**Q #1) What is JAVA?**

**Ans:** Java is a high-level programming language and is platform independent.

Java is a collection of objects. It was developed by Sun Microsystems. There are a lot of applications, websites and Games that are developed using Java.

**Q #2) What are the features in JAVA?**

**Ans: Features of Java:**

* **Oops concepts**
  + Object-oriented
  + Inheritance
  + Encapsulation
  + Polymorphism
  + Abstraction
* **Platform independent:** A single program works on different platforms without any modification.
* **High Performance:** JIT (Just In Time compiler) enables high performance in Java. JIT converts the bytecode into machine language and then JVM starts the execution.
* **Multi-threaded:** A flow of execution is known as a Thread. JVM creates a thread which is called main thread. The user can create multiple threads by extending the thread class or by implementing Runnable interface.

**Q #3) How does Java enable high performance?**

**Ans:** Java uses Just In Time compiler to enable high performance. JIT is used to convert the instructions into bytecodes.

**Q #4) What are the Java IDE’s?**

**Ans:** Eclipse and NetBeans are the IDE's of JAVA.

**Q #7) What is a Class?**

**Ans:** All Java codes are defined in a class. A Class has variables and methods.

**Variables**are attributes which define the state of a class.

**Methods** are the place where the exact business logic has to be done. It contains a set of statements (or) instructions to satisfy the particular requirement.

**Q #10) What is Inheritance?**

**Ans:** Inheritance means one class can **extend**to another class. So that the codes can be reused from one class to another class.

Existing class is known as Super class whereas the derived class is known as a sub class.

**Q3. Why Java is platform independent?**

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

### ****Q5. What are wrapper classes in Java?****

Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays different primitive type, wrapper class and constructor argument.

Example: Integer, Byte, Float etc.

**Q15. Define a Java Class.**

A class in Java is a blueprint which includes all your data.  A class contains fields (variables) and methods to describe the behavior of an object. Let’s have a look at the syntax of a class.

class Abc {

member variables // class body

methods

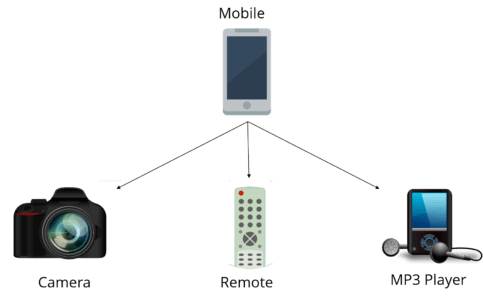
}

**Q29. What is a classloader in Java?**

The **Java ClassLoader** is a subset of JVM (Java Virtual Machine) that is responsible for loading the class files. Whenever a Java program is executed it is first loaded by the classloader. Java provides three built-in classloaders:

1. Bootstrap ClassLoader
2. Extension ClassLoader
3. System/Application ClassLoader

**Q1. What is Polymorphism?**

Polymorphism is briefly described as “one interface, many implementations”. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are two types of polymorphism:

1. Compile time polymorphism
2. Run time polymorphism

Compile time polymorphism is method overloading whereas Runtime time polymorphism is done using inheritance and interface.

**Q3. What is abstraction in Java?**

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. Thus you can say that abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them. Abstraction can be achieved in two ways:

1. **Abstract Classes** (0-100% of abstraction can be achieved)
2. **Interfaces** (100% of abstraction can be achieved)

### What do you understand by Java virtual machine?

[Java Virtual Machine](https://www.javatpoint.com/jvm-java-virtual-machine) is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.

### 7) What is JIT compiler?

**Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

### 8) What is the platform?

A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.

9) What are the main differences between the Java platform and other platforms?

There are the following differences between the Java platform and other platforms.

* Java is the software-based platform whereas other platforms may be the hardware platforms or software-based platforms.
* Java is executed on the top of other hardware platforms whereas other platforms can only have the hardware components.

10) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

21) What is the output of the following Java program?

|  |  |
| --- | --- |
| **class** Test  {  **public** **static** **void** main (String args[])      {          System.out.println(10 \* 20 + "Javatpoint");           System.out.println("Javatpoint" + 10 \* 20);      }  } | The output of the code will be  200Javatpoint  Javatpoint200  **Explanation**  In the first case, The numbers 10 and 20 will be multiplied first and then the result 200 is treated as the string and concatenated with the string **Javatpoint** to produce the output **200Javatpoint**.  In the second case, The numbers 10 and 20 will be multiplied first to be 200 because the precedence of the multiplication is higher than addition. The result 200 will be treated as the string and concatenated with the string **Javatpoint**to produce the output as **Javatpoint200**. |

What is object-oriented paradigm?

It is a programming paradigm based on objects having data and methods defined in the class to which it belongs. Object-oriented paradigm aims to incorporate the advantages of modularity and reusability. Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.

* Follows the bottom-up approach in program design.
* Focus on data with methods to operate upon the object's data
* Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality.
* Implements the real-time approach like inheritance, abstraction, etc.
* The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.

## Compiled Versus Interpreted Languages

Every program is a set of instructions, whether it’s to add two numbers or send a request over the internet. Compilers and interpreters take human-readable code and convert it to computer-readable machine code. In a compiled language, the target machine directly translates the program. In an interpreted language, the source code is not directly translated by the target machine. Instead, a different program, aka the interpreter, reads and executes the code.

### Okay… but what does that actually mean?

So let’s say you have a hummus recipe that you want to make, but it’s in Ancient Greek. There are two ways you, as a non-Ancient-Greek speaker, could follow its directions.

The first is if someone had translated it into English for you already. You (and anyone else who could speak English) could get the English version and make hummus. This is the compiled version.

The second is if you had a friend who knows Ancient Greek. Your friend can sit next to you and translate the Ancient Greek into English, line by line, as you go. In this case, your friend is the interpreter. This is the interpreted version.