Java Interview Questions for 5 years Experienced

In this section, we will see the common interview questions that has been asked to the 5 years experienced candidate.

Q1. Differentiate between Transient and Volatile Variable in Java.

Transient Variable	Volatile Variable
The keyword transient is used when one does not want the variable to be serialized.	The keyword volatile against the name of the variable shows that whatever content is there in the variable, is stored in the main memory. Thus, all the read of that variable must be done using the main memory and not from the CPU cache, and every write should also be done to the main memory.
The Transient keyword provides control and flexibility over the different object's attributes from getting serialized.	The volatile keyword guarantees that the Java Virtual Machine does not do the re-ordering of the variables and makes sure that the issues related to synchronization are avoided.
The variables decorated with the transient keyword are assigned default values as per their data types when deserialization occurs.	There are no default values assigned to a volatile variable.
Static keyword can not be used with the transient variable. It is because static variables are directly associated with the Class and not with the instances of the class, and this matters a lot during serialization.	One can use static keyword with the volatile keyword.

Q2. Observe the following code and answer the following.

The two threads: Th1 and Th3 are given to us. Th1 is accessing the method1() method. Will Th3 be able to access the method2() method at the same time and on the same instance?

```
1. Class ABC
2. {
3.     synchronized void method1()
4.     {
5.         System.out.println("In method - 1");
6.     }
7.     void method2()
8.     {
9.         System.out.println("In method - 2");
10.    }
11. }
```

Answer: Yes, Th3 will be able to access method2() as it is not decorated with the keyword synchronized and hence does not require a lock for accessing it.

Q3. Explain the significance of … in the following method's parameters.

```
1. public void fooMethod(String... infoArgs)
2. {
3. // body of the method
4. }
```

In Java 5, the 3 dots feature was introduced. The feature is also called varargs (which means variable arguments). It implies that the method can receive one or more than one String, as mentioned below:

fooMethod("Java", "T", "point");

fooMethod("Interview", "Java", "Questions");

fooMethod(new String[]{"Questions", "of", "Java", "Interview", "Questions"});

These received arguments can be used as an array and can be accessed by iterating

through loops as shown below:

```
1. public void someMethod(String... infoArgs)
2. {
3. for(String fewInfo : infoArgs)
4. {
5. // any operation
6. }
7. // The infoArgs can be acquired with the help of index-based loops too.
8. for( int i = 0; i < infoArgs.length; i++)
9. {
10. String s = info[i];
11. //some operation
12. }
13. }
```

Q4. Distinguish between the ArrayList and Vector in Java.

Vector and ArrayList are the classes of collection. Both the classes are derived from the AbstractList and implement the List interface.

ArrayList	Vector
ArrayList is neither synchronized nor thread-safe.	Vector is, by default, synchronized and thread-safe. It means th internal state of the Class is not affected even if the multip threads are operating simultaneously.
Since ArrayList is not synchronized, it works quickly as compared to Vector.	There is an overhead of synchronization that comes along wit Vector. Hence, it is slower than ArrayList.

Q5. Explain the significance of the equals() and hashCode() contract.

Consider a scenario of the object of HashMap. It is a known fact that the HashMap key utilizes the hashCode() and the equals() method for finding the index or finding a key's value. Therefore, it is required to implement these methods properly. If these methods are not implemented properly, then hashMap will not work properly. The HashMap will pick the wrong key for updating the values. Therefore, it is important to correctly implement the equals() and the hashCode() methods. It is done properly only when we follow the hashCode-equals contract in a proper way.

The contract hashCode-equals contract says:

If two objects are the same or equal, then the method hashCode() should always generate the same result for both objects.

To make sure the above contract, one has to override the method hashCode() whenever the method equals() is overridden.

Q6. What will be printed on the console if the following print statement is run?

System.out.println(1.0/0.0);

The values 1.0 or 0.0 are double values. The Double class has a specific set of rules, such as -0.0, NaN, Double.INFINITY, etc., which supports in arithmetic computations. The above print statement will print Infinity on the console without giving any arithmetic exceptions.

Q7. What is the difference between the path variables and the classpath variables?

In the operating system, the path variable is present and is utilized for spotting the system executables. The classpath variables are used for finding the .class files and are related to Java executables.

Q8. Obtain the result of the following code. Also, find the reason for that result.

```
1. public class SomeClass
2. {
3. // main method
4. public static void main(String[] argvs)
5. {
6. meth1(null);
7. }
8. public static void meth1(Object o)
9. {
10. System.out.println("Object method is getting Invoked.");
11. }
12. public static void meth1(String str)
13. {
14. System.out.println("String method is getting Invoked.");
15. }
16. }
```

The output of the above program is "*String method is getting Invoked.*".

It is a known fact that a *null* value can be assigned to any reference type object in Java. Therefore, the object o, and the String str, can both take the null value. However, the output is only related to the print statement of the method meth1(), which takes the String as a parameter. The reason behind it is the nature of the Java compiler. The Java compiler picks that method that has more specific parameters, which is String in our Case. Note that the Object class is a class that is the parent class of each and every class in Java.

Q9. Identify the more appropriate approach from the given two approaches to invoke the wait() method

1) Using the loop construct

2) Using the if construct

The first approach is the more appropriate one. It is because the wait() method should be invoked using the loop construct. It is because when any thread gets the required

resources to start execution again, it is advised to verify the condition before starting the execution.

The way of doing it is mentioned below.

```
1. synchronized (res)
2. {
3. while(wait condtn)
4. res.wait(); // resource lock is released and reacquire it after waiting
5. // Do the tasks
6. }
```

Q10. In a multi-threaded environment, can someone use a HashMap?

Yes, it is possible to use a HashMap in a multi-threaded environment. But whether the HashMap will work properly or not depends on the user that is using it. If the initialization of the HashMap is done by only one thread, and the rest of the other threads do only the reading from the HashMap, then the HashMap will work fine.

The problem comes when one of the threads does the updation work by deleting, updating, or adding the content of the map. Thus, resizing the HashMap that can lead to an infinite loop or deadlock. In such a case, one can use ConcurrentHashMap or HashTable.

Q11. Find the result of the following code. Also, find the reason for it.

```
1. public class JTPProblem
2. {
3. // main method
4. public static void main(String[] argvs)
5. {
6. // the two print statements
7. System.out.println(0.2 == 0.1*2);
```

```
8.  System.out.println(0.3 == 0.1*3);
9.
10. }
11. }
```

Output:

```
System.out.println(0.2 == 0.1*2); // gives a true value
System.out.println(0.3 == 0.1*3); // gives a false value
```

Reason: The above-given output is because of that mismatch that is occurring because of the error that occurs in rounding the floating-point numbers. Only the floating-point numbers that are power of 2, are represented precisely by the binary representation. Others are rounded off. Therefore, the first print statement gives a true value and the second one a false value.

Q12. Comment on the following statement.

It is mandatory to declare all the objects that are immutable as final.

The answer is: it is not mandatory to declare all the immutable objects as final. The functionality of accomplishing immutability can be done by declaring the class members as private and not defining a setter method to update or modify the values. In case of reference members, ensure that they are not leaked outside of the class. The reference members can be initialized with the help of a constructor that is parameterized. Please note that variables decorated with the keyword final only prevent from re-assigning to some other value. It does not guarantee that individual aspects of the object can not be altered.

Q13. Briefly discuss the Factory Design pattern.

One of the most commonly used patterns in the software industry is Factory Design Pattern. The Factory Design Pattern comes under the category of Creational Pattern. It is used to create different types of objects as per the requirements. The logic of the creation of objects is not open to the client. Implementation of this pattern is done using the

interface. The Factory Design Pattern allows the subclass to make decisions about which class to instantiate.

Q14. Distinguish between the String creation using the operator new() and the String literal.

When the object is created using the String literal, the String is created in the string pool, whereas using the new() operator, the String is created in the heap. Note that in a heap, String pool is the part of the perm area. A number of Strings (of the same value) created with the help of literals contains the same value and point to the same object. Hence, it stops object duplication.

Q15. Give the output of the following code with the explanation.

```
1. public class JTPProblem
2. {
3. // main method
4. public static void main(String[] argvs)
5. {
6. System.out.println(Math.min(0.0d, Double.MIN_VALUE));
7. }
8. }
```

Output:

```
0.0
```

Reason: At first glance, it seems that the value of Double.MIN_VALUE should be printed because the minimum value should be the smallest negative value. However, it is not the case with Double. In Java, the type Double has MAX_VALUE & MIN_VALUE, and both are positive numbers. Thus, Double.MIN_VALUE is larger than 0.0.

Q16. Write some important features of Java 8.

Some of the important features of Java 8 are mentioned below.

- JDBC and IO enhancements
- StringJoiner & Collectors class
- Time / Date API
- Introduction of Default methods inside the interfaces
- Functional Interfaces
- Optional Class
- Lambda Expressions
- Stream API
- forEach Method

Q17. In Singleton Pattern, what is the significance of double-level locking?

The significance of double-level locking is to make it thread-safe.

```
1. public static Singleton retrieveInstance()
2. {
3.     // checking whether the instance has been
4.     // already created or not
5.     if (instnce == null)
6.     {
7.         synchronized (Singleton.class)
8.         {
9.             if (instnce == null)// Double level check
10.
11.            {
12.                instnce = new Singleton();
13.            }
14.        }
15.    } return instnce;
```

```
16. }
```

Suppose two threads(Th1 and Th3) validated the first if condition (*if(instance == null)*) and then reached the synchronized block (synchronized (Singleton.class)) for null, and both reached synchronized (Singleton.class). Th1 gets access to the lock and creates an instance of the Singleton, and returns. Now Th3 gets the entry in the synchronized block, and in the synchronized block again, we check the if condition, and it is found that *instance*is not null. Hence the instantiation will not happen again.

Q18. Discuss Dependency Injection and its significance in Object Oriented Programming.

When a software component depends upon other resources to complete its intended purpose, it needs to know which resources it should communicate with, where to locate them, and how to communicate with them.

Whenever one of the components of software is dependent on some other resources, to accomplish its work, that component must know how to communicate to the appropriate resources.

One approach is to structure the code in such a way that each resource that is required by the component is properly mapped. Another approach is to use dependency injections and let the external code take the responsibility of locating the resources. Usually, that external code is implemented using a framework, such as Spring Framework.

Dependency injections are helpful in generating the loosely coupled structure while following the SOLID principles. The reusability of code is enhanced using Dependency injection.

One way of structuring the code is to map the location of each required resource. Another way is to use dependency injections and have an external piece of code assume the responsibility of locating the resources. Typically, the external piece of code is implemented by using a framework, such as the Spring Framework for Java applications.

Dependency injections are useful to create loosely coupled programs while also following the SOLID software design principles. It helps improve the reusability of code while also reducing the frequency of needing to change a class, a template of the methods, or

variables in an object. Since programs are loosely coupled, it helps to develop code that is testable.

Q19. Predict the output of the following program. Also, give a reason for it.

```
1. public class JTPProblem
2. {
3. // main method
4. public static void main(String[] argvs)
5. {
6. Integer n1 = 2000, n2 = 2000;
7. System.out.println(n1 == n2); // the first print statement
8. Integer n3 = 10, n4 = 10;
9. System.out.println(n3 == n4); // the second print statement
10. }
11. }
```

Output:

```
false
true
```

Explanation: The first print statement prints false, even though the value is the same. It is because n1 and n2 point to different objects. Thus, references are different, and their comparison results in a false value. If we continue by this logic, the second print statement should also display the false value. However, it is a true value. It is because the Integer class has the inner class that is private and is known as IntegerCache class. The IntegerCache class does the caching of Integer objects that are within the range -128 to 127. Therefore, when we define Integer n3 = 10, it internally gets converted to

Integer n3 = Integer.valueOf(10). The valueOf() method of the Integer class is defined as:

```
1. public static Integer valueOf(int num)
2. {
3. if (num >= IntegerCache.low && num <= IntegerCache.high)
4. return IntegerCache.cache[num + (-IntegerCache.low)];
```

5. return new Integer(num);
6. }

By looking at the above definition of the method valueOf(), we see that when the value is within the range (-128 to 127), the first return statement is executed, and hence instance from the cache is returned. When the value of*num*is out of the range, the second return statement gets executed, resulting in the creation of a completely new instance. Thus, for references n3 and n4, an instance from cached is returned, leading the comparison to generate a true value.

Q20. Is it possible to overload the main method?

Yes, it is possible to overload the main method. Treat it like other methods. The way we overload the other methods, we overload the main method. JVM searches for the signature of the method for launching the program. The normal main method acts as the entry point of the program, and JVM first executes the normal main method first. Note that the program can not execute the overloaded main method unless we invoke that overloaded main method explicitly in the program.

Q21. Differentiate between an Enumeration and an Iterator.

The difference between an Enumeration and Iterator is that Enumeration does not have the remove() method, while Iterator has the remove() method. Therefore, manipulation of objects is possible using Iterator since one can manipulate the objects by removing or adding them from collections. The enumerations can only do object traversal and fetch them, it's behaviour is like the read-only interface. Also, Enumeration is a legacy interface, while Iterator is not a legacy interface. Enumeration is only applicable to legacy classes and is not a universal cursor. The Iterator is applicable to all of the collection classes and is a universal cursor.

Q22. Describe EnumSet.

The EnumSet implements the Set interface mainly for the enumeration-type objects. The EnumSet class extends the AbstractSet class and implements Set Interface. The EnumSet is not synchronized. The EnumSet class is part of the Collections Framework. The EnumSet

class is used for the creation of an enum set that contains all the elements in the specific element type.

Q23. Explain SerialVersionUID.

When serialization is done on an object, a version ID number is stamped on that object. It is known as SerialVersionUID. It is used during the process of deserialization to verify what one is going to deserialize is a proper object.

Q24. In the hashing-based collection, state the default size of the load factor.

When the load factor increases, the capacity increases in order to maintain the HashMap if the ratio of initial capacity to the current capacity exceeds the threshold. operational complexity as O(1). The operational complexity as O(1) means insertion and retrieval consume constant time. The default size of the load factor is 0.75.

Q25. Differentiate between fail-safe and fail-fast.

The differences are mentioned below.

Fail - Fast	Fail - Safe
When the object is modified during the process of iteration, ConcurrentModificationException is thrown.	An exception is not thrown.
For processing, Fail - Fast requires less memory	For processing, Fail - Safe require more memory
During the iteration process, a clone object is not created.	During the iteration process, a clon object or copy is created.
Fail - Fast is quick.	As compared to Fail - Fast, Fail - Safe a bit slower.

In the process of iteration of Fail - Fast, modification is not allowed.	In Fail - Safe, modification is allowed.
HashSet, HashMap, Vector, ArrayList, etc.	CopyOnWriteArrayList, ConcurrentHashMap, etc.

Q26. Discuss IdentityHashMap.

The IdentityHashMap, with the help of HashMap, implements the Map interface. It compares keys as well as values with the help of reference equality in lieu of object equality. The class does implement the Map interface but does break the general contract of Map intentionally, which has the demand that objects are getting compared with the help of the equals() method. The class is utilized whenever the user allows the comparison of the objects with the help of references. The class is present in the*java.util*package.

For more information about the class click here.

Q27. Mention the benefits and limitations of Spring AOP.

One of the biggest advantages of using Spring AOP is its configuration is quite easy. Also, the requirement of separate class loader or separate compilation unit is not required. It also has other benefits, like creating aspects with the help of @aspetJ or integration of cross-cutting concerns into the classes or XML annotations. For limitations, it is usually observed that code debugging is time-consuming in the AOP framework. Also, aspects can't advise the other aspects. It is due to the fact that when a class is marked as an aspect, the Spring framework stops it from auto-proxied.

Q28. What is the Properties class?

The subclass of the HashTable is the Properties class. The properties class keeps the list of values whose value is a string and whose key is also a string.

Characteristics of the Properties class:

- A subclass of the HashTable.

- The Properties file is used to keep and fetch string data type for a values list where the value and the key are also a string.
- If the main properties list is not containing a specified key property, then instead of it, the default properties list will be looked for.
- Without external synchronization, sharing of objects can occur among multiple Objects.
- Retrieval of system properties is done with the help of the Properties class.

Q29. Predict the output of the following code.

```
1. public class JTPProgram
2. {
3. // main method
4. public static void main(String argvs[])
5. {
6. short s = 8; // print statement 1
7. ss = s + 8; // print statement 2
8. // print statement
9. System.out.println(s);
10. }
11. }
```

Explanation:

At first glance, it seems that the program prints 16 on the console. However, it is not true. The program on compilation gives compilation errors. It is because of the print statement 2. In the statement *s = s + 8;* the data type of the number 8 is int, and we are assigning it to a variable short, which is not correct. It is because the capacity of int is larger than the capacity of the short data type. When we compile the above code, we get the error like the following.

Output:

```
/JTPProgram.java:7: error: incompatible types: possible lossy conversion from
int to short
s = s + 8;
```

```
      ^
1 error
```

Q30. Mention 5 best practices that is being used with the threads.

The 5 best practices are:

1) Mention the name of the thread

2) keep the thread and the task separate. Use Callable or Runnable with the thread pool executor.

3) usage of the thread pool

4) use the keyword volatile to indicate the compiler about the visibility, ordering, and the atomicity.

5) avoid using the local thread variable. It is because of the reason that the improper usage of the ThreadLocal class can lead to memory leaks.

Q31. When should one use the Flyweight pattern?

Flyweight pattern facilitates sharing object for supporting large numbers without actually creating too many objects. In order to use the Flyweight pattern, one needs to ensure that the objects are Immutable such that it can be shared safely. The pool of Long and Integer objects and String pool are examples of Flyweight patterns.

Q32. What is the difference between SAX and DOM parser?

SAX parser is an event parser and hence doesn't load the complete XML in the memory, whereas the DOM parser loads the complete XML in the memory to create the tree-based DOM model. That helps to locate the nodes and make the change in the structure of XML. It is because of this reason SAX is slower than the DOM. DOM requires more than SAX and is not good for parsing the XML files that are large