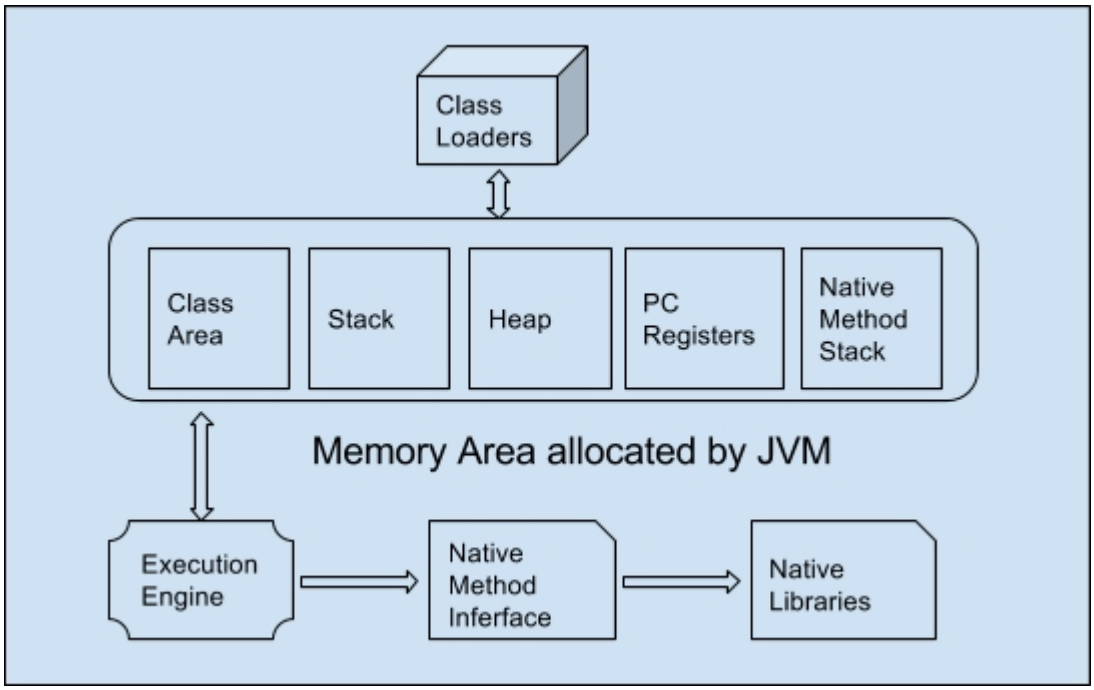
1. **Explain the JVM architecture?**



2. **Classloader** – Loads the class file into the JVM.
3. **Class Area** – Storage areas for a class elements structure like fields, method data, code of method etc.
4. **Heap** – Runtime storage allocation for objects.
5. **Stack** – Storage for local variables and partial results. A stack contains frames and allocates one for each thread. Once a thread gets completed, this frame also gets destroyed. It also plays roles in method invocation and returns.
6. **PC Registers** – Program Counter Registers contains the address of an instruction that JVM is currently executing.
7. **Execution Engine** – It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.
8. **Native method stack** – It contains all the native methods used by the application.
9.

2. What is the use of Classloader in Java?

It loads classes from different resources. Java ClassLoader is used to load the classes at run time. In other words, JVM performs the linking process at runtime. Classes are loaded into the JVM according to need.

3. What is the inner and anonymous inner class?

A local inner class consists of a class declared within a method, whereas an **anonymous class is declared when an instance is created**. So the anonymous class is created on the fly or during program execution. An anonymous class has no name declared for the class, which differs from the local inner class, which has a class name.

4. What is break and continue statement?

The break and continue statements are the jump statements that are used to skip some statements inside the loop or terminate the loop immediately without checking the test expression. These statements can be used inside any loops such as for, while, do-while loop.

What is an instanceof keyword?

The instanceof keyword **checks whether an object is an instance of a specific class or an interface**. The instanceof keyword compares the instance with type. The return value is either true or false.

5. What is aggregation and composition in Java?

Composition

Composition is a “belongs-to” type of relationship. It means that one of the objects is a logically larger structure, which contains the other object. In other words, it's part or member of the other object.

Aggregation

Aggregation is also a “has-a” relationship. What distinguishes it from composition, that it doesn't involve owning. As a result, the lifecycles of the objects aren't tied: every one of them can exist independently of each other.

6. What do you understand about Thread Priority?

Thread Priority in Java: Priority means the number of resources allocated to the particular thread. Every thread created in JVM is assigned with a priority. The priority range is between 1 and 10.

7. What is Thread Scheduler and Time Slicing?

A component of Java that decides which thread to run or execute and which thread to wait is called a **thread scheduler in Java**.

8. Explain the Java Exception Hierarchy.

What happens when an exception is thrown by the main method?

When exception is thrown by main() method, ***Java Runtime terminates the program and print the exception message and stack trace in system console****. The throws clause only states that the method throws a checked FileNotFoundException and the calling method should catch or rethrow it.*

10.What are the different types of variable can be defined in a program ?

1. Local Variables

A variable defined within a block or method or constructor is called a local variable.

2. Instance Variables

Instance variables are non-static variables and are declared in a class outside of any method, constructor, or block.

3. Static Variables

Static variables are also known as class variables.

- These variables are declared similarly as instance variables. The difference is that static variables are declared using the static keyword within a class outside of any method, constructor or block.

10.What is the Java Collections Framework?

Java collection framework represents a hierarchy of set of interfaces and classes that are used to manipulate group of objects.

Java collections framework is contained in java.util package. It provides many important classes and interfaces to collect and organize group of objects.

12.Explain the difference among Hash Table and Hash Map?

HashMap vs Hashtable

Comparison Chart

HashMap	Hashtable
HashMap is an unsynchronized Map.	Hashtable is a synchronized Map.
It is not thread-safe and cannot be shared between multiple threads without proper synchronization.	It is thread-safe and can be shared between multiple threads.
It supports all Map operations and allows multiple null values but only one null key in a collection so that it could maintain unique key properties.	It does not support null values and null keys because there is null check in the put method implementation of Hashtable.
It is much faster and better than a Hashtable in terms of performance.	It is a bit slower than a HashMap but faster than a synchronized HashMap.
HashMap performs better in a multi threaded environment.	Hashtable performs better in a single threaded environment.
It uses iterator to iterate the values of HashMap object.	It uses iterator and Enumerator to iterate the Hashtable object values.
The iterator is fail-fast meaning it throws an exception if the system fails.	The enumerator in Hashtable does not have this behavior.

13.Explain the difference among HashSet and Hash Map?

Basis	HashMap	HashSet
Definition	Java HashMap is a hash table based implementation of Map interface.	HashSet is a Set. It creates a collection that uses a hash table for storage.
Implementation	HashMap implements **Map, Cloneable, and Serializable** interface es.	HashSet implements **Set, Cloneable, Serializable,** **Iterable** and **Collection** interfaces.
Stores	In HashMap we store a **key-value pair**. It maintains the mapping of key and value.	In HashSet, we store **objects**.
Duplicate values	It does not allow **duplicate keys**, but **duplicate values** are **allowed**.	It does not allow **duplicate values**.
Null values	It can contain a **single null key** and **multiple null values**.	It can contain **a single null value**.
Method of insertion	HashMap uses the **put()** method to add the elements in the HashMap.	HashSet uses the **add()** method to add elements in the HashSet.
Performance	HashMap is **faster/ than HashSet because values are associated with a unique key.**	HashSet is **slower** than HashMap because the member object is used for calculating hashcode value, which can be same for two objects.
The Number of objects	Only **one** object is created during the add operation.	There are **two** objects created during put operation, one for **key** and one for **value**.
Storing Mechanism	HashMap internally uses **hashing** to store objects.	HashSet internally uses a **HashMap** object to store objects.
Uses	Always prefer when we do not maintain the **uniqueness**.	It is used when we need to maintain the **uniqueness** of data.

Example	**{a->4, b->9, c->5}** Where **a, b, c** are **keys** and **4, 9, 5** are **values** associated with key.	**{6, 43, 2, 90, 4}** It denotes a set.

15.How many types of memory areas are allocated by JVM?

1. The Method or Class Area: This type stores details, that has class level like static variables, class name, methods, constant pool, etc. When JVM starts, it creates the method area. The method or class area size can be constant, or it can also vary. If its memory is not sufficient, JVM throws OutOfMemoryError.
2. The Heap Area: It usually stores the objects instantiated by the application. These objects that are present in the heap can be split between threads. Usually, programmers restrict their size to avoid garbage collection pauses.
3. Stack Area: JVM create one run-time stack for every thread. That is present in the stack area. The stack area is used to store either the data or partial results of the returning value from the methods. Once the thread stops, the run-time stack will get destroyed.
4. Program Counter Register or PCR: Each thread has a separate Program Counter Register or PCR. This type is basically used to store addresses of current execution instruction of a particular thread.
5. The Native Method Stacks: Similar to PCR, separate native method stacks are created for every thread. Its purpose is to store the native method information.

16.Timeslice

Time slice :

It is timeframe for which process is allotted to run in preemptive multitasking CPU. The scheduler runs each process every single time-slice. The period of each time slice can be very significant and crucial to balance CPUs performance and responsiveness.