the source code	

to object code.		object code.
Execution time The execution time of compiler is less, hence it is preferred. One line of code is scanned, and errors encountered are shown.		It is not preferred due to its slow speed. Usually, interpreter is slow, and hence takes more time to execute the object code.
Interpreters do not convert the source code into		
Need of source code	Programming languages that use compilers include C, C++, C#, etc..	Programming languages that uses interpreter include Python, Ruby, Perl, MATLAB, etc.
Programming languages		
Types of errors detected	Compiler can check syntactic and semantic errors in the program simultaneously.	Interpreter checks the syntactic errors only.
Compiler doesn't require the source code for execution later.	It requires the source code for execution later.	
Size Compiler are larger in size. Interpreters are smaller in size. Flexibility Compilers		
are not flexible. Interpreters are relatively flexible. Efficiency Compilers are more		
efficient. Interpreters are less efficient.		

//

9. What is the difference between Java interpreter (in JVM) and O/S interpreter (Command Interpreter)?

- ➢ **CLI Interpreter** The CLI interpreter has different syntaxes for different operating system. It normally is used to run other programs, and only a small amount of time is actually spent in processing the CLI statements.
- ➢ **Java Virtual Machine with JVM** The Java compiler converts the Java source code into

an intermediate form called Java class files. The class files are the same for all operating systems. A great deal of checking for syntax errors and inconsistencies is done while creating the class file. The Java interpreter reads and executes the instructions in the class files.

//

10. Explain the concept of a "just-in-time" (JIT) compiler. In which programming languages is it commonly used?

➢ A "just-in-time" (JIT) compiler is a type of compiler that translates code from a high-level programming language or intermediate representation (such as bytecode) into machine code or a lower-level representation at runtime, specifically just before the code is executed. The primary purpose of a JIT compiler is to improve the execution speed of programs and to combine some of the advantages of both traditional compilation and interpretation.

➢ commonly used

Java
C#:
JavaScript
Python
Ruby

//

11. Ravi is a Java developer who has created a Java program on his Windows-based computer. He
shared the compiled Java program (Class File) with his colleague Priya, who uses a MacBook for
development. However, when Priya tries to run the program on her MacBook, she encounters issues,
and the program doesn't execute as expected.

a. Identify and explain the possible reasons why the Java program created by Ravi on his Windows computer may not work on Priya's MacBook.

1. **Platform-Dependent Code**: One common reason for compatibility issues is platform-dependent code. If Ravi's Java program includes platform-specific file paths, system-dependent behavior, or uses non-portable features, it may not work correctly on Priya's MacBook.
2. **Java Version**: If Ravi compiled the Java program using a version of Java that is not compatible with the version of Java installed on Priya's MacBook, it could result in

issues. Java programs compiled with a newer version of Java may not run on older Java runtime environments (JREs).

b. What steps can Priya take to troubleshoot and resolve the issue to make Ravi's Java program work on her MacBook?

1. **Check Java Version**: Priya should ensure that she has the correct version of Java installed on her MacBook that matches the version used by Ravi. She can check the Java version by running the following command in the terminal

```
java -version
```

2. **Recompile the Code**: To ensure compatibility, Priya can recompile the Java source code on her MacBook using the Java compiler (`javac`). This ensures that the bytecode is generated for the macOS environment.
3. **Check Line Endings**: Priya can use a text editor or a command-line tool to convert line endings in the source code to the appropriate format (LF for macOS). Tools like `dos2unix` or `unix2dos` can help with this conversion.
4. **Inspect Platform-Dependent Code**: If the code contains platform-specific logic, Priya should review and adapt it to work correctly on macOS. She may need to replace Windows-specific file paths or system calls with macOS equivalents.
5. **Verify Classpath**: Priya should verify that the classpath is correctly configured, including the paths to any external libraries or dependencies used by the program. She can use the `java -cp` option to specify the classpath if needed.
6. **Check for Case Sensitivity**: If the program relies on file or class names with a specific case, Priya should ensure that the case sensitivity matches macOS conventions. Renaming files or classes to match the expected case may be necessary.

7. **Debug and Test**: Priya can use debugging tools and step through the code to identify specific issues. Debugging can help pinpoint the root causes of runtime errors. 8. **Consult Documentation and Resources**: Priya can consult Java documentation, forums, or online resources for guidance on resolving platform-specific compatibility issues. The Java community often provides solutions to common cross-platform problems.

By following these steps and addressing platform-specific issues, Priya can increase the chances of making Ravi's Java program work correctly on her MacBook.

//

12. List the features of the Java Programming language?

Features of Java

The primary objective of Java programming language creation was to make it a portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

A list of the most important features of the Java language is given below.

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

//

13. Explain the concept of a "main" method in Java. Why is it necessary?

➢ It is the starting point of execution when you run a Java program. The `main` method has a specific signature and structure that the Java Virtual Machine (JVM) expects

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

// Your Java code goes here
}

- ➢ `public`: This keyword indicates that the `main` method is accessible from outside of the class. It allows the JVM to call this method to start the program.
- ➢ `static`: the `main` method belongs to the class itself rather than an instance of the class. This is necessary because when a Java program starts, no objects of the class have been created yet. The `static` keyword allows the method to be called without creating an instance of the class.
- ➢ `void`: This is the return type of the `main` method, indicating that it does not return any value. The `main` method is meant to execute code and initiate the program, not return a result.
- ➢ `main`: This is the name of the method and is fixed. It cannot be changed. ➢ `(String[] args)`: This is the parameter list for the `main` method. It accepts an array of strings (`String[]`) called `args`

The main method is necessary because it provides a standard entry point for Java applications. When you run a Java program, the JVM looks for the main method in the class you specify as the entry point (usually specified on the command line or in the project configuration). Once found, the JVM starts executing the code inside the main method, and this is where your program's logic begins.

Without a main method, the JVM wouldn't know where to start executing your code, and your Java program wouldn't be able to run as an independent application. It's essentially the gatekeeper of your Java program, enabling it to be launched and executed.

//

14. Which of the following “main” method declarations are valid (Runs without errors)?

a. public static void main(String args[]){ }

- ➢ valid

b. public void main(String args[]){}

- ➢ invalid

c. static void main(String args[]){}

- ➢ valid

d. public static void main(String args){ }

➢ invalid

e. void main(String args[]){ }

➢ invalid

f. public static void main(){ }

➢ valid

g. static public void main(String args[]){ }

➢ valid

h. void main(String args){ }

➢ invalid

i. public static main(String args){ }

➢ invalid

j. public static void main(String []){ }

➢ valid

k. static void public main(String args){ }

➢ invalid

//

15. Explain JVM, JDK, JRE.

➢JVM

JVM (Java Virtual Machine) is an abstract machine that enables your computer to run a Java program.

When you run the Java program, Java compiler first compiles your Java code to bytecode. Then, the JVM translates bytecode into native machine code (set of instructions that a computer's CPU executes directly).

Java is a platform-independent language. It's because when you write Java code, it's ultimately written for JVM but not your physical machine (computer). Since JVM executes the Java bytecode which is platform-independent, Java is platform-independent.

➢JRE

JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.

➢JDK

JDK (Java Development Kit) is a software development kit required to develop applications in Java. When you download JDK, JRE is also downloaded with it.

In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger, etc).

//

16. Who are the founders of the Java programming language, and in which year was it first released?

➢ The Java programming language was developed by a team of engineers at Sun Microsystems, which was later acquired by Oracle Corporation. The key individuals involved in the creation of Java were:

1. **James Gosling**: Often referred to as the "father of Java," James Gosling played a central role in the design and development of the language.
2. **Mike Sheridan**: Mike Sheridan was one of the original members of the Java team and contributed to the early development of the language.
3. **Patrick Naughton**: Patrick Naughton was another early member of the Java team and contributed to the initial design and development efforts.

Java was first released to the public in **1995**, and it quickly gained popularity due to its "Write Once, Run Anywhere" philosophy, which allows Java programs to be executed on any platform with a compatible Java Virtual Machine (JVM). Java's portability and versatility have made it one of the most widely used and enduring programming languages in the software development industry.

//

17. Who is known as the father of the Java Programming language?

➢ James Gosling is often referred to as the "father of the Java programming language." He played a central and influential role in the design and development of Java during his time at Sun Microsystems. James Gosling's contributions were instrumental in the creation of Java, and he is widely recognized for his work in bringing the language to fruition.

//

18. What motivated the creators of Java to develop the Java programming language, and what were
some of their primary goals and objectives?

The creators of the Java programming language, led by James Gosling and his team at Sun Microsystems, had several motivations, goals, and objectives when developing Java:

1. **Platform Independence**: One of the primary motivations behind Java was to create a programming language that would be platform-independent. They wanted to enable developers to write code once and run it on any platform that had a compatible Java Virtual Machine (JVM). This "Write Once, Run Anywhere" (WORA) capability was a revolutionary concept at the time.
2. **Embedded Systems**: Sun Microsystems was interested in developing software for consumer electronics and other embedded systems. Java was designed with a focus on being compact and efficient, making it suitable for these resource-constrained environments.
3. **Internet and Network Computing**: Java was born during the early days of the internet, and the creators saw the need for a language that could enable the development of interactive and networked applications. Java was designed with built-in support for network programming, making it well-suited for web and client-server applications. 4. **Safety and Security**: The creators of Java were also concerned about the security and safety of software. Java introduced features like a strong type system, automatic memory management (garbage collection), and a security model that could be used to run untrusted code in a sandboxed environment.
5. **Ease of Use**: Java was designed to be user-friendly and relatively easy to learn. It borrowed syntax and concepts from C and C++ but aimed to eliminate some of the complexities and potential pitfalls associated with those languages.
6. **Object-Oriented Programming**: Java embraced object-oriented programming (OOP) principles, making it a natural choice for developing modular and reusable code. 7.

Robustness and Reliability: The creators aimed to build a language that would produce reliable and robust software. Java's strong type checking, exception handling, and other features contribute to this goal.

8. **Community and Collaboration**: Java was developed with the intention of fostering a strong developer community. Sun Microsystems promoted the idea of open collaboration and encouraged third-party developers to extend Java through APIs and libraries.

Overall, Java's creators envisioned a versatile, safe, and portable programming language that could meet the demands of a rapidly evolving technology landscape. Their goals and motivations led to the development of Java as a widely adopted and influential language in the world of software development.

//

19. Which company currently maintains and oversees the development of the Java platform.

in September 2021, Oracle Corporation is the company that maintains and oversees the development of the Java platform. Oracle acquired Sun Microsystems, the original creator of Java, in 2010, which included ownership of the Java technology. Since then, Oracle has been responsible for the stewardship and development of the Java platform.

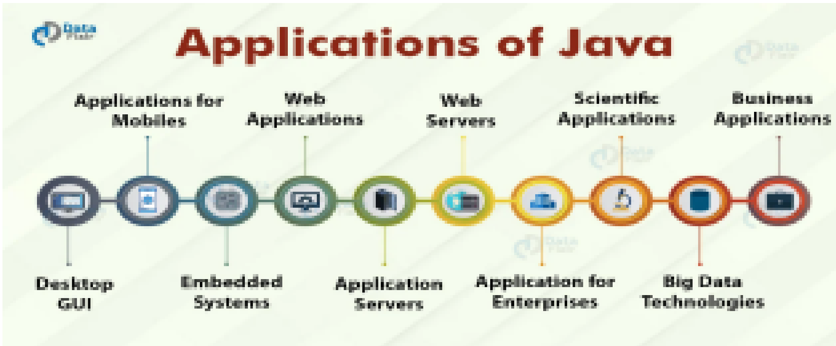
//

20. What are the well-known applications and industries where Java is widely used?

Applications of Java

Some of the many applications of Java are:

- Desktop GUI
- Applications for Mobiles
- Embedded Systems
- Web Applications
- Application Servers
- Web Servers
- Applications for Enterprises
- Scientific Applications
- Big Data Technologies
- Business Applications



//

21. What is a program in the context of computer science and software development?

In the context of computer science and software development, a "program" refers to a set of instructions or code that is written to perform a specific task or a series of tasks on a computer. These instructions are typically written in a programming language and are processed by a computer's central processing unit (CPU) to carry out the desired operations.

//

22. What is a programming language, and why is it needed in computer programming?

A programming language is a formalised and structured means of communicating instructions to a computer. It is a set of rules, symbols, and conventions that programmers use to write code that can be executed by a computer. Programming languages serve as intermediaries between human programmers and the computer's hardware, allowing developers to create software and automate various tasks.

Here are the key reasons why programming languages are needed in computer programming:

1. **Human-Readable Communication**: Programming languages provide a way for programmers to express their ideas, algorithms, and logic in a human-readable and understandable form. This abstraction makes it easier for developers to write code and collaborate with others.
2. **Machine Execution**: Computers operate at a very low level of abstraction, dealing with binary instructions. Programming languages bridge the gap between human understanding and machine execution by translating high-level code into machine code, which the computer's CPU can execute.
3. **Abstraction and Modularity**: Programming languages offer abstraction mechanisms that allow developers to hide complex details and focus on solving specific problems. Modularity, achieved through functions, classes, and modules, helps in breaking down complex programs into manageable parts.

4. **Reusability**: Programming languages promote code reuse. Developers can write functions, libraries, and modules that can be used in multiple projects, saving time and effort.
5. **Portability**: Many programming languages are designed to be portable, meaning that code written in a particular language can be executed on different platforms without modification. This portability is especially important for cross-platform software development.

//

23. What is the role of the Java Community Process (JCP) in the evolution of Java?

The Java Community Process (JCP) plays a crucial role in the evolution and development of the Java platform. It is a formal mechanism established by Oracle (previously Sun Microsystems) to ensure that Java remains an open, community-driven, and consensus-based platform. The JCP facilitates collaboration among various stakeholders, including individuals, organisations, and corporations, to evolve the Java platform in a structured and inclusive manner. Here are the key roles and functions of the Java Community Process:

1. **Specification Development**: The JCP is responsible for defining and updating the specifications for the Java platform, including the Java Standard Edition (Java SE), Java Enterprise Edition (Java EE), and Java Micro Edition (Java ME). These specifications outline the features, APIs, and behaviours of Java components and libraries.
2. **Creating Java Standards**: Through the JCP, standards for Java are established and maintained. These standards ensure that Java-based applications and libraries are interoperable across different implementations and platforms.
3. **JSR (Java Specification Request) Process**: The JCP uses the JSR process to propose, review, and approve changes to Java specifications. A JSR is a formal proposal for a new feature, improvement, or change in the Java platform. It includes detailed documentation, technical specifications, and an expert group responsible for reviewing and developing the proposed changes.
4. **Expert Groups**: Each JSR forms an expert group consisting of experts from the Java community, including developers, engineers, and representatives from various organizations. These expert groups collaborate to refine and finalize the proposed specifications and APIs.
5. **Transparency and Openness**: The JCP operates in an open and transparent manner, allowing anyone to participate, review, and provide feedback on proposed changes. This
openness ensures that Java's development is not controlled solely by a single entity and benefits from a diverse range of perspectives.

//

24. What is the Java Language Specification (often abbreviated as JLS) and who maintains it?

The Java Language Specification (JLS) is a comprehensive document that defines the syntax, semantics, and behavior of the Java programming language. It serves as the official reference for Java developers, compiler writers, and those involved in the implementation of the Java language. The JLS provides precise and unambiguous rules that dictate how Java programs should be written and how they should behave when executed.

The Java Language Specification covers various aspects of the language, including:

1. **Syntax**: The rules governing the structure of Java programs, such as the format of declarations, statements, and expressions.
2. **Semantics**: The meaning and interpretation of Java code, including variable scoping, type checking, and method invocation.
3. **Class and Interface Declarations**: Guidelines for defining classes, interfaces, constructors, fields, and methods.
4. **Exception Handling**: How exceptions are thrown, caught, and propagated. 5. **Concurrent Programming**: Rules and recommendations for multi-threading and synchronization.
6. **Type System**: The Java type system, including primitive types, reference types, and type conversions.
7. **Virtual Machine Behavior**: How Java programs are executed on the Java Virtual Machine (JVM).
8. **Java Libraries**: The JLS often references the Java Standard Library classes and their behavior.

The Java Language Specification is an essential resource for developers to ensure that their code adheres to the Java language's standards. It also provides guidance to compiler and tool developers, helping them implement the Java language correctly.

The JLS is maintained and updated through the Java Community Process (JCP). Expert groups within the JCP are responsible for proposing, reviewing, and finalizing changes to the specification. These expert groups typically consist of language designers, compiler experts, and representatives from organizations involved in Java development. Once a new version of the JLS is approved through the JCP process, it becomes the authoritative reference for the Java language until the next version is developed and ratified.