### Why is java not 100% object oriented?

Java supports primitive data types like int, float, char etc. which is not object oriented. So, java is not fully object oriented. Most languages are based on object oriented concepts but not purely object oriented. Only Smalltalk are recognised as purely object-oriented languages.

### Why are pointers not used in Java?

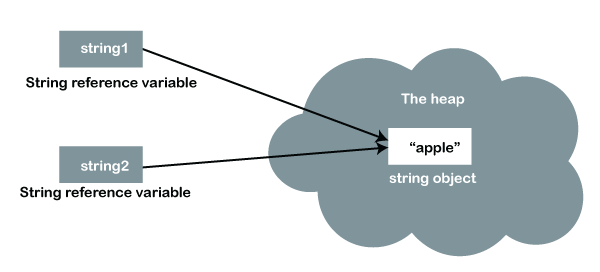
A pointer is a variable which can hold the address of another variable or object. But, Java does not support pointers due to security reasons, because if you get the address of any variable you could access it anywhere from the program without any restriction even if the variable is private.

### Why String JIT compiler in Java?

The String is immutable in Java because of the security, synchronization and concurrency, caching, and class loading.

The String objects are cached in the String pool, and it makes the String immutable. The cached String literals are accessed by multiple clients. So, there is always a risk, where action performs by one client affects all other clients.

Since Strings are immutable in Java, the JVM optimizes the amount of memory allocated for them by storing only one copy of each literal String in the pool. This process is called interning.



### What is the marker interface?

It is an empty interface (no fields or methods). Ex: Serializable, Cloneable, Remote interface.

* Cloneable interface: It is used to indicate that it is legal for clone(), it might give CloneNotSupportedException exception if you do not implement cloneable interface. By Convention, classes that implement this interface should override clone method.
* Serializable interface: It is used to make an object eligible for saving its state into a file.This is called Serialization. Classes that do not implement this interface will not have any of their state serialized or deserialized.
* Remote interface: A Remote object is an object which is stored at one machine and accessed from another machine. RMI(Remote Method Invocation) provides some convenience classes that remote object implementations can extend which facilitates remote object creation.

### Can you override a private or static method in java?

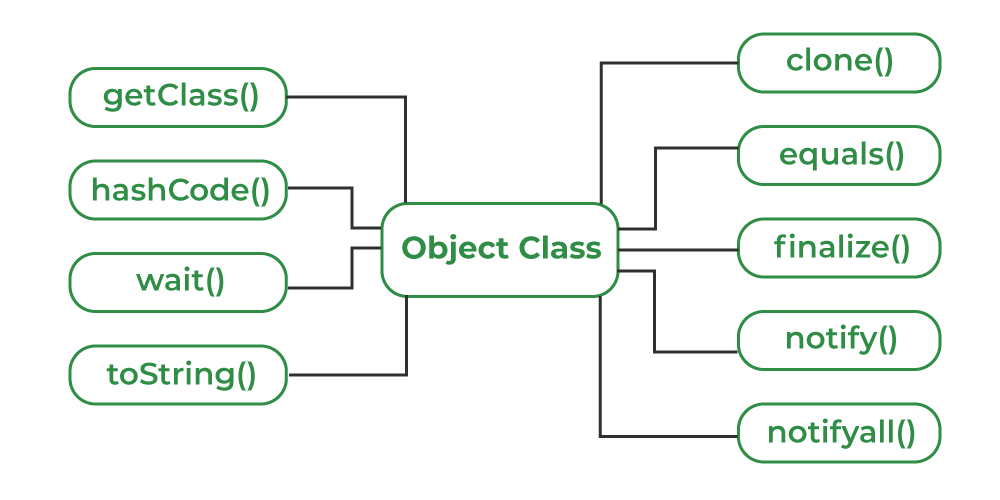
No, we cannot override static method because method overriding is based on dynamic binding at runtime and the static methods are bonded at compile time.

No, Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.

### Does finally always execute in java?

The finally block is always executed unless there is abnormal program termination, either resulting from a JVM crash or from a call to System.

### What methods does the object class have?

****

### How can you make a class immutable?

Immutable class in java means that once an object is created, we cannot change its content. In Java, all the [wrapper classes](https://www.geeksforgeeks.org/wrapper-classes-java/) (like Integer, Boolean, Byte, Short) and String class are immutable.

We can create our own immutable class by:

* Class must declare final.
* Data members in class must be private and final
* A Parameterized constructor should initialize all the fields performing Deep copy so data members can’t be modified by object reference.
* Deep copy of an object should be performed in the getter methods to return a copy rather than returning the object reference.
* There should be no setter method or any other way to change the value.

**Method Hiding:** if a sub class defines the static method with the same signature as parent class, then the method in the subclass hides the one in the superclass.This mechanism happens because the static method is resolved at the compiled time using type of reference not type of object.

**Deep Cloning:** Deep copy/ cloning is the process of creating exactly the independent duplicate objects in the heap memory and manually assigning the values of the second object where values are supposed to be copied.

For Java

- Java 8 features ?

- Stream API, Functional Interface ?

- Exception handling(checked & Unchecked) ?

- HashTreeSet (internal mapping) ?

- Overriding vs Overloading ?

- ArrayList vs Tree Hashset which is better for insertion(for sorting which is better) ?

- StringBuilder vs String ?

- What is static import?

- What is Static ?

- static vs final ?

For Spring Boot

- put vs patch ?

- spring boot annotations(Major) ?

- all mappings difference (requestmapping, getmapping, postmapping, putmapping) ?

- requestparam vs queryparam ?

- in JPA repository how to use multiple tables ?

- dependency injection ?

- basics of angular ?

## Java 8 features?

**Lambda expressions**: Lambda expression helps us to write our code in functional style. It provides a clear and concise way to implement SAM interface(Single Abstract Method) by using an expression. It is very useful in collection library in which it helps to iterate, filter and extract data.

**Method references**: Java 8 Method reference is used to refer method of functional interface . It is compact and easy form of lambda expression. Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

**Functional interfaces**: An Interface that contains only one abstract method is known as functional interface. It can have any number of default and static methods. It can also declare methods of object class.

Functional interfaces are also known as Single Abstract Method Interfaces (SAM Interfaces).

**Optional class**: Java introduced a new class Optional in Java 8. It is a public final class which is used to deal with NullPointerException in Java applications. It provides methods to check the presence of value for a particular variable.

**forEach() method**: Java provides a new method forEach() to iterate the elements. It is defined in Iterable and Stream interfaces.

**Nashorn JavaScript Engine**: Nashorn is a JavaScript engine. It is used to execute JavaScript code dynamically at JVM (Java Virtual Machine). Java provides a command-line tool **jjs** which is used to execute JavaScript code.

You can execute JavaScript code by two ways:

1. Using jjs command-line tool, and
2. By embedding into Java source code.

**Collectors**: Collectors is a final class that extends Object class. It provides reduction operations, such as accumulating elements into collections, summarizing elements according to various criteria etc.

Ex: > productsList.stream()

.map(x->x.price) // fetching price

.collect(Collectors.toList()); // collecting as list

> .collect(Collectors.toSet()); // collecting as list

> productsList.stream().collect(Collectors.summingInt(x->x.id));

> productsList.stream().collect(Collectors.averagingDouble(p->p.price));

> productsList.stream().collect(Collectors.counting());

**Stream API**: Java 8 java.util.stream package consists of classes, interfaces and an enum to allow functional-style operations on the elements. It performs lazy computation. So, it executes only when it is required.

* It simply conveys elements from a source through a pipeline of computational operations.
* Operation performed on stream does not modify it’s source.
* Stream is lazy.

**Base64 Encode and Decode**: Java provides a class Base64 to deal with encryption and decryption. This class provides three different encoders and decoders to encrypt information at each level.

**Parallel Array Sorting**:

**Stream API**:

Default methods,

Base64 Encode Decode,

Static methods in interface,

Optional class,

Collectors class,

ForEach() method,

Nashorn JavaScript Engine,

Parallel Array Sorting,

Type and Repating Annotations,

IO Enhancements,

Concurrency Enhancements,

JDBC Enhancements etc.