**Questions**

1. What are the main features of Java?

2. What is the difference between JDK, JRE, and JVM?

3. What are primitive data types in Java?

4. What is the difference between == and .equals()?

5. What is a constructor?

6. What is method overloading and method overriding?

7. What is the difference between ArrayList and LinkedList?

8. How is memory managed in Java (Stack vs Heap)?

9. What is the use of the final, finally, and finalize keywords?

10. Explain access modifiers in Java.

What are the four pillars of OOP?

Explain inheritance with an example.

What is polymorphism in Java?

What is abstraction? How is it implemented in Java?

What is encapsulation and why is it important?

What’s the difference between an interface and an abstract class?

Can a class extend multiple classes in Java? Why or why not?  
  
**Collections & Generics**

What is the Java Collections Framework?

Difference between HashMap, TreeMap, and LinkedHashMap.

How does HashSet ensure uniqueness?

What are generics and why are they useful?

What is the difference between List, Set, and Map?

What’s the difference between Comparable and Comparator?

**Concurrency & Multithreading**

What is the difference between Thread and Runnable?

What are synchronized methods and blocks?

Explain thread lifecycle in Java.

What is the use of volatile keyword?

How does wait(), notify(), and notifyAll() work?

What is a ThreadPoolExecutor?

**Exception Handling**

What is the difference between checked and unchecked exceptions?

What is a try-catch-finally block?

Can you catch multiple exceptions in one block?

How to create a custom exception in Java?

**Java 8+ Features (Lambdas, Streams, etc.)**

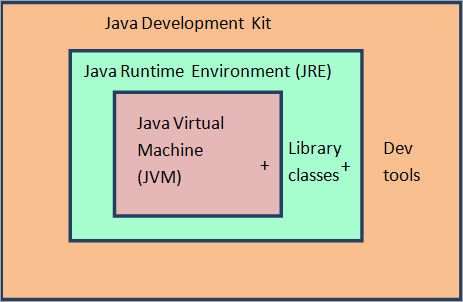
What is a lambda expression in Java?

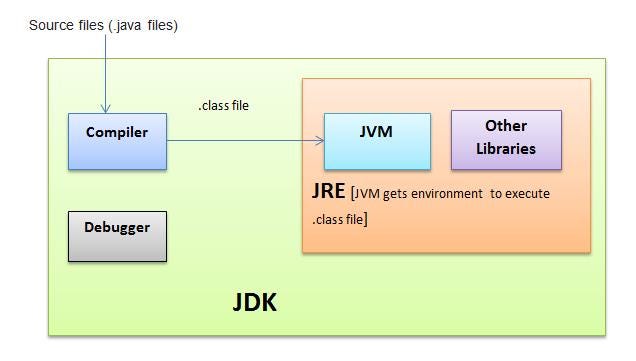
What are functional interfaces?

What is the Stream API and how is it used?

Difference between map() and flatMap() in streams.

What is Optional and how is it used to avoid NullPointerException?

**Java**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **How it Works** | |  |  | public | All classes |
| Hardware | OS | JVM |  | private | Declared class |
| my code | bytecode | jvm |  | default | Same Package |
| .java -> | java compiler -> | .class |  | protected | [Same Package and Subclass](https://www.w3schools.com/java/java_inheritance.asp) |

|  |  |  |
| --- | --- | --- |
| **Java Language** | | |
| Portable | Multithreaded | High performance |
| Object-oriented | Architecture-neutral | Distributed |
| Robust | Interpreted | Dynamic |

|  |  |
| --- | --- |
| **Static (Belongs to Type)** | |
| The static keyword is used for class-level members, meaning they belong to the class itself rather than instances (objects) Shared Memory, Access without instantiation, cant access non-static | |
| Blocks | Runs once do computation for variables |
| Variables | Single copy all classes |
| Methods | Access without creating class |
| Classes | For an inner nested class |

|  |  |  |
| --- | --- | --- |
| **OOP** | | |
| Inheritance | I do all the time | extends |
| Interface |  | implements |
| Abstract Class |  | extends |
| Method Overloading | Same name but different inputs |  |
| Polymorphism | Child can override parent method |  |
|  | Interface Animal - Dog Horse Sound |  |

|  |  |
| --- | --- |
| **Version** | **Features** |
| **Java 8** | Lambdas, Collections and Streams |
| **Java 9** | Collections new helper methods (Easily make set, list Map) |
|  | Interfaces got private methods (sensitive code) |
|  | HTTP Client |
| **Java 10** | var detect type |
| **Java 11** | HTTP Client- finalized |
|  | Strings got methods is blank and strip |
| **Java 13** | Switch statements can return a value and lambda syntax |
| **Java 14** | standarized switch |
| **Java 15** | Strings in multiline |
| **Java 16** | Patterm matching for instance of |
| **Java 17 (LTS)** |  |
| **Java 18** | UTF-8 by default |
|  | Simple Web Server (jwebserver) |
| **Java 19** | virtual threads |
| **Java 20** |  |