JAVA TECHNICAL INTERVIEW

Kevin chiloane

QUICK QUESTIONS

Q1. Explain JVM, JRE, and JDK.

JVM (Java Virtual Machine): JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that calls the main method present in Java code. JVM is a part of JRE(Java Runtime Environment).
JRE (Java Runtime Environment): JRE refers to a runtime environment in which Java bytecode can be executed. It implements the JVM (Java Virtual Machine) and provides all the class libraries and other support files that JVM uses at runtime. So JRE is a software package that contains what is required to run a Java program. It's an implementation of the JVM which physically exists.

- JDK(Java Development Kit): It is the tool necessary to compile, document, and package Java programs. The JDK completely includes JRE which contains tools for Java programmers. The Java Development Kit is provided free of charge. Along with JRE, it includes an interpreter/loader, a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed in Java development. In short, it contains JRE + development tools.
- Q2. Explain public static void main(String args[]).
- Public: Public is an access modifier. Public means that this Method will be accessible by any Class.
  static: It is a keyword in java that identifies it as class-based i.e it can be accessed without creating the instance of a Class. Since we want the main method to be executed without any instance also, we use static.

- Void: It is the return type of the method. Void defines the method which will not return any value.
  main: This is the first method executed by JVM. The signature of the method must be the same.
- Q3. Why Java is platform independent?
- Platform independence practically means “write once run anywhere”. Java is called so because of its byte codes which can run on any system irrespective of its underlying operating system.
- **Q4. Why is Java not purely Object-oriented?**
- Java is not considered pure Object-oriented because it supports primitive data types such as boolean, byte, char, int, float, double, long, and short.

- Q5. Define class and object. Explain them with an example using java.
- **Class:** A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:
  Superclass(if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
  Interfaces: A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
  Object: It is a basic unit of Object Oriented Programming and represents real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :
  State: It is represented by attributes of an object. It also reflects the properties of an object.

- **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
  Identity: It gives a unique name to an object and enables one object to interact with other objects.
- **Q6. What is a method? Provide several signatures of the methods.**
- A Java method is a set of statements to perform a task. A method is placed in a class.
- Signatures of methods: The method's name, return type, and the number of parameters comprises the method signature.
- A method can have the following elements in its signature:
  – Access specifier – public, private, protected, etc. (Not mandatory)
  – Access modifier – static, synchronized, etc. (Not mandatory)
  – Return type – void, int, String, etc. (Mandatory)
  – Method name – show() (Mandatory)
  – With or without parameters – (int number, String name); (parenthesis are mandatory)

- Q7. Explain the difference between an instance variable and a class variable.
- An instance variable is a variable that has one copy per object/instance. That means every object will have one copy of it.
  A class variable is a variable that has one copy per class. The class variables will not have a copy in the object.
- Q8. Which class is the superclass of all classes?
  java.lang.Object is the root class for all the java classes and we don't need to extend it.

- **Q9. What are constructors in Java?**
- In Java, a constructor refers to a block of code that is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and is automatically called when an object is created.
  If a class does not explicitly declare any, the Java compiler automatically provides a no-argument constructor, also called the default constructor.
  This default constructor calls the class parent's no-argument constructor (as it contains only one statement i.e. super();), or the Object class constructor if the class has no other parent (as the Object class is a parent of all classes either directly or indirectly).
- There are two types of constructors:

1. Default constructor
2. Parameterized constructor

- **Q11. What's the purpose of Static methods and static variables?**
- When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.
  Static variable: Static variables are also known as Class variables.
  These variables are declared similarly to instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
  Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.
  Static variables are created at the start of program execution and destroyed automatically when execution ends.
  To access static variables, we need not create an object of that class.

- **Static methods:** A static method can be accessed without creating objects. Just by using the Class name, the method can be accessed. The static method can only access static variables, not local or global non-static variables.
- **Q12. Why can static methods not access non-static variables or methods?**
- A static method cannot access non-static variables or methods because static methods can be accessed without instantiating the class, so if the class is not instantiated the variables are not initialized and thus cannot be accessed from a static method.
- **Q13. What is a static class?**
- A class can be said to be a static class if all the variables and methods of the class are static and the constructor is private. Making the constructor private will prevent the class to be instantiated. So the only possibility to access is using the Class name only.

- **Q17. Differences between HashMap and HashTable in Java.**
- 1. HashMap is non-synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.
  2. HashMap allows one null key and multiple null values whereas Hashtable doesn't allow any null key or value.
  3. HashMap is generally preferred over HashTable if thread synchronization is not needed.
- **Q18. What happens if you remove the static modifier from the main method?**
- Program compiles successfully. But at runtime throws an error "NoSuchMethodError".

- **Q19. Can we overload the main() method?**
- The main method in Java is no extra-terrestrial method. Apart from the fact that main() is just like any other method & can be overloaded similarly, JVM always looks for the method signature to launch the program.
- The normal main method acts as an entry point for the JVM to start the execution of the program.
- We can overload the main method in Java. But the program doesn’t execute the overloaded main method when we run your program, we need to call the overloaded main method from the actual main method only.

- **Q20. What are wrapper classes in Java?**
- Wrapper classes convert the Java primitives into 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.
- **Q22. What is the meaning of Collections in Java?**
- The collection is a framework that is designed to store the objects and manipulate the design to store the objects.
- Collections are used to perform the following operations:
- -> Searching
   -> Sorting
   -> Manipulation
   -> Insertion
   -> Deletion

   A group of objects is known as collection. All the classes and interfaces for collecting are available in the Java util package.

- Synchronization makes only one thread access a block of code at a time. If multiple threads access the block of code, then there is a chance for inaccurate results at the end. To avoid this issue, we can provide synchronization for the sensitive block of codes.
- The **synchronized** keyword means that a thread needs a key to access the synchronized code.
- Every Java object has a lock. A lock has only one key. A thread can access a synchronized method only if the thread can get the key to the objects to lock. Locks are per object.
- Refer to Synchronization for more details.
- **Q24. What is the disadvantage of Synchronization?**
- Synchronization is not recommended to implement all the methods. Because if one thread accesses the synchronized code then the next thread should have to wait. So it makes a slow performance on the other end.

- **Why is Java called the ‘Platform Independent Programming Language’?**
- Platform independence means that execution of your program does not dependent on type of operating system(it could be any : Linux, windows, Mac ..etc). So compile code only once and run it on any System (In C/C++, we need to compile the code for every machine on which we run it). Java is both compiler(javac) and interpreter(jvm) based language. Your java source code is first compiled into byte code using javac compiler. This byte code can be easily converted to equivalent machine code using JVM. JVM(Java Virtual Machine) is available in all operating systems we install. Hence, byte code generated by javac is universal and can be converted to machine code on any operating system, this is the reason why java is platform independent.

- **Explain Final keyword in java?**
- Final keyword in java is used to restrict usage of variable, class and method. Variable: Value of Final variable is constant, you can not change it. Method: you can't override a Final method. Class: you can't inherit from Final class. Refer this for details.
- **When is the super keyword used?**
- Super keyword is used to refer:
- immediate parent class constructor,
- immediate parent class variable,
- immediate parent class method.

- **What is the difference between StringBuffer and String?**
- String is an Immutable class, i.e. you can not modify its content once created. While StringBuffer is a mutable class, means you can change its content later. Whenever we alter content of String object, it creates a new string and refer to that,it does not modify the existing one. This is the reason that the performance with StringBuffer is better than with String.
- **Difference in Set and List interface?**
- Set and List both are child interface of Collection interface. There are following two main differences between them
- List can hold duplicate values but Set doesn’t allow this.
- In List interface data is present in the order you inserted but in the case of Set insertion order is not preserved.

- **Why multiple inheritance is not supported in java?**
- Java supports multiple inheritance but not through classes, it supports only through its interfaces. The reason for not supporting multiple inheritance is to avoid the conflict and complexity arises due to it and keep Java a Simple Object Oriented Language. If we recall this in C++, there is a special case of multiple inheritance (diamond problem) where you have a multiple inheritance with two classes which have methods in conflicts. So, Java developers decided to avoid such conflicts and didn't allow multiple inheritance through classes at all.
- **Can a top level class be private or protected?**
- Top level classes in java can't be private or protected, but inner classes in java can. The reason for not making a top level class as private is very obvious, because nobody can see a private class and thus they can not use it. Declaring a class as protected also doesn't make any sense. The only difference between default visibility and protected visibility is that we can use it in any package by inheriting it. Since in java there is no such concept of package inheritance, defining a class as protected is no different from default.

- **What is the difference between ‘throw’ and ‘throws’ in Java Exception Handling?**
- Following are the differences between two:
- throw keyword is used to throw Exception from any method or static block whereas throws is used to indicate that which Exception can possibly be thrown by this method
- If any method throws checked Exception, then caller can either handle this exception(using try catch block)or can re throw it by declaring another ‘throws’ clause in method declaration.
- throw clause can be used in any part of code where you feel a specific exception needs to be thrown to the calling method
- E.g. **throw** throw new Exception(“You have some exception”) throw new IOException(“Connection failed!!”) **throws** throws IOException, NullPointerException, ArithmeticException

- **What is finalize() method?**
- Unlike c++ , we don’t need to destroy objects explicitly in Java. ‘Garbage Collector‘ does that automatically for us. Garbage Collector checks if no references to an object exist, that object is assumed to be no longer required, and the memory occupied by the object can be freed. Sometimes an object can hold non-java resources such as file handle or database connection, then you want to make sure these resources are also released before object is destroyed. To perform such operation Java provide protected void finalize() in object class. You can override this method in your class and do the required tasks. Right before an object is freed, the java run time calls the finalize() method on that object.

- **What will happen if you put System.exit(0) on try or catch block? Will finally block execute?**
- By Calling System.exit(0) in try or catch block, we can skip the finally block. System.exit(int) method can throw a SecurityException. If System.exit(0) exits the JVM without throwing that exception then finally block will not execute. But, if System.exit(0) does throw security exception then finally block will be executed.
- **Q3. What happens if you remove static modifier from the main method?**
  Program compiles successfully. But at runtime throws an error “NoSuchMethodError”.

- **Q1. Can we Overload or Override static methods in java?**
- **Overriding :** Overriding is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in superclass (or base class) at runtime.
- **Overloading:** Overloading is related to compile time (or static) polymorphism. This feature allows different methods to have same name, but different signatures, especially number of input parameters and type of input parameters.
- **Can we overload static methods?** The answer is **'Yes'**. We can have two ore more static methods with same name, but differences in input parameters
- **Can we Override static methods in java?** We can declare static methods with same signature in subclass, but it is not considered overriding as there won't be any run-time polymorphism. Hence the answer is **'No'**. Static methods cannot be overridden because method overriding only occurs in the context of dynamic (i.e. runtime) lookup of methods. Static methods (by their name) are looked up statically (i.e. at compile-time).

- **Q4. What is the scope of variables in Java in following cases?**
- **Member Variables** (Class Level Scope) : The member variables must be declared inside class (outside any function). They can be directly accessed anywhere in class
- **Local Variables** (Method Level Scope) : Variables declared inside a method have method level scope and can’t be accessed outside the method.
- **Loop Variables** (Block Scope) : A variable declared inside pair of brackets “{” and “}” in a method has scope within the brackets only.

- **Q5. What is “this” keyword in java?**
  Within an instance method or a constructor, this is a reference to the current object — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.
  Usage of this keyword
- Used to refer current class instance variable.
- To invoke current class constructor.
- It can be passed as an argument in the method call.
- It can be passed as argument in the constructor call.
- Used to return the current class instance.
- Used to invoke current class method (implicitly)

- **Q6. What is an abstract class? How abstract classes are similar or different in Java from C++?** Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.
- Like C++, in Java, an instance of an abstract class cannot be created, we can have references of abstract class type though.
- Like C++, an abstract class can contain constructors in Java. And a constructor of abstract class is called when an instance of a inherited class is created
- In Java, we can have an abstract class without any abstract method. This allows us to create classes that cannot be instantiated, but can only be inherited.
- Abstract classes can also have final methods (methods that cannot be overridden). For example, the following program compiles and runs fine.

- **Q7. Which class is the superclass for every class?**
  Object class
- **Q8. <u>Can we overload main() method?</u>**
  The main method in Java is no extra-terrestrial method. Apart from the fact that main() is just like any other method & can be overloaded in a similar manner, JVM always looks for the method signature to launch the program.
- The normal main method acts as an entry point for the JVM to start the execution of program.
- We can overload the main method in Java. But the program doesn't execute the overloaded main method when we run your program, we need to call the overloaded main method from the actual main method only.

- **Q14. What is “super” keyword in java?**
  The super keyword in java is a reference variable that is used to refer parent class objects. The keyword “super” came into the picture with the concept of Inheritance. Whenever you create the instance of a subclass, an instance of parent class is created implicitly i.e. referred by super reference variable.
  Various scenarios of using java super Keyword:
- super is used to refer immediate parent instance variable
- super is used to call parent class method
- super() is used to call immediate parent constructor

- **Q15. What is static variable in Java?**
  The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than the instance of the class.
- The static can be:
- variable (also known as class variable)
- method (also known as class method)
- block
- nested class

- java ee is a platform with already built in tool that extends the capabilities java programming language and help build powerful applications ,one is example of its libralies is the servelet which acts to accept request from user in a WEB BASED PLATFORM ACTING A waite. i'm not practically experinced in it as it resource demanding and slow and I haven't been in a situation i felt the need to implement it

Java spring framework follows the definition of java ee but it a succesor of java ee to implement java appliaction by a faster rate and also being versatile by accomodating smaller applications

- some key functionality of it will be the DI(dependecy injection) allows develop to maintain dependecies in object rather that manual crerating objects and managing their dependecies mvc separates
- fifferent type of appication model (data) view(interfave) and controoller(handing request) provides abstraction in handing https requests making it easier to build applications

- rest api's are set of rules or insturction that computer programs use to comminate with each other, computer can use rest api to send or ask for datat from another in java they can build using
- boot
- kubernetes k8s are partially like docker containers they orhestratte the depolyment and automation, scaling and management of contanerized applications, kubernets contains a collection of nodes
- that act like parking spot for computers these nodes are called servesm, kubernetes allocates these nodes efficiently ensuring each computer program has has necessary resources to run, say a game of fortnite
- is being populated by gamers kubernes can manage the game as to provide more resources and dervers to accomodate the demand and it case of a crash kubernes can restart a computer program