**Core Java**

**What is a constructor in Java?**

* A constructor is a block of code used to initialize an object.
* It has the same name as the class and does not have a return type.

**What is the difference between JDK, JRE, and JVM?**

* **JDK-** Java Development Kit. used for developing Java applications.
* **JRE-** Java Runtime Environment. Used to only run(not develop) the java application onto your machine.
* **JVM-** Java Virtual Machine. JVM runs Java bytecode.

**What are the main features of Java?**

* **Object-Oriented-** Java is an object-oriented programming (OOP) language, which means it uses objects to model real-world entities. Key principles include inheritance, encapsulation, polymorphism, and abstraction.
* **Platform Independent-** Java code is compiled into bytecode, which can be executed on any device with a Java Virtual Machine (JVM). This "write once, run anywhere" capability makes Java highly portable.
* **Simple-** Java is designed to be easy to learn and use. Its syntax is clean and concise, and it removes complex features like pointers and operator overloading found in other languages like C++.
* **Robust-** Java emphasizes early error checking, runtime checking, and memory management. Features like garbage collection, exception handling, and strong type checking contribute to Java's robustness.
* **Multithreaded-** Java supports multithreading, allowing multiple threads to run concurrently within a program. This makes it easy to develop applications that can perform many tasks simultaneously, such as games and web servers.
* **Portable-** Java's portability is due to its architecture-neutral nature and lack of implementation-dependent aspects. Java programs can move easily from one system to another.
* **High Performance-** Although Java is an interpreted language, it achieves high performance through the use of Just-In-Time (JIT) compilers, which convert bytecode into native machine code at runtime.

**What are the different types of memory areas allocated by JVM?**

* **Method Area:** Stores class structures like metadata, the constant pool, and the code for methods.
* **Heap:** Runtime data area from which memory for all class instances and arrays is allocated.
* **Stack:** Stores frames. Each frame contains a local variable array and operand stack.
* **Program Counter Register:** Contains the address of the JVM instruction currently being executed.
* **Native Method Stack:** Contains all the native methods used in the application.

**What is the purpose of the super an this keyword in Java?**

### super Keyword- The super keyword is used to refer to the immediate parent class object. It can be used for various purposes:

* **Calling Parent Class Constructors:** super can be used to call a parent class constructor from a child class constructor. This is particularly useful when the parent class has a parameterized constructor.
* **Accessing Parent Class Methods:** super can be used to call a method from the parent class that is overridden in the child class.
* **Accessing Parent Class Fields:** super can be used to access fields of the parent class when they are hidden by fields of the same name in the child class.

### this Keyword- The this keyword is used to refer to the current class instance. It is commonly used in several scenarios:

* **Calling Current Class Constructors:** this can be used to call another constructor of the same class. This is known as constructor chaining.
* **Calling Current Class Methods:** this can be used to call another method of the current class
* **Referencing Current Class Fields:** this is used to differentiate between class fields and parameters with the same name.

**What are the differences between == and equals() in Java?**

* == checks if two references point to the same object.
* equals() method is used to check the equivalence of two objects
* **eg.**

String s1 = new String("hello");

String s2 = new String("hello");

System.out.println(s1 == s2); // false

System.out.println(s1.equals(s2)); // true

**What is a final keyword in Java?**

* The final keyword is used to define constants, prevent method overriding, and inheritance.
* Eg. final int MAX = 100; // Constant final class
* Final Class { } // No class can inherit this class
* class A { final void method() { } // Cannot be overridden }

https://media.licdn.com/dms/image/D4D22AQHJieSOZPmZcg/feedshare-shrink\_800/0/1720178839991?e=1723680000&v=beta&t=NYJb-nro32-KYa8aZrmT9DNkevgFrDrG6r2gLCKc9w8