



Java Interview Questions & Answers - 2

Core Java

Introduction



What does this presentation Include?

These set of slides include basic Java Interview Questions and answers which can be very helpful for a Software Engineer to attend an interview. These questions are bit complex than that are included in the presentation '**Java Interview Questions & Answers - 1**' presentation. I hope this will be a useful resource for the beginners and the people who wish to attend technical interviews

1. What's the difference between Throwable, throw and throws key words in java?

Throwable : in Java, all error's and exception's class are derived from java.lang. Throwable class. It is the top of the hierarchy of classes of error and exceptions. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement.

No.	throw	throws
1)	Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
2)	Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
3)	Throw is followed by an instance.	Throws is followed by class.
4)	Throw is used within the method.	Throws is used with the method signature.
5)	You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException, SQLException.

2. Explain what an Abstract class is

- A class which contains the abstract keyword in its declaration is known as abstract class.
- Abstract classes may or may not contain abstract methods ie., methods without body (`public void get();`)
- But, if a class have at least one abstract method, then the class must be declared abstract.
- If a class is declared abstract it cannot be instantiated.
- To use an abstract class you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class you have to provide implementations to all the abstract methods in it.



3. Why do we call String as Immutable?

We call that String is Immutable since once we create a String , it cannot be changed.



4. State the differences between String, StringBuffer and StringBuilder

	String	StringBuffer	StringBuilder
Storage Area	Constant String Pool	Heap	Heap
Modifiable	No (immutable)	Yes(mutable)	Yes(mutable)
Thread Safe	Yes	Yes	No
Performance	Fast	Very slow	Fast



5. Print the output of the below code

```
public class StringExample{  
    public static void main(String[] args){
```

```
        String s1="Hello";  
        s1.concat(" World");
```

```
        System.out.println("String Object: "+s1);
```

```
        StringBuffer s2=new StringBuffer("Hello");  
        s2.append(" World");
```

```
        System.out.println("StringBuffer Object: "+s2);
```

```
    }
```

```
}
```

String Object: Hello

StringBuffer Object: Hello World

6. Write code to find the First non-repeated character in the String?

Approach followed:-

First scan through the String and add each of the letter and its no of appearance in the String in HashMap. Then traverse through the HashMap and find which key has the count as 1. Return the key (character) which is found first.


```
import java.util.HashMap;
import java.util.Scanner;

public class FirstNonRepeat{

    public static void main(String[] args){

        System.out.println("Enter the preferred String : ");
        Scanner in=new Scanner(System.in);
        String s=in.nextLine();
        char c=firstNonRepeatedCharacter(s);
        System.out.println("The first non repeated character is: " + c);

    }
```

```
public static Character firstNonRepeatedCharacter(String str){
    HashMap<Character,Integer> characterHashMap=new HashMap<Character,Integer>{
};
    int i,length;
    Character c;
    length=str.length();

    //scanning the string and building the HashMap
    for(i=0;i<length;i++){
        c=str.charAt(i);

        if(characterHashMap.containsKey(c)){
            characterHashMap.put(c,characterHashMap.get(c) + 1);
        }
        else{
            characterHashMap.put(c,1);
        }
    }

    //search the Hashtable till you find the first Character whose value is 1
    for(i=0; i<length; i++){
        c=str.charAt(i);

        if(characterHashMap.get(c) == 1){
            return c;
        }
    }
    return null;
}
```

7. State two examples when `ArrayIndexOutOfBoundsException` is caused in a java program.

When you have an `Array` or `ArrayList` and when you try to reach the indexes which are

- Less than 0
- Greater than the size of the `Array/ArrayList`



8. State the difference between an Array and an ArrayList

- Arrays have a fixed size (once they are defined, they cannot grow or shrink).
- ArrayLists are created with a size and when the size is exceeded the ArrayList can automatically grow. When items are removed from it then it will automatically shrink.



9. State the difference between ArrayList and Vector

ArrayList	Vector
Not Synchronized(two threads can access the ArrayList at the same time)	Synchronized (two threads cannot access the vector at the same time)
ArrayList is faster than Vectors since it's not synchronized	Vector is slower than ArrayList since it is thread safe. Vector is limited to one thread at a time
ArrayList does not define the increment size. There is no setSize() method available in ArrayList	Vector defines the increment size by dynamically by using the setSize(int i)
Introduced in java version 1.2	Exists from the very first version of java

10. How do you handle error condition while writing stored procedure or accessing stored procedure from java?

When you call a stored procedure the stored procedure will execute in database server so if there any exception occurs that can be handled in EXCEPTION block in the store procedure. If the stored procedure it self fails it throws sql exception which can be handled by try/catch block and wrap it to your project specific exception.



11. What will happen if you call return statement or System.exit on try or catch block? Will finally block execute?

In Java, finally block will execute even if you put return statement in try block or catch block but finally block won't run if you call System.exit from try or catch.



12. Can you override private or static method in Java ?

No, Static methods can't be overridden as it is part of a class rather than an object. But one can overload static method.



13. What will happen if we put a key object in a HashMap which is already there?

If you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.



14. Can you access non static variable in static context?

A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.



15. What is the Difference between JDK and JRE ?

The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed.


The Java Development Kit (JDK) is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like JavaDoc, and Java Debugger), in order for a user to develop, compile and execute Java applications.



16. Explain Autoboxing and Unboxing in java

Autoboxing is the automatic conversion made by the Java compiler between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this operation is called unboxing.





17. What is the difference between `String s = "Test"` and `String s = new String("Test")`? Which is better and why?

In general, `String s = "Test"` is more efficient to use than `String s = new String("Test")`

In the case of `String s = "Test"`, a `String` with the value "Test" will be created in the `String` pool. If another `String` with the same value is then created (e.g., `String s2 = "Test"`), it will reference this same object in the `String` pool.

However, if you use `String s = new String("Test")`, in addition to creating a `String` with the value "Test" in the `String` pool, that `String` object will then be passed to the constructor of the `String` Object (i.e., `new String("Test")`) and will create another `String` object (not in the `String` pool) with that value. Each such call will therefore create an additional `String` object (e.g., `String s2 = new String("Test")`) would create an additional `String` object, rather than just reusing the same `String` object from the `String` pool).

18. What is Function Overriding and Overloading in Java ?

Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.



19. What is a Constructor in Java?

A constructor gets invoked when a new object is created. Every class has a constructor. In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class.



20. Does Java support multiple inheritance?

No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces.



21. What is the difference between an Interface and an Abstract class?

Java provides and supports the creation of both abstract classes and interfaces. Both implementations share some common characteristics, but they differ in the following features:

- All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
- A class may implement a number of Interfaces, but can extend only one abstract class.
- In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
- Abstract classes can implement interfaces without even providing the implementation of interface methods.



- Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
- Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
- An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

