Q1. Features of java version 8

* Oracle released a new version of Java as Java 8 in March 18, 2014

1. Lamda expressions :- Lamda expression help us to write code in functional interface ,Lamda expressions are basically anonymous functions ,Lamda expression reduce line of codes, to call API’s very efffectively and it helps us to write readable and maintainable consise code
2. Method references: Method references are another way to write more concise and readable code in Java 8. They provide a way to refer to a method by name, rather than defining an anonymous function. Method references can be used in conjunction with lambda expressions to further simplify code and improve readability.
3. Optional

Java introduced a new class Optional in Java 8. It is a public final class which is used to deal with NullPointerException in Java application. We must import java.util package to use this class. It provides methods to check the presence of value for particular variable.

1. Date and Time API: Prior to Java 8, the date and time APIs in Java were considered to be cumbersome and difficult to use. The Date and Time API in Java 8 provides a much simpler and more intuitive way to work with dates and times.
2. for-each loop - Java provides a new method forEach() to iterate the elements. It is defined in Iterable and Stream interfaces.
3. Default Methods

Java provides a facility to create default methods inside the interface. Methods which are defined inside the interface and tagged with default keyword are known as default methods. These methods are non-abstract methods and can have method body.

1. Nashorn JavaScript engine: Java 8 introduced a new JavaScript engine called Nashorn,  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.

## Java Parallel Array Sorting -> Java provides a new additional feature in Arrays class which is used to sort array elements parallelly.

Q2. Features of java 15,17,19

Q3. Difference in c++ and Java

* Memory management: C++ requires manual memory management, which means that the programmer is responsible for allocating and deallocating memory for objects. This can be powerful, but it can also lead to errors such as memory leaks. In contrast, Java uses automatic memory management through garbage collection, which means that the JVM automatically frees up memory that is no longer being used.
* Platform independence: Java is designed to be platform-independent, meaning that Java code can run on any platform that has a JVM installed. This is because Java code is compiled to bytecode, which can be run on any platform that has a JVM. C++, on the other hand, is platform-dependent, which means that code written for one platform may not work on another platform without modification.
* Syntax: C++ has a more complex syntax than Java, with more low-level features such as pointers and operator overloading. Java has a simpler syntax and is generally easier to learn for beginners.
* Performance: C++ is generally faster than Java because it's a lower-level language that allows for more direct manipulation of hardware resources. However, Java's performance has improved significantly in recent years, and it's still a fast language.
* Object-oriented programming: Both C++ and Java are object-oriented programming (OOP) languages, but Java is more purely object-oriented, with all code being written in classes and objects. C++ allows for a mix of OOP and procedural programming.
* Standard libraries: Java has a larger and more comprehensive standard library than C++, with built-in support for networking, I/O, and other common tasks. C++ also has a standard library, but it's not as extensive as Java's.
* Exceptions handling: Java has a built-in exception handling mechanism, which makes it easier to write robust and error-free code. In C++, exception handling is optional and must be implemented manually, which can make code more error-prone.
* Pointer arithmetic: C++ allows pointer arithmetic, which can be a powerful feature but can also lead to errors such as buffer overflows and null pointer dereferences. Java does not allow pointer arithmetic, which makes it harder to make such errors.
* Multiple inheritance: C++ allows multiple inheritance, which means that a class can inherit from multiple classes. Java only allows single inheritance and interfaces, which limits the complexity of the class hierarchy but also makes the language easier to use.
* Operator overloading: C++ allows operator overloading, which means that operators such as +, -, \*, and / can be defined for user-defined types. Java does not allow operator overloading, which simplifies the language but also limits its expressiveness.

Q.