**Q. what are the different feature of Java 8?**

Ans:-Java 8 comes into picture for the functional programming approach. To Promote functional programming approach it comes with different feature.

1. Lambda Expression:- It is a concise way to represent anonymous function using functional interface. Function interface is nothing but an interface which has only one abstract method in it. It should be annotated with @FuctionalInterface annotation.

2.Stream API:- Stream API is the most powerful tool of java 8 as it is use to process the collection API. Using Stream API we can filter ,map, collect,reduce the collections.

3.Default and static method:- Default and static method in interface is introduced in the java 8. Using these method we can define the method inside the interface.

4.Method reference:- Method reference is another feature of java 8 using method reference we can easily call the method of the class using :: colon we just need to specify the class name after that we just need to use :: and we need to write the method name to call it. It is like shorthand to call the method.

5.Optional:- It is a kind of container object which may or may not contain non-null values. It is designed to avoid null pointer exceptions in java programs.

**Q:- Can one Functional interface extend another functional interface?**

Ans:-yes, Functional interface can extend another functional interface provided parent class has only one abstract method and child class has only default or static method. It shouldn’t have any other abstract method in it.

**Q:-What is Function Programming?**

Ans:-It’s a pattern that treats computation as the evaluation of mathematical function and avoids changing the state and mutable data.

If you see stream, in will perform operation one time if we try to use same stream it’s not going to happen and in stream it didn’t change the value of the existing variable it takes the value from the variable and execute the operation and stores inside their own heap area until and unless we need to collect and need to store in other variable.

**Q. Why default and static method introduced in java 8?**

Ans:- Default method is introduced for backward compatibility, backward compatibility means suppose there is some change in the interface then we need to write in the implemented class. It will create extra line of code in implemented class which is working properly so we can directly write in the default method in the interface which can be used in the future.

Static block is like a initialisation block in the interface which is use to define the method in the interface that is only be used in the interface and we know that static method are class method so these method are introduced so that it is only be used for the purpose of the interface.

**Q) What is FlatMap and how it is different from map?**

Ans:-Map is used for data transformation, it works on entire stream and gives one output for one input, it mean if you give one stream it will provide one stream. It mean it will do one to one operation

Suppose

input[2,3,4,5]-->[2,3,4,5]

But Flat Map is combination of flat and map it also does data transformation but suppose the is stream of stream like list of list if we do Flat map operation it will provide one stream from stream of stream input like

Suppose

Input ->[[2,3],[3,4]] output->[2,3,3,4]

**Q) What is Multi Threading?**

Ans:-Multi Threading in java is like multiple worker in a single factory which does different task at the same time so in case of java multiple threads do different sub-processes which is independent of each other.

**Q) What are different types of Thread in java?**

Ans:-1. Main Thread- This thread is a thread which is managed by the application code itself.

2.Daemon Thread:- This Thread looks after the auxiliary work like garbage collection and other thing it has less priority

3.Slave/worker thread :- This thread is created if we want to do some subprocess which are not related with other processes.

**Q)What are the different stage of the Thread?**

Ans:-A) New B)Runnable C)Running(wait,sleep, notify) d) terminate

**Q What are reflection api**

Ans:- In java, the Reflection api is a set classes in the java.lang.reflect package that allows a java program to inspect and manipulate itself at runtime. Method used in reflection api are getclass , Class.forName. isInstance, of Instance.

**Q) can we overload run method in multi threading?**

ans:-You can overload run() method in multithreading in java. In java we can achieve multiThreading

⇒ By Extending Thread

⇒ By implementing Runnable

Runnable interface has only one method i.e. run method and thread class extend runnable method and internally start method of the Thread class use the run method.

So whenever we overload the run method , the overloaded run method will acts like an normal method as the signature of it will be different.

**Q How JVM memory management work?**

Ans:-

Java Virtual Machine (JVM) memory management is a vital process that ensures efficient allocation, use, and recycling of memory during the execution of Java applications. It consists of several key components and processes:

**1. JVM Memory Structure:**

The JVM memory is divided into several regions:

* **Heap Memory:** Used to store objects and class instances. It is the primary area for dynamic memory allocation.
  + *Young Generation:* Stores short-lived objects and is further divided into:
    - **Eden Space**: Where new objects are created.
    - **Survivor Spaces (S0 and S1)**: Where objects are moved after surviving a garbage collection cycle in Eden.
  + *Old Generation:* Stores long-lived objects.
* **Method Area (Metaspace in Java 8+):** Stores class metadata, static variables, and method data.
* **Stack Memory:** Stores method call frames, including local variables and partial results.
* **Program Counter (PC) Register:** Holds the address of the currently executing instruction.
* **Native Method Stack:** Used for native (non-Java) method calls.

**2. Garbage Collection (GC):**

Garbage Collection is an automatic process by which the JVM reclaims memory by destroying unreachable objects. Common GC algorithms include:

* **Serial GC:** Single-threaded collector suitable for small applications.
* **Parallel GC:** Uses multiple threads for minor collections.
* **CMS (Concurrent Mark Sweep):** Reduces pause times by performing most work concurrently.
* **G1 (Garbage First) GC:** Divides heap into regions and collects in parallel; suitable for large heap sizes.

**3. Memory Allocation Process:**

* New objects are created in the Eden space.
* When Eden fills up, a Minor GC occurs, and surviving objects are moved to Survivor spaces.
* After multiple GCs, long-lived objects are promoted to the Old Generation.
* A Major GC (or Full GC) cleans the Old Generation.

**4. Memory Leaks in JVM:**

Even with automatic GC, memory leaks can occur if references to unused objects are unintentionally maintained. Tools like VisualVM, JConsole, and heap dump analyzers help identify leaks.

**5. JVM Tuning:**

Memory management can be optimized using JVM options:

* -Xms and -Xmx: Set initial and maximum heap size.
* -XX:NewSize and -XX:MaxNewSize: Control young generation size.
* -XX:+UseG1GC: Use G1 garbage collector.

**Q What is Garbage collection and can we implement garbage collection manually in java?**

Ans:- **Garbage Collection in Java**

Garbage Collection (GC) in Java is the process by which the JVM automatically deallocates memory by removing objects that are no longer reachable in the application. This helps in preventing memory leaks and reducing the burden of manual memory management on developers.

**How It Works:**

* Every time an object is created using the new keyword, it is stored in the heap memory.
* If no live thread can access an object, it is considered unreachable.
* The GC identifies and removes these unreachable objects to free up memory.

**Types of Garbage Collectors in Java:**

* **Serial GC**: Uses a single thread for both minor and major GC events.
* **Parallel GC**: Uses multiple threads for minor collections, improving performance for multi-core processors.
* **Concurrent Mark Sweep (CMS)**: Aims for low-latency by performing most GC steps concurrently.
* **G1 Garbage Collector**: Designed for applications running on multi-processor machines with large memory spaces.

Yes we can implements garbage collection manually using **System.gc() and finally()** methods

public class ManualGCExample {

public static void main(String[] args) {

// Create a large number of objects

for (int i = 0; i < 1000000; i++) {

DummyObject obj = new DummyObject(i);

}

// Nullify reference to encourage GC

System.out.println("Objects created. Suggesting garbage collection...");

// Suggesting garbage collection manually

**System.gc();**

System.out.println("Garbage collection requested.");

}

}

class DummyObject {

private int id;

private int[] largeArray = new int[1000];

public DummyObject(int id) {

this.id = id;

}

@Override

protected void finalize() throws Throwable {

// Called by GC before object is removed from memory

System.out.println("Finalizing object with id: " + id);

}

}

**Q.Does Finally executes every time , weather there is an error,return statement or system.exit() in a java code?**

**Ans:-**

public class FinallyBlockExample {

public static void main(String[] args) {

try {

System.out.println("Inside try block");

// Uncomment one line at a time to test different scenarios

// return;

// int result = 10 / 0; // Throws ArithmeticException

// System.exit(0); // Terminates JVM, finally will NOT execute

} catch (Exception e) {

System.out.println("Inside catch block");

} finally {

System.out.println("Inside finally block");

}

System.out.println("End of program");

}

}

/\*

Output Analysis:

1. If 'return;' is used => 'finally' executes before returning.

2. If an exception is thrown => 'finally' executes after catch.

3. If System.exit(0) is used => 'finally' does NOT execute.

4.If error occur finally will not executes.

\*/

**Q Explain the role of class loader in java, including bootstrap extensible and system class loader**

**Ans:-** Class loader is the part of JRE(Java Runtime Environment) which dynamically load the class into the JVM during runtime.

There are various types of class loaders like :-

1. **BootStrap Class Loader:-**Load the class from the JDK’s rt.jar(java.lang…,java.util)
2. **Extension Class Loader:-**It load the class from the jar which is external like (mysql libraries)
3. **System Class Loader:-**Load the class from the system.

**Q.Can we call the not static method from static method or static block?**

**Ans:-** No, we can’t call the non static variable from the static method as static method are loaded only once during the loading of the class where as non static variable are loader after object creation. So we can’t call non-static method from static method.

**Q. Discuss the purpose of static method for memory management?**

Ans:- In Java, a **static method** belongs to the class rather than to instances of the class.

**1. Memory Efficiency**

* Static methods are stored in the **Method Area (or Metaspace in Java 8+)**.
* Only **one copy** of the static method is maintained per class, regardless of how many instances are created.
* This helps reduce memory footprint compared to instance methods, which may involve creating multiple object references.

### ****2. Shared Behavior Across Instances****

* Since static methods are shared among all instances, they can provide **shared functionality**, such as logging or configuration management, without redundant object storage.

**Q Why Character-Array is used for password instead of String?**

Ans:-