By OnlineInterviewQuestions.com

Cognizant Interview Questions for Java Developers

Q1. What is a Singleton in Java?

Singleton class in Java contains only one object at one time. For the designing of the singleton class, the construction needs to be private. The static method needs to be written to return the object of the singleton class. In singleton class, methods like **getInstance()** are used.

When the getInstance method is called for the first time in the singleton class, an object of the class name single_isntance is created and is returned to the variable. A single class is initiated in the main class by 3 objects x,y and z by calling the method getInstance(). Singleton class is also implemented to offer global point access to the object.

Q2. Is Java supports multiple inheritance?

When one class extends for more than one class, then it is known as multiple inheritance.

No, Java does not support multiple inheritance to prevent the ambiguity that it can cause. The diamond problem is one such problem that can exist in multiple inheritance. The problem persists when the methods exist in both the sub and superclasses. Multiple inheritance can cause different problems like constructor chaining and cast in Java.

Q3. What is JPA and Junit?

JPA or **Java Persistence API** is a mechanism through which JAVA can outlive the application processes that have created them. JPA allows you to define which objects are persisted and how they should be persisted in the Java applications. At first, JPA was introduced as an EJB 3.0 subset. Every EJB container contains a persistent layer that is defined by the JPA.

Junit can be defined as the open-source framework for unit testing in Java. It is an example of xUnit architecture. Junit identifies the bugs in the code much early that makes them more trustworthy. It is ideal for developers working in a test-driven environment.

Q4. What is difference between set and list?

List can be defined as the type of ordered collection that regulates the elements in the insertion order. It also

allows duplicate elements. Inside the List interface, new methods are introduced. Listiterator can insert a List in both forward and backward directions. The list is implemented to store objects that are non-unique according to the instruction order.

Set can be defined as a type of unordered collection in which elements do not maintain any order. All the elements inside a Set are unique. The set interface does not introduce any new interface. Only collection interface methods can be implemented in the Set subclasses. The set can be inserted only in the forward direction with the assistance of an iterator.

Q5. Explain difference between put and post?

PUT inserts a resource or a file at the specified URI. If there is already a resource or file at the specific URI, it is then replaced by PUT. PUT is idempotent, but the responses of PUT are non-cacheable. The URI inside a PUT finds out the entity that is enclosed within a request.

A specific set of data is sent to the URI by **POST**. The resource present at that URI handles the request. The web server next determines what to do with the data in the specified resource. The POST is non-idempotent. The responses of the POST are cacheable as the server determines the ideal Expire Headers and Cache-control.

Q6. List various design patterns available in Java?

Design patterns/models are a well-proved answer for determining a particular problem or job. Design patterns/models are classified into two sections:

1. JEE Design Patterns.
2. Core Java (or JSE) Design Patterns.

In core java, there happens to be essentially three sorts of design models, which are additionally separated into their sub-sections:

- **Structural Design Pattern:** Composite Pattern, Decorator Pattern, Adapter Pattern, Bridge Pattern, Proxy Pattern, Facade Pattern, Flyweight Pattern.
- **Creational Design Pattern**: Singleton Pattern, Prototype Pattern, Factory Pattern, Abstract Factory Pattern, Builder Pattern.
- **Behavioural Design Pattern:** Interpreter Pattern, Iterator Pattern, Chain Of Responsibility Pattern, Observer Pattern, State Pattern, Command Pattern, Mediator Pattern, Memento Pattern, Strategy Pattern, Visitor Pattern, Template Pattern

Q7. How java removes unused objects from Memory?

The **Java runtime environment** has a trash collector that regularly clears the Memory managed by items that are not largely referenced. The trash compiler does its work automatically, although, in many situations, you can

need to explicitly demand trash collection by requesting the GC process in the System type.

You may eliminate an item in Java by excluding the reference to it by allowing null. Following that, it will incline to be automatically removed by the Garbage Collector. You introduced it to void. However, Java does not provide you the opportunity to deallocate retention.

Q8. Why Strings are immutable in Java?

The **strings are Immutable** in Java as the String items are stored in the **String pool**. Since stored String literals appear to be shared among multiple users there is constantly a danger, where one user's action might affect all different users. For instance, if one user adjusts the estimation of String "Test" to "TEST", all different users will also recognize that as "TEST".

Since storing String items was necessary from administration reason this danger was circumvented by creating String class Changeless. At a similar time, String was declared final so that no one may yield invariant of String type like Immutability, hashcode calculation, Caching, etc by increasing and reversing functions.

Q9. What is the use of hashcode in Java?

A **hashcode** is a number of a Java Object. This is something that enables objects to be collected or retrieved immediately in a Hashtable. hashCode() is utilized for bucketing in Hash executions like HashMap, HashSet, HashTable, etc. The amount collected from hashCode() is utilized as the container amount for collecting components of the map or set. This container quantity is the position of the component within the map or set.

The goal of the hashCode() program is to present a numeric description of an item's contents to present an alternate device to loosely classify it. The hashCode() yields an entity that describes the in-house concept location of the item.

Q10. List some Immutable classes in Java?

An **immutable state** indicates that once an item is designed, we may not be able to alter its element. Seldom it is important to create an immutable set as per the terms. ImmutableList, as implied by the title, is a kind of List that is changeless. It indicates that the content of the Program is established or unswerving after the declaration, which means, these are read-only. If any effort made to compute, delete and refresh components in the List, UnsupportedOperationException is discharged.

For instance, Character, Double, Boolean, Byte, Float, Long, Short, String, Integer and all fundamental wrapper groups are immutable classes.

Q11. Why do we use streams in Java?

Included in Java 8, the **Stream API** is utilized to concoct collections of items. A stream is a series of items that support various processes which may be pipelined to create the desired effect.

The specialties of Java stream imply –

- A stream is a non-data structure rather it necessitates data from the Arrays, Collections, or I/O carriers.
- Streams don't alter the original information arrangement; they only present the outcome as per the pipelined plans.
- Each common operation is gradually completed and delivers a stream as a consequence, hence multiple intermediate signs of progress may be pipelined. Terminal processes mark the conclusion of the current and deliver the outcome.

Q12. Tell me some difference between String builder and String buffer?

Differences between String builder and String buffer in Java

- StringBuffer is synchronized that is, thread-safe. It indicates that two threads may not describe the practices of StringBuffer concurrently. Whereas StringBuilder is non-synchronized which means that it is not thread-safe. It indicates that two threads may describe the processes of StringBuilder concurrently.
- Again, StringBuffer is less effective than StringBuilder. Whereas StringBuilder is more effective than StringBuffer.
- StringBuffer had been the single option for String administration till Java 1.4 but it produces one problem that each of its common methods is synchronized. Thus, StringBuffer produces Thread safety but on production value.
- Mostly, we do not apply String in a multithreaded situation, so Java 1.5 launched a distinct type StringBuilder that is alike to StringBuffer except for synchronization and thread-safety.

Q13. What is AOP in Spring framework?

Spring AOP allows Aspect-Oriented Programming in spring administrations. In AOP, features allow the modularization of matters like transaction administration, logging or safety that cut beyond multiple sorts and items (frequently termed crosscutting matters).

Moreover, **Spring AOP** is proxy-based. Spring utilizes both JDK proxies (approved whenever the proxied spot performs at least one alliance) and CGLIB proxies (if the marked does not perform any alliance) to generate the proxy for a presented target node. Thus, aspect-oriented programming (AOP) as the title suggests utilizes features in programming. It may be described as the separating of a cipher into distinctive modules, also identified as modularisation, wherever the feature is the fundamental part of modularity.

Q14. What is encapsulation in oops?

Encapsulation is one of the most basic concepts in object-oriented programming (OOP). Encapsulation describes the concept of bundling methods and data that operate on that data within a unit like a Java class. It also manipulates the data and keeps it secure from both the outside and inside from misuse and interference.

Encapsulation of the data led to the vital concept of data hiding in OOP. In a nutshell, encapsulation is the wrapping of information and data under a sole unit. It is a process of binding together the functions and data that manipulates them. Encapsulation is implemented through access and class modifiers.

Q15. Explain Collection in Java?

A Collection is an assortment of distinct objects described as a particular unit. Java presents Collection Framework which describes several types and interfaces to describe a group of articles as a single system. The Collection alliance (java.util.Collection) and Map alliance (java.util.Map) are a couple of principal "root" assemblage of Java collection types. The necessity for Collection Structure:

Uses of Collection Structure:

- **Compatible API:** The API holds a basic collection of interfaces such as Collection, Map, Set, or List. All collections that perform these interfaces have a common assortment of styles.
- **Decreases programming work:** A developer doesn’t need to bother about the purpose of Stock, and he may concentrate on its most beneficial value in his presentation.
- **Progresses program rate and quality:** Boosts performance by implementing high-performance executions of helpful data constructions and algorithms.

Please Visit OnlineInterviewquestions.com to download more pdfs