1. Fill in the blanks in each of the following sentences about the Java environment:

a) The __java__ command from the JDK executes a Java application.
b) The __javac__ command from the JDK compiles a Java program.
c) A Java program file must end with the __.java__ file extension.
d) When a Java program is compiled, the file produced by the compiler ends with the __.class__ file extension.
e) The file produced by the Java compiler contains __bytecode__ that are executed by the Java Virtual Machine.

2. What is the task of the compiler in programming languages?

   The task of a compiler in programming languages is to translate the source code written in a high-level programming language into machine code or bytecode that can be executed by the computer's processor.

3. What is the difference between C and Java languages?

   C is a low-level, procedural language that requires manual memory management and is platform-dependent. Java, on the other hand, is a high-level, object-oriented language with automatic memory management, platform independence, and a rich set of libraries and frameworks.

4. If we compile a Java program in the Windows environment, can we run that program in any other operating system? Explain your answer briefly.

   Yes, we can run a Java program compiled in Windows environment on any other operating system, such as Linux, macOS, or any other platform that supports the Java Virtual Machine (JVM). Java programs are compiled into bytecode, which is a platform-independent format that can be executed by any device with a JVM installed. This makes Java a platform-independent language, meaning that code written in Java can be run on any platform that has a JVM, regardless of the underlying operating system or hardware.

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

   The Java interpreter is part of the Java Virtual Machine (JVM) and is responsible for executing Java bytecode. It translates bytecode into machine code that can be executed by the computer's processor. The Java interpreter also provides many other features, such as automatic memory management and exception handling.
   The main difference between the two is that the Java interpreter executes Java bytecode, whereas the O/S interpreter executes commands entered by the user in the command line

interface. Additionally, the Java interpreter is part of the JVM and runs inside a Java runtime environment, while the O/S interpreter runs directly on the operating system.

6. What is the meaning of “Compile” in Java language? And why is that necessary?

   compiling is the process of translating the source code written in the Java programming language into bytecode that can be executed by the Java Virtual Machine

7. What are the steps needed to create and run a Java program in a Linux environment?

   to create and run a Java program in a Linux environment, you need to install the JDK, create a directory for your project, write your Java code, compile it using the javac command, and then run it using the java command.

8. Akila creates a Java program in his HP laptop machine. He gave the compiled code (Class File) to Manoj to run that program in his computer. But Manoj’s laptop brand is Dell. When Manoj tries to run the program, it didn’t work. What can be the reason for that?

   The reason why the Java program did not work on Manoj's Dell laptop could be due to differences in the underlying hardware and operating system architecture of the two machines.

9. Explain the command “java Example” that we write in Terminal.

   The command java Example that we write in the terminal is used to run a Java program that has been compiled into bytecode.

   Assuming that we have a Java program named "Example" and we have compiled it into bytecode using the javac command, running java Example in the terminal will execute the compiled bytecode and run the program.

10. 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[]){} - invalid
    d. public static void main(String args){ } - invalid
    e. void main(String args[]){ } - invalid
    f. public static void main(){ } - invalid
    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 []){ } - inavlid

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