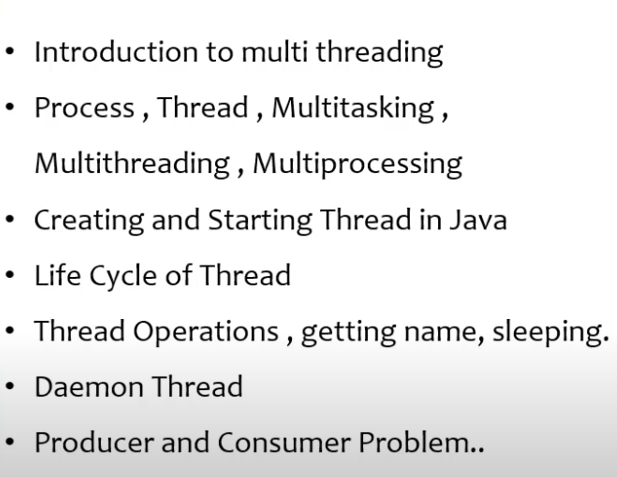
**MULTITHREADING**

Multithreading Notes and details



**205.What is thread? what is multithreading?**

**Thread** are meant for **multi-tasking**.

Process of executing **multiple thread** at a time is called as **multithreading**.

Thread Concepts are not applicable if modules are linked with each other. Or if there is any dependency.

Main purpose of thread is to improve performance by reducing execution time.

**213.Demon thread**

demon thread low priority thread which runs on background to support non demon thread to execute.

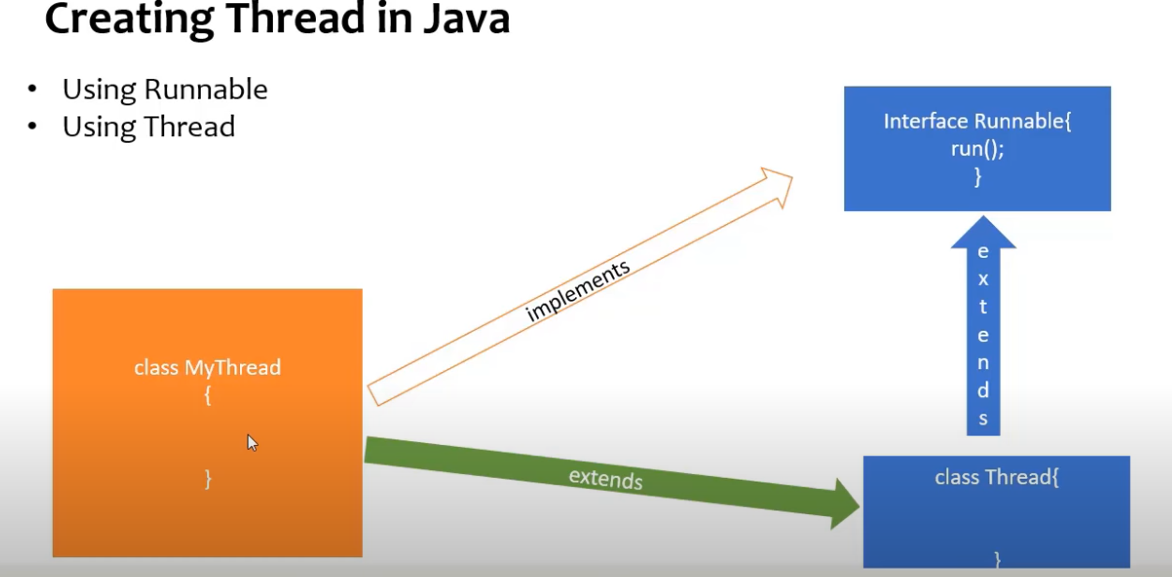
Daemon threads run in the background and perform tasks such as garbage collection.

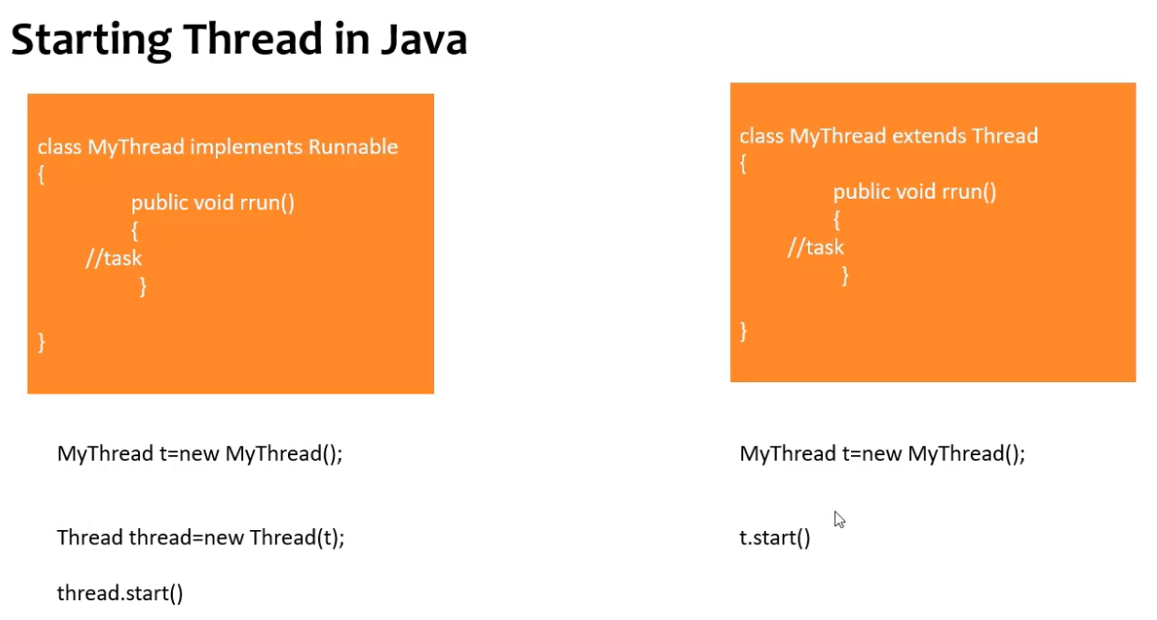
The JVM terminates when all user threads are done, even if daemon threads are still running.

A thread must be set as daemon before it starts, using set Daemon(true).

Example Use Cases: Logging, monitoring, or cleanup tasks.

**Creating a thread**

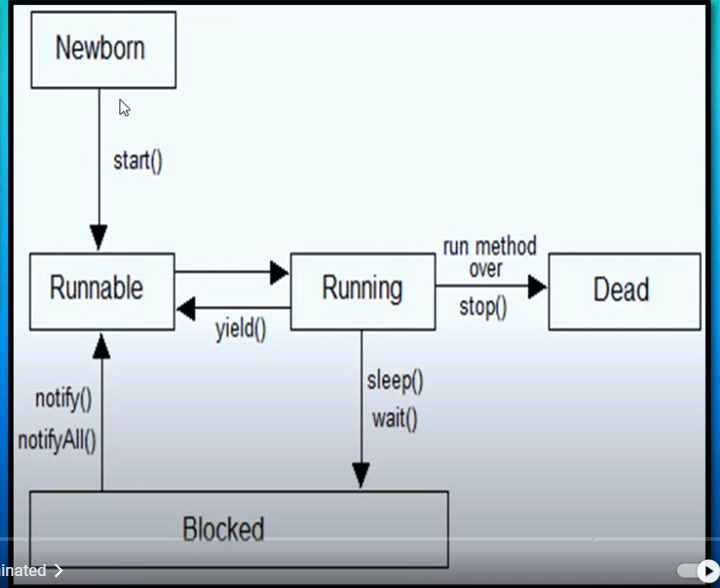
l

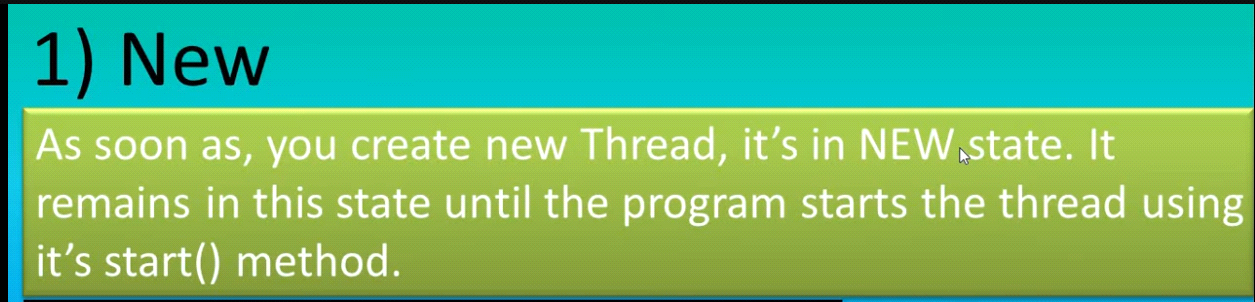


**Thread Life Cycle**

In Java, the **thread lifecycle** represents the various states a thread goes through from its creation to termination.

Java threads are managed by the JVM and can be in one of several states as defined in the Thread. State Enum. These states are:

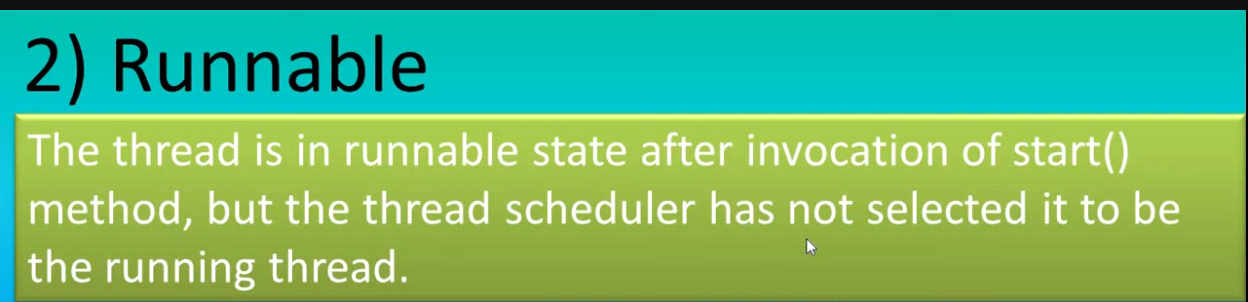




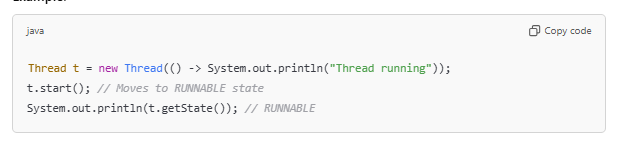


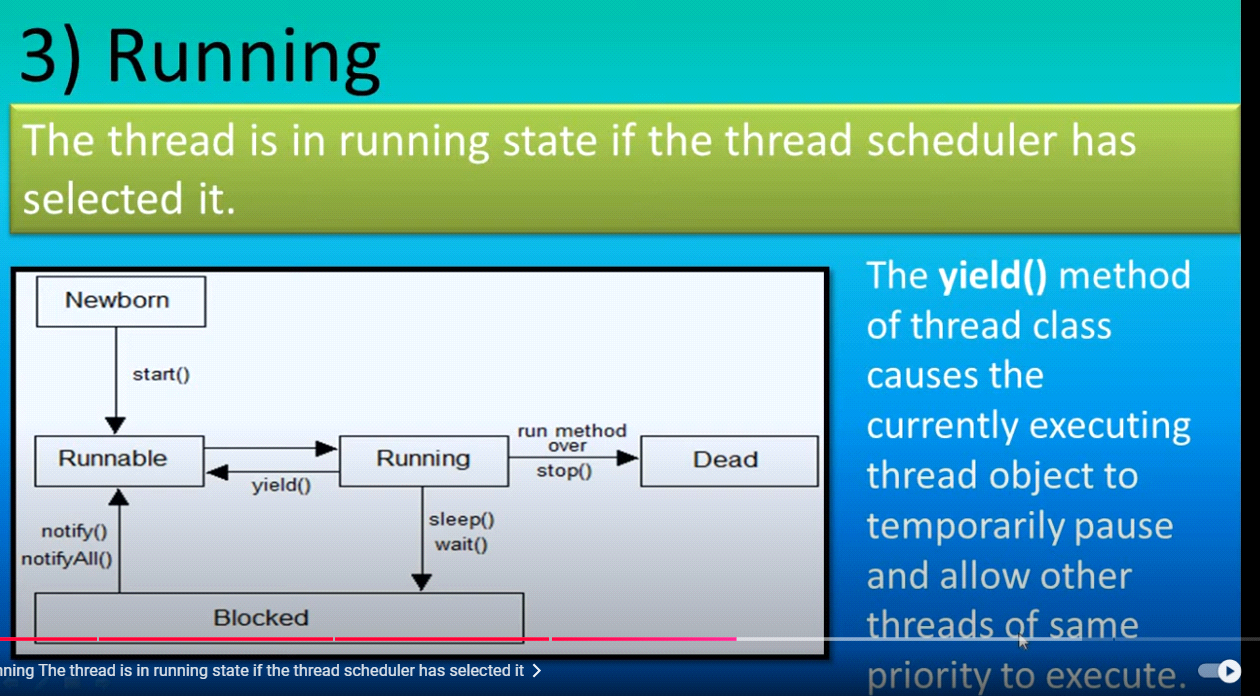
**2. Runnable**

* **State**: The thread is ready to run but may not be actively running. It is waiting for the CPU to schedule it.
* **Action**: The start() method is called, and the thread is eligible to run.
* **Transition**: The thread moves between RUNNABLE and RUNNING states depending on CPU scheduling



**Example-**





**Sleep()**

**Wait() in blocked**





**Key Methods Affecting Thread Lifecycle:**

* **start(): Moves thread from NEW to RUNNABLE.**
* **sleep(): Puts the thread in TIMED\_WAITING.**
* **wait(): Puts the thread in WAITING.**

**notify()**

* Wakes up **one single thread** waiting on the object's monitor.
* If multiple threads are waiting, only one (chosen arbitrarily by the JVM) will be notified.
* The notified thread competes with others for the lock after it is released by the current thread.

**notify All()**

* Wakes up **all threads** waiting on the object's monitor.
* All waiting threads will compete for the lock once it is released by the notifying thread.
* Useful when multiple threads need to proceed, but only one thread will actually acquire the lock
* **join(): Causes the current thread to wait until another thread completes.**

**Process to start thread**



**2nd process from thread class**



**215.Process to create thread and which is best and why**



**where thread are used**

Threads are used in programming to achieve multitasking and improve performance by allowing concurrent execution of multiple parts of a program. Here are some common use cases where threads are utilized:

**Real-Time Applications**

**Purpose: To ensure timely execution of critical tasks.**

**Example:**

**A flight simulation program may use threads to update the simulation environment, track user inputs, and calculate trajectories.**

Difference bewen wait and sleep

Gaming

Purpose: To handle multiple tasks like game logic, rendering, and user input simultaneously.

Example:

One thread processes game physics, another handles rendering graphics, and another listens for user input.

**Key Differences:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **notify()** | **notifyAll()** |
| **Number of Threads** | Wakes up one thread only | Wakes up all waiting threads |
| **Usage** | Suitable when only one thread needs to proceed | Suitable when all waiting threads may need to proceed |
| **Lock Acquisition** | Notified thread competes for the lock | All notified threads compete for the lock |

**DRAWBACK OF THREAD:**

**Drawback of Thread:**

* When two threads are operating on a common data then Data corruption will happen. So to prevent data corruption we apply Thread Synchronization and this Thread Synchronization is the drawback.
* We use threads for better performance but When we apply thread synchronization to avoid data corruption, here **it slows down the execution time** as one thread is kept in wait status and the other threads keeps on executing.
* When two or multiple people is trying to book a ticket at same time, then it might be same ticket will booked for two or multiple persons.
* For thread we **can‟t set priority** which thread will run first.

***212. Thread synchronization*** *:*

* When two threads are working with common data concurrently, there is a risk of data corruption due to multitasking.
* We use the „synchronized‟ keyword to control. This keyword is applied to a method or a block of code, making sure that only one thread can execute that block at a time.
* In a synchronized block, when a thread acquires the lock (permission to execute), it can exclusively operate on the shared data. Other threads that attempt to access the synchronized block have to wait until the first thread releases the lock. Once the lock is released, one of the waiting threads is granted the lock, and it can then execute the synchronized block. This sequential access helps prevent data corruption and ensures the orderly execution of threads.

***What is join:***

* When a thread performs a task and pauses, another thread can pick up the work from where the first thread left off. This cooperative working approach is known as 'joining' threads. Thread 1 works until a certain point, then Thread 2 takes over from there. When Thread 2 pauses, Thread 1 resumes its work from the last point. This cycle continues, allowing threads to collaboratively complete a task.

**206.THREAD POOL**

**Thread Pool**

* A thread pool is a collection of threads. As soon as the pool has an idle thread, a task is assigned to one of them and executed.
* Thread pools are useful when you need to limit the number of threads running in your application simultaneously,
* It improves overall performance.
* Instead of creating a new thread for each task, tasks can be assigned to a thread pool for execution.

**Continuation:**

* Thread pools are often used in servers.
* Each connection arriving at the server via the network that is wrapped as a task and passed on to a thread pool.
* The threads in the thread pool concurrently process requests from the connections.

**Thread Safe? How to achieve it?**

In Java, **thread safety** refers to the ability of a program, method, or class to be safely used by multiple threads at the same time without causing any issues like data corruption or inconsistent behavior.

When multiple threads (or tasks) access the same resources (like variables, data structures, or files) concurrently, thread safety ensures that these threads don’t interfere with each other in a harmful way.

**Key Concepts of Thread Safety**

* **Shared Resources**:  
  Multiple threads might try to read or modify the same resource (like a variable or object) at the same time. Without proper control, this can lead to problems like **race** conditions or inconsistent states.
* **Race Conditions**:  
  A race condition occurs when multiple threads access shared data at the same time, and the result depends on the timing of thread execution. This can cause errors or unexpected behavior.
* **Critical Section**:  
  A critical section is a part of the code where shared resources are accessed or modified. It's important to ensure that only one thread can execute this section at a time to avoid conflicts.

**Ways to Achieve Thread Safety in Java**

* **Synchronization**:
* You can use the synchronized keyword to make sure that only one thread can execute a particular piece of code at a time.
* This can be applied to methods or specific blocks of code.

**Example**:

java

Copy code

public class Counter {

private int count = 0;

public synchronized void increment() {

count++;

}

public synchronized int getCount() {

return count;

}

}

* The synchronized keyword ensures that when one thread is executing the increment() method, no other thread can execute it simultaneously.
* **Volatile Keyword**:
* The volatile keyword ensures that changes made by one thread to a variable are immediately visible to all other threads.
* This is useful for simple flags or variables.

**Example**:

java

Copy code

private volatile boolean flag = false;

public void setFlagTrue() {

flag = true; // Visible to all threads immediately

}

* **Atomic Classes**:
* Java provides **atomic classes** in java.util.concurrent.atomic (like AtomicInteger, AtomicBoolean, etc.) that perform thread-safe operations without the need for synchronization.
* These classes use low-level concurrency mechanisms like Compare-And-Swap (CAS) to safely update variables.

**Example**:

java

Copy code

import java.util.concurrent.atomic.AtomicInteger;

public class AtomicCounter {

private AtomicInteger count = new AtomicInteger(0);

public void increment() {

count.incrementAndGet(); // Atomically increments the value

}

public int getCount() {

return count.get();

}

}

* **Thread-Safe Collections**:
* Java provides thread-safe collections like ConcurrentHashMap, CopyOnWriteArrayList, etc., which handle concurrency internally and allow multiple threads to access them safely.

**Example**:

java

Copy code

import java.util.concurrent.ConcurrentHashMap;

public class ThreadSafeMap {

private ConcurrentHashMap<String, Integer> map = new ConcurrentHashMap<>();

public void putValue(String key, Integer value) {

map.put(key, value); // Thread-safe operation

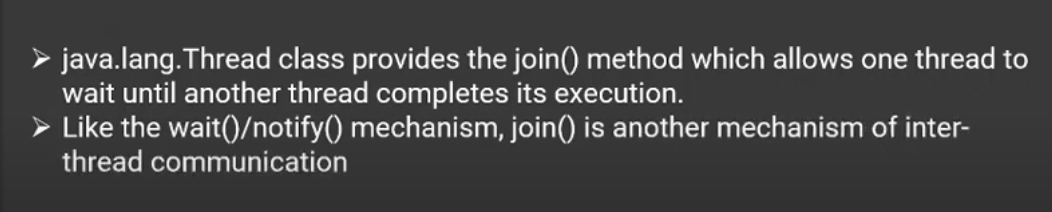
}

}

**208. Difference between thread and process**

|  |  |  |
| --- | --- | --- |
| Feature | Process | Thread |
| definition | A heavyweight, independent execution unit with its own memory space. | A lightweight, smaller execution unit within a process. |
| memory | Separate memory space | Shared memory with in a process. |
| Communication | Slower(inter process communication ) | Faster (shared memory used so faster. |
| overhead | Higher . | Lower. |
| dependency | Independent of other process | Dependent on other threads in same process. |
| creation | Required more resource | Faster because of light weight |
| Use case | Running independent programs | Concurrent tasks with in same program. |

**209. What is thread join?**

****

In Java, the Thread.join() method is used to ensure that the current thread waits until the thread on which join() is called has completed its execution. It is particularly helpful when you want one thread to complete before the current thread continues its execution.

**Key Points:**

1. **Purpose:** To make the current thread wait for another thread to finish.
2. **Signature:**
   * void join(): Waits indefinitely for the thread to finish.
   * void join(long millis): Waits for the thread to finish for a maximum of the specified milliseconds.
   * void join(long millis, int nanos): Waits for the thread to finish for a maximum of the specified milliseconds and nanoseconds.

**Exception:** It throws InterruptedException if the current thread is interrupted while waiting.

**214. Difference between thread join and thread sleep method**

The Thread.sleep method pauses the execution of the current thread for a specified amount of time. It is commonly used to introduce delays in execution or simulate time-consuming tasks like loading, waiting for external resources, or pacing an application.

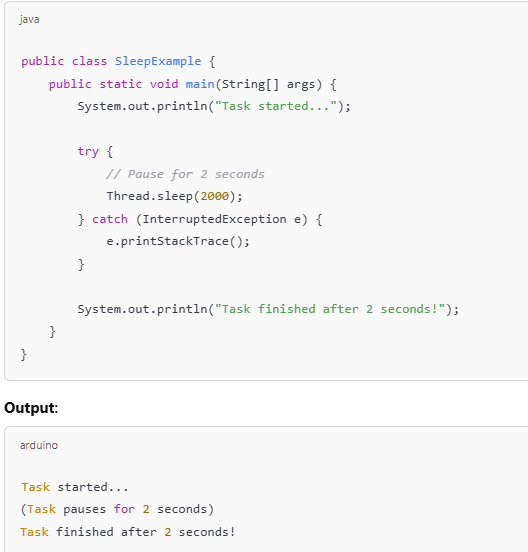
 **Static Method**: It is a static method of the Thread class, so it always pauses the currently executing thread.

 **Duration**:

* Accepts the duration in milliseconds.
* An optional nanoseconds argument can also be used.

 **Throws Exception**: Throws InterruptedException if the thread is interrupted while sleeping.

 **State**: The thread enters the TIMED\_WAITING state during sleep.

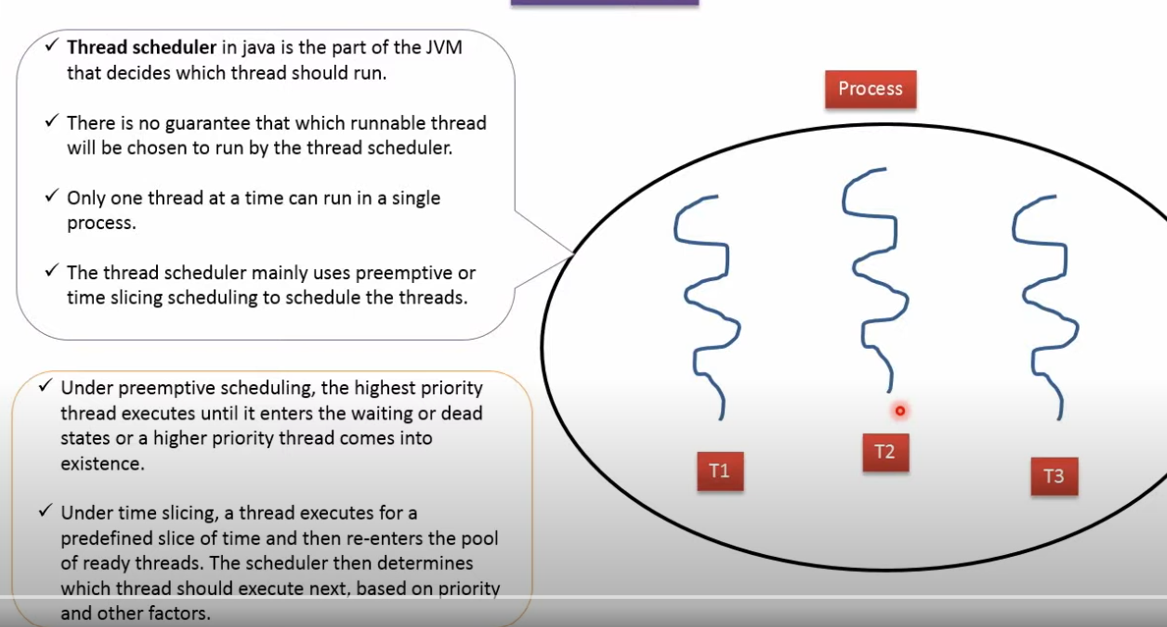


**Thread.join()**

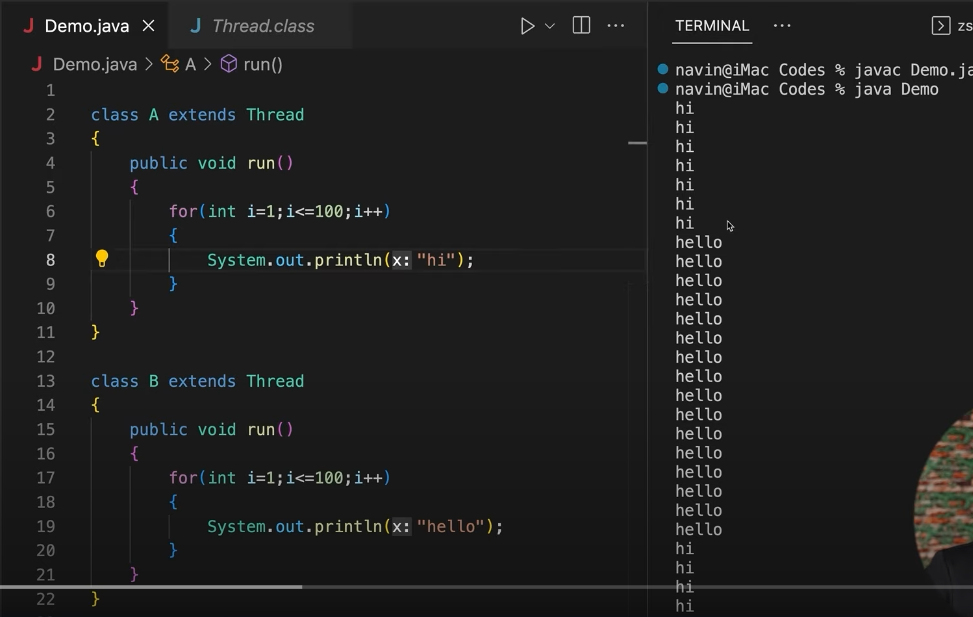
If we **want to wait** for a thread to finish its task **before moving on**. That’s where join() comes in.

Useful when you want threads to run sequentially.

**216. What is thread scheduler**

****

**217. Explain about priority of thread?**

****

**217.Explain about priority of thread?**

Threads with higher priority are typically executed before threads with lower priority.

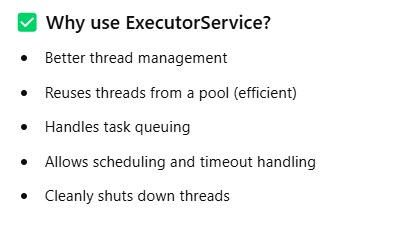
However, thread priority is not a guarantee of execution order since it is dependent on the **thread scheduler** implementation, which is platform-dependent.



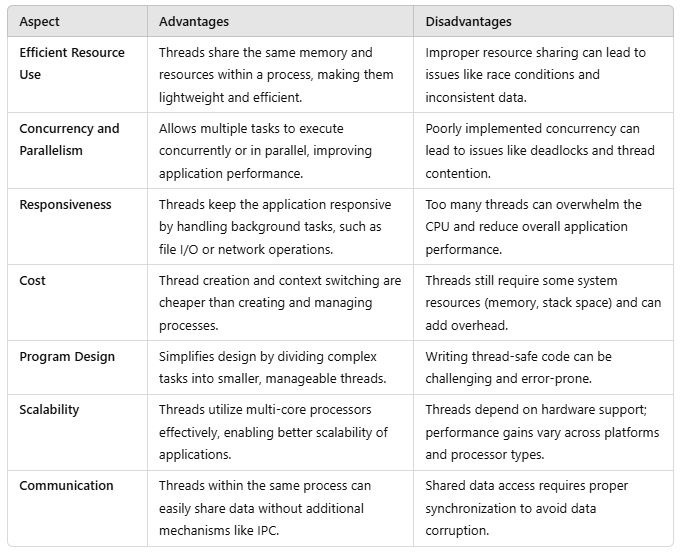
**218. What are executor services**

**ExecutorService is part of Java’s concurrency framework. It's used to manage a pool of threads to run tasks asynchronously without manually creating and starting threads.**

****

****

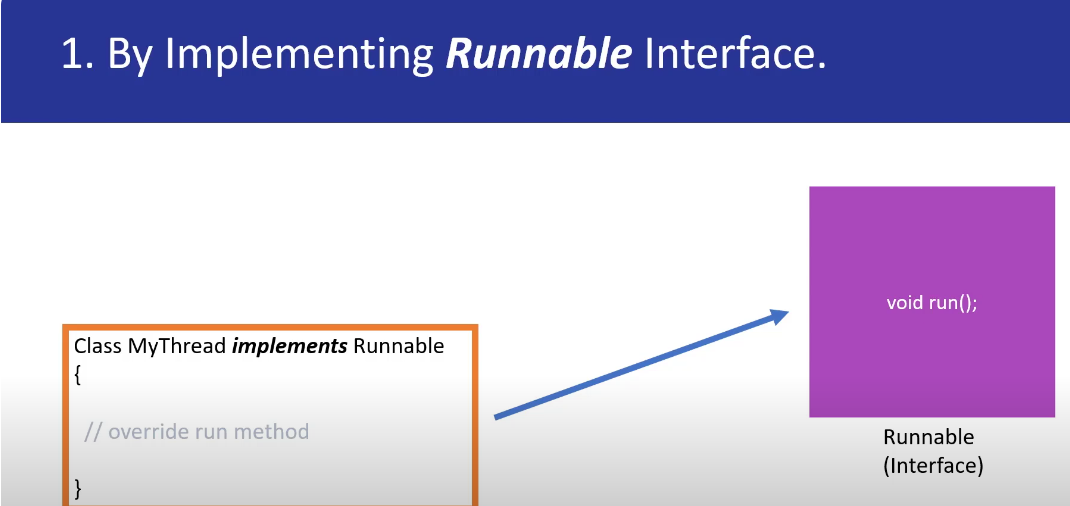
**220.Advantage and Disadvantage of thread**

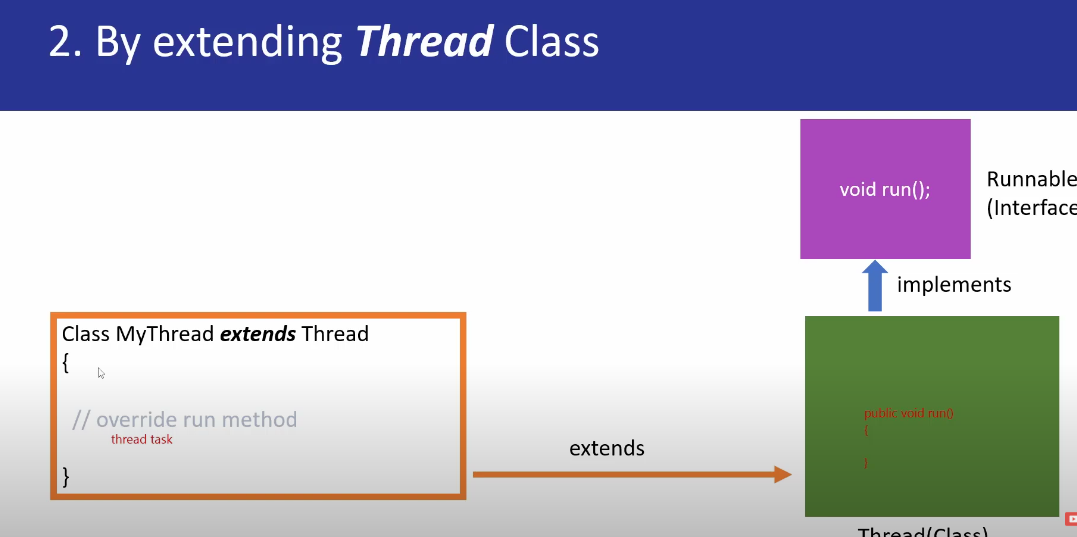


**224. Define Threads and its types/ Define Threads and its types/** Ways to create thread?

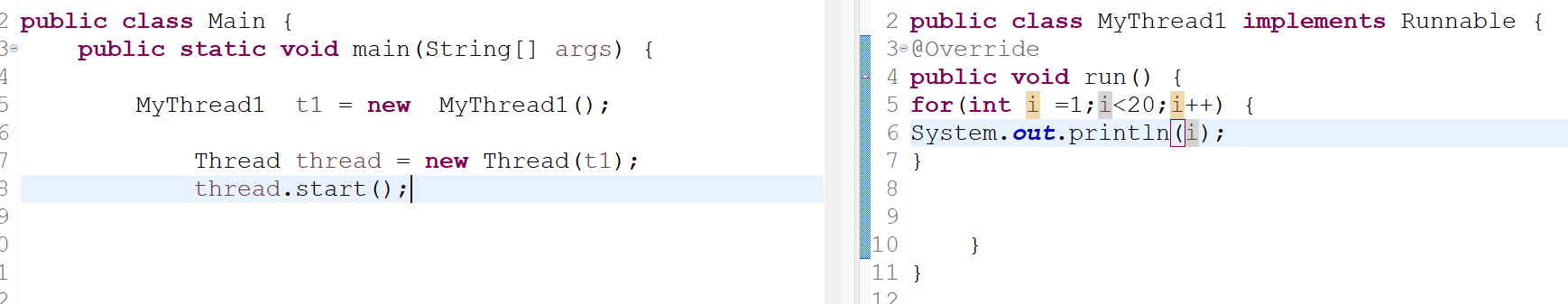
****

<https://www.youtube.com/watch?v=tHgCt6TRrWA>

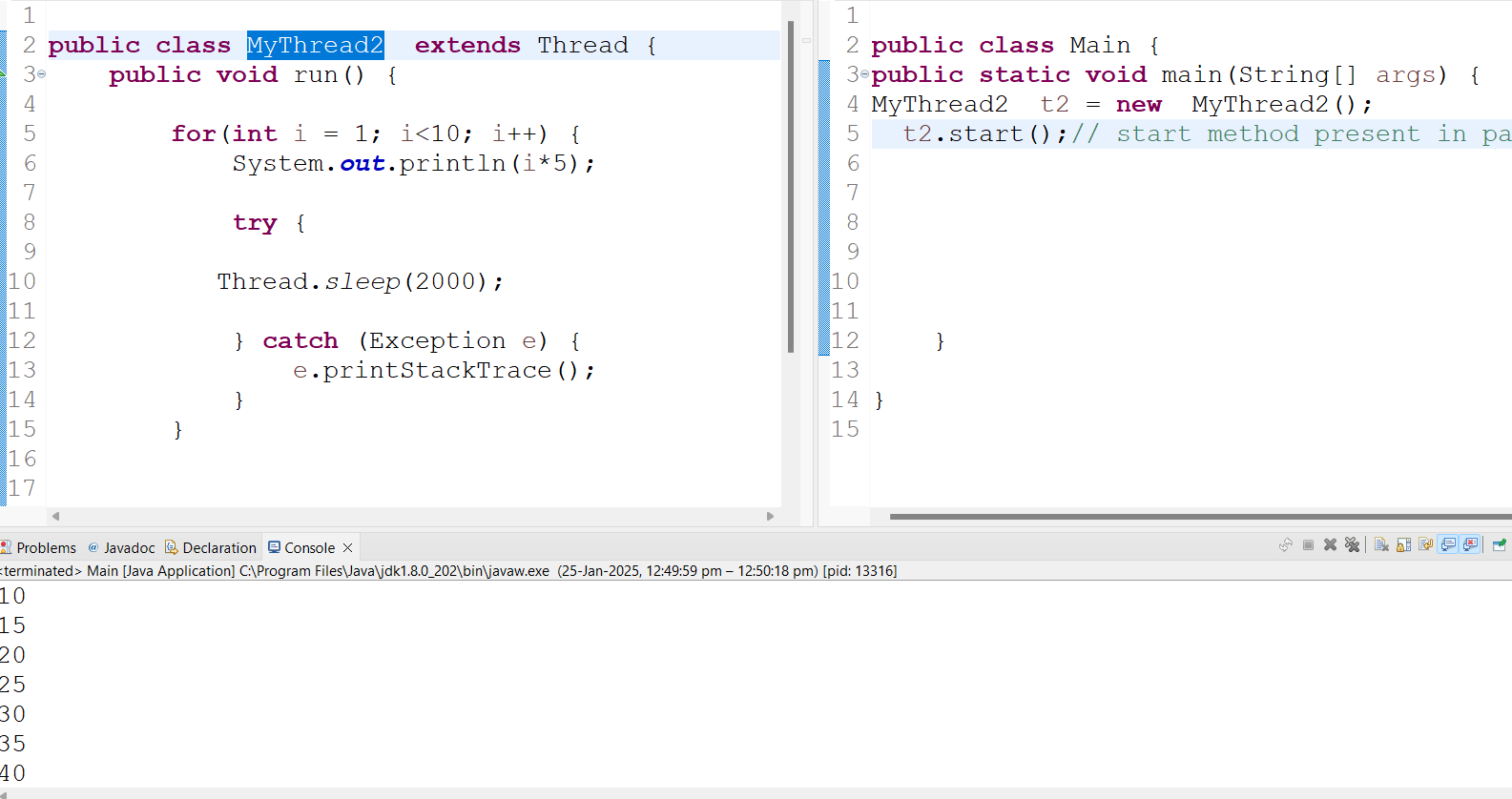




Thread1



Thread-2



**Which is recommended?**

**233.What is object lock vs class lock**

**Object Lock**

A lock associated with a specific instance of a class.

Ensures that only one thread can access synchronized instance methods or blocks of the same object at a time.

**How It Works**:

When a thread acquires an object lock, other threads are prevented from executing any synchronized instance method/block of that particular object.Each object has its own lock.



**Class Lock**

* **Definition**: A lock associated with the .class object of a class, rather than a specific instance.
* **Purpose**: Ensures that only one thread can access synchronized **static methods/blocks** of a class at a time.
* **How It Works**:
  + When a thread acquires a class lock, it prevents other threads from executing any synchronized **static method/block** of the same class.
  + Class locks are associated with the **class object**, not instances.

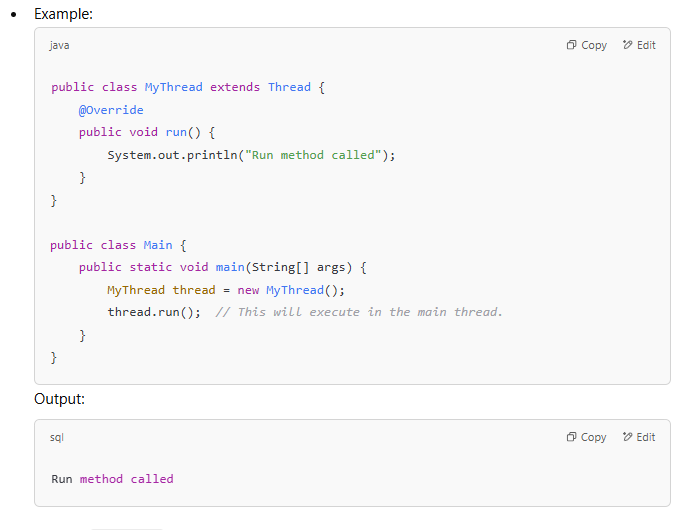


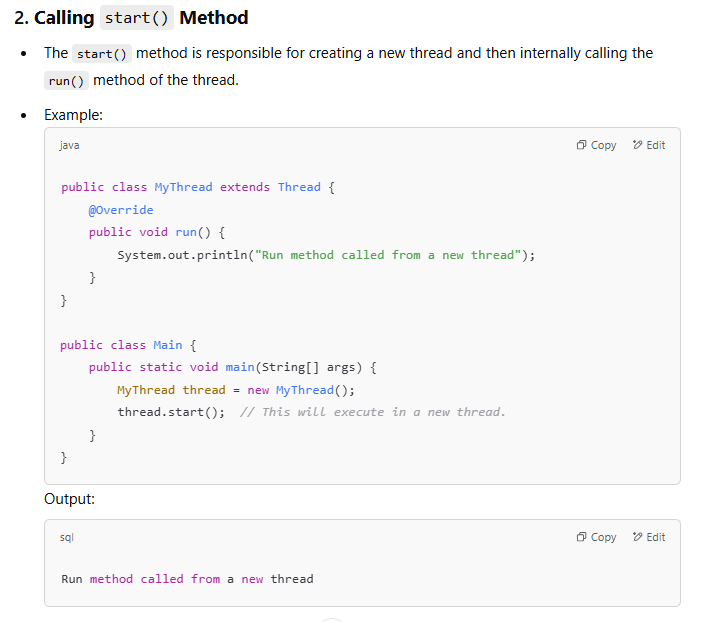
**235.Can we call run method before start method**

**Yes, you can call the run() method directly before calling the start() method in Java, but it will not create a new thread. Here's a detailed explanation:**

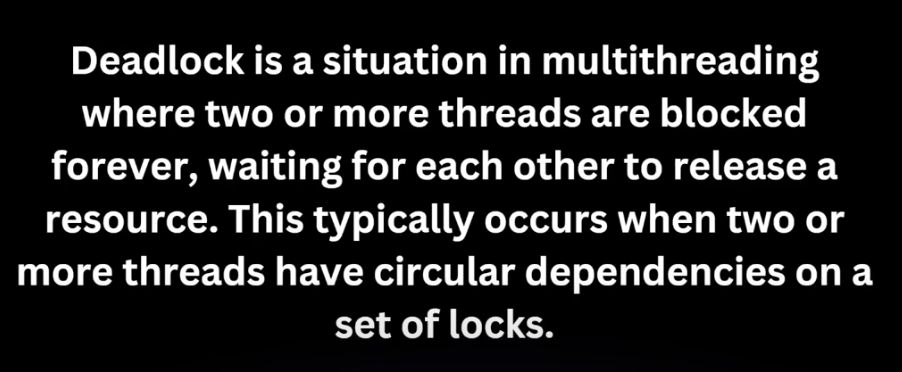
**1. Calling run() Directly**

* **The run() method is just a regular method in the Thread class or your custom class implementing Runnable.**
* **If you call run() directly, it will execute on the current thread instead of creating a new thread**

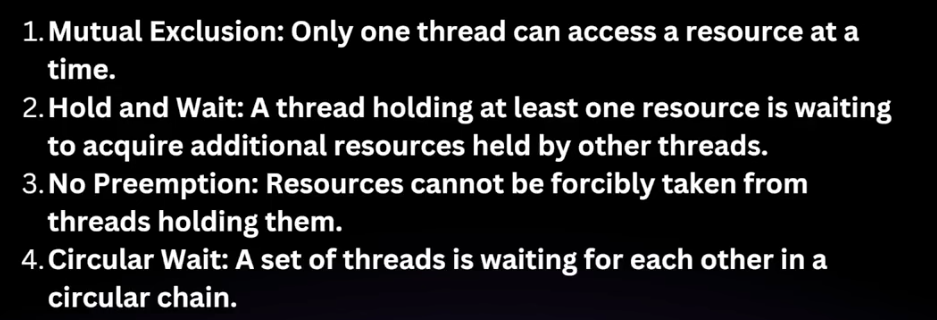
****

****

**238.** What is deadlock and how to avoid it?

**+**

**When dead lock happens**

****

**What is thread joiner?**

**How to avoid it?**

**239.What is thread safe? Benefits of thread safe.**

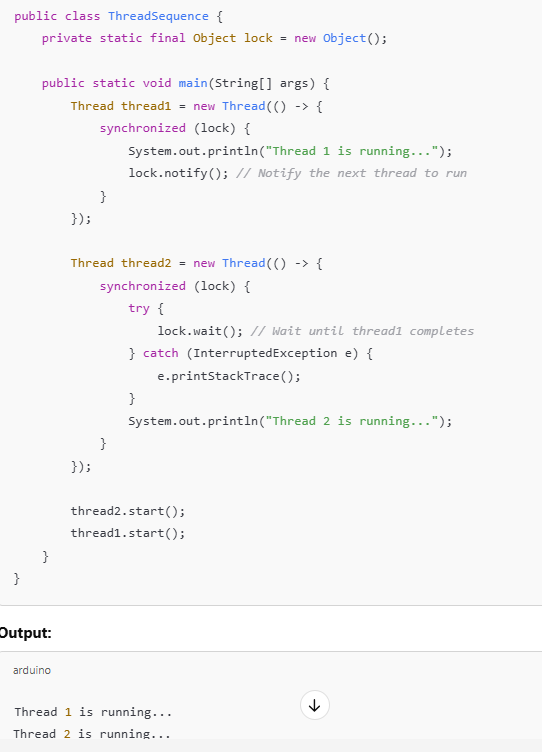
**A piece of code, object, or class is considered thread-safe if it functions correctly when accessed or modified by multiple threads concurrently, without causing any race conditions, inconsistent data, or unexpected behavior.**

**Thread safety ensures that shared resources (e.g., variables, data structures) are accessed in a controlled and predictable manner in multithreaded applications.**

****

**240.How to make 2 threads run one after another.**

To make two threads run **one after another** in Java, you can use synchronization techniques to ensure that one thread completes its execution before the other starts. Below are several ways to achieve this



**Most important Udemy part 2**

1. Tell us about yourself

-> Can you tell us about yourself or can you talk about your work experience while walking us through your

résumé, no matter what the question is and how it is asked, this icebreaking question is a great opportunity

to market our skills and all the great work we have been doing over the years.

-> It is like fishing where we will leave several technical baits hoping that the interviewer will be caught

in one or most of them and ask us questions from those topics as the interview progresses.

-> This is the very first question we will be responding to, and it conveys a lot about our communication,

confidence and experience.

-> So we should prepare six to 12 points and practice those points 30 to 40 times, at least in the next

lecture.

-> I will share some important points from my about me that I have been using over the years that will

help you.

-> And you can use it as a template if you want to.

1. About Me Preparation Template

-> I'll walk you through some of the important points from my about me, which I have been using over the

years, I'll always start by saying I have started my career as a Java developer in 2002.

-> Since then, I have worked across domains ranging from ERP to CRM banking to finance, project management

to health care and travel to gaming.

-> This point tells them my overall experience and also my experience across various domains, which can

be very important for certain roles.

-> Next, I move on to the technology side where I state while working across the Java technology full stack,

I have designed and developed applications using various spring boot modules and also frontend java script frameworks

like Angular and React.

-> So depending on whether it is a full stack position or if it is a backend position, you can change

this point slightly.

-> And based on your experience, whether you are a junior or senior developer etc, you can add more technologies

here if you want to.

-> Next, I have worked closely with the product owners and business analysts to get the requirements right

and with the architects to implement the non-functional requirements like security, scalability, reliability,

etc., This point makes you a real good professional at the same time.

-> It also showcases that you are not just a simple developer, but you can interpret the requirements

and also, if required, you can work with the architects to implement the non functional requirements.

-> Again, if you are a junior developer, you can take out the nonfunctional side of things if you are

not into it yet.

-> Next, we'll move on to the devops side of things where I state

I have also worked with devops engineers to set up the CI/CD pipeline for our project using tools

like Maven,Docker, Jenkins' and Kubernetes.

-> So these are all the bits we are leaving them so that they can ask us more questions on these

later on.So we should be thorough in those topics.

-> In some cases, we may not have the devops team. We ourselves will set up this whole thing.

So depending on your project, you can change this point accordingly.

-> But the key here, state your devops experience and your CI/CD experience and the tools you have

used.

-> I have also worked with various aws components, such as EC2 , S3, EKS and so on, to deploy our apps to

the cloud.

You can modify this to the cloud you have worked with pretty much every job description has a cloud

requirement today.

-> So with which our cloud you are familiar with, you can use that here instead of aws so you can replace

it with Azure GCP, etc..

-> Some projects that do not need cloud, they might be using Kubernetes for their deployment or they

are still migrating to the cloud.

-> In such cases, you can skip it if you do not have that experience.

But if you do have that experience, please state all that in detail.

-> Moving on to the agile side of things, I have participated in several Agile conferences and played

a key role in implementing Scrum and other agile practices like tdd pair programming, etc. So depending

on your experience, you can state your agile experience here.

-> You can say how you have followed the scrum ceremonies, the planning meetings and all that.

You can put it into one point, which is very important.

-> As soon as you get onto a team, they expect you to immediately become agile and be a part of their

agile process that they are following last and very important.

-> I am a continuous learner.

I have recently learned creating serverless projects using AWS lambdas.

-> You can state whatever you have learned recently as this last point because teams need continuous learners

as the industry evolves, as new things evolve.

-> And also, I also like sharing what I learn and I do it through my YouTube channel and blog.

This has helped me tremendously over the years.

-> Not many people share from what I see in the industry, so when you love to share.

People love it because they need mentors.

-> Once you get onto the team, they allow you to share what you are learning on a continuous basis and

also learning from others as well.

-> So whatever you are doing, please add those points at the end.

-> And if you do not have a blogger YouTube channel yet, please do create one right away so you can use

this as a template by downloading it from the resources section of this lecture, but create one of

your own because you are unique.

1. Your recent project

-> Many were asked to explain about your most recent project.

-> Always remember that it is another great opportunity to highlight and sell the skills that you have

used on your recent project for the position you are being interviewed for.

-> Start with the high level use case of the problem you are trying to solve.

For example, if you are working on a flight reservation and checkin application, we can tell them

that our application allows the end users to book their flight tickets and check into the flights online

or through their mobile applications.

-> You then draw the high level architectural diagram, which will show them the various components of

your application.

-> Feel free to use the whiteboard if it is a Face-To-Face interview.

If there is a whiteboard available, grab the whiteboard with their permission, of course, and draw

the high level architectural diagram and show the flow in between these components.

-> If a whiteboard is not available, use a white sheet of paper.

If it is a virtual meeting, feel free to share your screen and paint the picture.

This will leave a lasting impression on them.

-> As you showcase your presentation skills, you can then talk a little more about how you have implemented

these components.

-> If it is completely a Java based full stack development, talk about it.

If the backend is in Java and the Frontend is developed using other JavaScript frameworks, talk about

them as well.

-> You can then jump into the layered architecture of your Java back end typically will have the data access

layer services layer where the business logic leaves the presentation layer and the integration layer

where you can use restful web services or messaging.

-> You can also talk about each of these layers and the technologies that you have used in these layers.

This is very important and this is where you can highlight those technologies that are really required

for the job on hand.

-> For the data access layer typically will use spring data jpa with hibernate for the restful web services

layer.

-> We would have use springboot web or frameworks like Jersey and the list goes on and on.

So we have a lot of technologies which fall into these layers.

-> Again, this is where your responsibilities on your resume under your current project will come in.

So go through all those responsibilities and highlight those portions which are required for the current

job opening, which you are being interviewed for.

-> Don't stop with just the development side of things.

Talk on how you how written unit tests using unit test frame works like Junit, Mockito, etc., how we have reported

the test coverage of those unit tests, the servers to which you have deployed your application

to.

-> If you did that, if you have used messaging, talk about the messaging provider or the broker you

have used and if you have developed the front end using Angular react etc do talk about that as well.

-> Once you are done with the development side of things, move on to the devops side of things.

This is where you highlight your skills, working with one of the build tools like Maven, Gradle,

etc. Talk about how you how Dockerized your project, how you have created the docker file or the docker compose

file for the application components within your architecture.

-> And if the job requirement needs kubernetes, highlight that as a container orchestration tool.

-> How you have depolyed your containers to the kubernetes

cluster if it needs aws or if your last project has used one of these clouds, highlight that

as well.

-> Keep the conversation interactive Don't let it be a one side affair.

-> You can always stop and ask them if they want to know more details of any of these as you mention them,

especially you can ask them if they want to hear more about how you have

implemented these layers at a very low level.

-> If they say yes, you can walk them through the classes and interfaces which you typically create in

a Java application, starting with the model classes, then the data access layer classes, as the service layer

classes, the restful and Web controllers, the utility classes in your application, the validators

views and many more.

-> This will give them a lot of confidence that you are a very hands on guy and you are ready to start

coding whenever it is required.

-> Also, do talk about how you have implemented the non-functional requirements or the ILities, as

we call them, the security, scalability, reliability, adaptability when it comes to your project.

-> If not, they will ask you these questions later on anyways.

They will ask you how did you secure your recent project or what are the security best practices?

How do you ensure that your application is scalable?

And all those questions will come in throughout the interview.

1. Resume Preparation Tips and Template

-> In this lecture, I will give you some résumé preparation tips and also share a resume template

I have been using over the years in the USA.

-> Before we go ahead and do that, launch your Web browser and search for Java developer resumé sample.

You can even be specific here.

-> You can say Junior java developer, Senior Java developer, java lead architect and so on.

Depending on where you are in your career, go to the images part and you will see hundreds and thousands

of resumes you can use as a reference.

-> There is no hard and fast rule.

You can be as creative as you can, but make sure that your resume talks to the recruiter or the interviewer.

-> Here is a template I have been using over the years.

-> I always start with my name and my contact information so that they can reach out to me immediately.

This YouTube channel I mentioned right up top has been helping me not only qualify for the interviews,

but to clear the interview rounds very easily because the recruiters and the interviewers already know

my technical knowledge and communication skills even before they talk to me.

So I suggest you create your own YouTube channel, your own blog, and share it right on top of your

resume.

-> This visa status is important in certain countries and for certain roles you can mention your visa status,

especially in the US.

-> Certain positions need certain immigration status that will help you with that.

-> Next, you can directly move on to the project experience and talk about your proje->ct and the roles

and responsibilities.

-> But I personally like sharing my overall experience right on top of my resume.

This is like a mini about me where I summarize my overall experience.

Very first line tells them that I'm a senior Java consultant with 18 plus years of IT experience and I

have experience with them devops tools and aws cloud components.

-> This point makes me a good fit for several positions which are out there today.

-> Then I also show them that I have not only the development experience, but deployment and client

supporting experience as well.

-> I summarize my object oriented design experience and Hands-On experience.

Several recruiters and interviewers want to make sure that you have a hands on experience, not just

theoretical knowledge, and they want to ensure that you haven't lost touch with your hands on work.

-> Then I tell them that I have worked closely with the architects and also the product owners in implementing

behavior driven development BDD.

-> So depending on the project and the methodologies you have been following, you can mention that here,

again, depending on whether you are a junior consultant, senior consultant, you can tweak these points

and feel free to remove whatever you want to from within this template and add to it as required.

-> Here I mention I summarize my Devops and AWS experience, and I also tell them that I have experience leading complex

projects as well.

-> I summarize this portion with my uml, design pattern experience following that, you can

talk about your education qualification and all the certifications you how should be on the very first

page of your resume.

-> This table here on the second page is optional, but I like having it here because several recruiters

will do a quick control F and search for a particular skill because there are not technical, they are

not expected to be technical.

-> So they do search for skills like Spring, Springboot, etc. So it will help them.

-> If you have a table like this, they can quickly look at this table to see if you have all the tools

and requirements they have on hand and then you can move on to the project

experience where you can mention

the client the location, then the role you have played there, the project name if you want to,

and summarize the environment, which is very important.

-> It tells them that the environment you are currently working on and they can see if that fits into that

environment they are in right now.

-> So this is where talking to the recruiter, going through the job description will help you.

And you should highlight all the technologies that are expected from within the job description or what

the recruiter sees within your environment and within your roles and responsibilities.

-> Even when you talk about your overall experience right here, ensure that whatever is the position you

are working with right now, you highlight all those requirements in these responsibilities here.

-> And also, when you go talk about your latest project, do mention about all the important tools,

you know, and what they are expecting.

-> Under the roles and responsibilities, I start by telling them that I have designed and developed.

rest ful API, which is required for pretty much every microservice today using the spring framework

tools.

-> I also tell them about my frontend experience with Angular and react.

-> And as a senior consultant, you are expected not only to just implement and develop an application,

but also work with non-functional requirements.

-> So I showcase all that here how i configured hibernate fetch and plus strategies to improve the performance of

my application by configuring, caching, fine-tuning, the various layers of the application and so

on.

-> In this point, I summarize the unit testing and integration testing tools that have worked with which

they definitely will look for.

-> I also tell them that i have dockerize the application and deployed to aws using various components.

And my kubernative's experience, is also summarized in the last point.

-> So depending on the Cloud you have worked with depending on the role you are applying for, you can pick all

these points.

-> You can then quickly summarize all your earlier projects.

I like leaving my roles and responsibilities for one more project, but that is not required.

You can just tell them the project.

-> You have worked for the location and quickly move on to the other projects as well so that it tells

them that you have worked across domains and the overall experience will be portrayed.

So feel free to use this template, tweak it as you need it.

CORE JAVA

1. What are the important components of java

-> What are the various important components that are involved in compiling and running a Java program?

-> When we start working on java applications, the very first software that we download and install is

a JDK, which stands for Java Development Kit.

-> The JDK is comprised of JRE and it also comes with several comments or programs such as the Java compiler,

the Java command that we use to run a Java program and more.

-> We download the JDK and install it for a particular operating system.

-> There are different versions for different operating systems for the JDK and the JRE, we can

install JRE separately as well, but usually we install the JDK and the JRE.

-> When we compile the Java program, the Java compiler will convert the Java class into bytecode, not

the machine code, which the underlying operating system understands, but the byte code, which is

platform independent, the bytecode is only understood by the JVM.

-> The Java Virtual machine, which we launch using the Java Command.

-> When we run our Java class file that gets generated during compilation with Java Command, it will launch

a JVM that can interpret the bytecode to the underlying operating system that makes Java platform

independent.

-> A JVM is an instance of the JRE.

It is a representation of the Java runtime environment.

So the platform independency comes because we can compile a Java program on a Windows operating system

and then we can take the classes and run them on Linux using the JVM on Linux, because the bytecode

is understood by the interpreter or the JVM on Linux, and it can convert it into the machine code,

which Linux understands.

-> JIT, which stands for Just in Time compiler, is another important component, which is a part of the

JVM that will interpret the code to the underlying operating system in a optimized fashion.

-> It knows much more details about the operating system and it converts the byte code little bit at a time

by default, if we don't use the JIT interpreter or compiler, the JVM will translate the entire

bytecode into machine code and it will execute against the operating system.

-> Whereas if we use the JIT, it converts a little bit of bytecode at a time

and then it can do certain

optimizations by inlining functions, etc., which makes our application perform better.

-> By default, the JIT is enabled when we use a Java command if you want to disable it for debugging reasons

or whatever, you can use the hyphen D option javadot compiler is equal to none.

-> This will disable the JIT.

1. What are constructors

-> What are constructor's how do they differ from other methods of the class, can we invoke one constructor

from another constructor and can we invoke the super class constructor from the child class constructor?

Let's take a look.

-> A constructor is a method that is used to initialize the properties of an object when it is created.

It has the same name as the class name, unlike the other methods in this case, Class Accord , the constructor

is also Accord.

-> But other methods start and stop, have different names of their own, but the constructor name is the

same name as the class name.

-> The constructor is only invoked when instance or the object of that class is created, whereas the other

methods can be invoked as many number of times as we want.

-> To invoke other constructor's in the same class, we used the this method and we can pass in the parameters

just like any other method.

-> And if the class accord extends Class Honda to invoke the super classes constructor the hondas constructor

from within the class constructor, we use the super method and pass in any parameters.

1. abstract class vs interface

-> What is the difference between an abstract class and a interface, if a class has at least one abstract

method, then that class needs to be marked as abstract class, if not the class wont compile any class

that extends abstract class should then provide the implementation for these abstract methods in a

abstract class.

-> If not, that class should also be marked as it abstract class.

Abstract class can have any number of methods with implementation, but as soon as it has one abstract

method, it becomes abstract class.

-> Whereas in interfaces all the methods are abstract.

-> The class that implements the interface should provide the implementation for those abstract methods.

A class can implement any number of interfaces, whereas it can only extend one abstract class.

1. Why is multiple inheritance no supported

-> Is multiple inheritance not supported in Java, multiple inheritance is not supported because if we

inherit from two classes, which have the same method, for example, here we have the rich confused kid class,

which extends father and mother, which have the same method money.

-> Now, when we invoke the money method using the rich, confused child's instance or object, the compiler

will not know which money method it should bind to or which money meant that it should invoke.

That is one reason why multiple inheritance is not supported will get a compile time issue.

-> If we try to do this to make things even more complex, we have the classic diamond problem that we

see in languages like C++.

-> If we have a grandparent class which also has a money method, then the compiler will find it even more

difficult to decide which money method should be invoked when the rich confused states object is used to

invoke the money method.

1. Can a class implement two interfaces with the same method

-> Can a class implement two interfaces with the same exact method signature?

-> Yes, here I have an interface called Car, which has the go method, which is void and also a avoid

stop method.

-> And I also have a driverless interface with sit down and relax method and a void go method, the same

go method we have in the car interface as well.

-> Now the Honda class implements both the car and the driverless.

It has no issues because it has to provide a implementation for the go method.

-> So it overrides the go method and it provides the implementation.

So we don't have the typical Diemen problem, which we see in case of extending classes.

1. What are the Object class methods

-> What are the different methods that every class inherits from the object class, the different methods

that every class inherits from the object class are equals hash code finalize clone and two string.

-> We also have wait, notify, etc. Those cannot be overridden, but we can use those methods in multi

threading.

-> But these methods equals hash code finalize.

Clone and two string can be overridden.

1. What is the Default hashCode implementation

-> If we do not override the hash code method and provide implementation for it in our class, what is

the value that will be returned when we create objects of that class and invoke the hash code method?

-> The default hashcode method in the object class returns a value a number that is nothing but the memory

location of that particular object.

-> So if we do not provide our own Hash code method, the value that will come back is the memory location

of that object.

1. What is the default toString implementation

-> What is the default implementation of the two string method do? If we do not override the two string method

in our classes and if we try to print an object of that class, then we will see a value which will

have the class name, followed by the @ symbol and the hexadecimal representation of the memory location

of that particular object.

-> So that is the default implementation in the object class.

-> The two string method is implemented in such a way that it will always return the class name followed by

@ symbol and then the hexadecimal representation of the objects address. And if we override it, then we will see whatever implementation we provide,

whatever we return back

from the two string method that will be used instead.

1. equals method vs == operator

-> Another important interview question is the differences between the equal two operator the double equals operator

and the equals method when we compare objects.

-> Let's take a look at the double equal to operator by creating two objects of user.

Here we are creating two instances, U1 and U2 of a class called user.

-> And setting the IDs to one and John, both of them have the same I.D. and name, but they have their

own memory locations on the heap.

-> Now, if we use the double equal to operator to compare these two objects, it will return a boolean false because

the double equals compares the object references or the memory locations and not the contents within the object.

-> So you can call it a shallow comparison and not a deep comparison

Now, if you use the equals method U1 dot equals U2 the equals method is available for every

Java class from the object class.

-> Every class implicitly extends the object class and the equals method is derived or inherited from the

object class.

-> The default implementation of the equals method uses the double equals operator.

So it is a shallow comparison by default.

-> So even this one, this statement U1 dot equals U2 will return a false.

-> We have to override the equals method in our classes, the user class, and provide the implementation

to compare the id and name.

-> There is an exception for string's primitives and enums, for the wrapper types strings and enums the

equals method is already overridden to do a deep comparison.

-> For example, if we create two strings with the same contents ABC and ABC. S1 double equals S2 will

return false because we are using a new operator to create a string.

-> In both cases, they both have two different memory locations.

-> This will return a false, but if we use the S1 dot equals S2, it will do a deep comparison because the

equals method is overridden on a string class and it will return a True.

-> Similarly for the wrapper types int I1, equal to one, two, three, integer two is equal to one, two,

three.

-> If we do a I1 double equals I2 it will return a false. But if we do I1 dot equals I2 It will return a true.

-> To summarize, the double equals operator compares or looks for the references or compares the references of

two objects by default, the equals method also does the same.

-> We should override the equals method and do a deep comparison and return boolean value on our own for

the string primitive types and enums.

-> The equals method will do a deep comparison of the contents.

1. final finally and finalize

-> What are the usages or differences between final finally and finalize?

-> Final is a key word that when we mark variables, for example, primitive types, they become constants.

We cannot do this once we define int i is equal to one, two, three as final.

-> The value of I cannot be changed if we use it against an object declaration.

The object reference cannot change later on.

-> We cannot point the object to a different memory location.

When used against a class, the class cannot be inherited and when used against a method, we cannot

override that method in a child class.

-> Finally is a block when we do exception handling, we use it along with try catch, we can also use

finally without a catch directly, with try.

-> Finally gets executed even when there is an exception and also and there is no exception, it gets executed

always.

-> So it's a good place to put all the clean up code where we can clean up all the DB connections.

The socket connections are input output files, streams.

-> Starting Java 7, we have a try with resource block, which, when used, automatically closes

all the resources that are declared inside the resource block.

-> So starting Java 7, if we use the try with resource block, then we don't need the finally.

-> Finally, we have the finalized method, which the JVM calls when the garbage collector is about to be

called, but we should not rely on this to clean up all our resources in the program or application

because we never know when the JVM is going to call the garbage collector.

1. What are generics

-> What are generics and what is type erasure?

-> Prior to Java one point five version, this is how we defined a collection here, have a list into

which I want to store all the employee I.D's., which are integer values.

-> when I do this, it is fine, I'm adding an integer even when I do this, where I'm trying to add

a string accidentally, the name of the employee that is allowed as well, because there is no way we

are mentioning the type.

-> That is where Generics came in starting Java version One point five.

-> Using generics, we can specify or abstract out that type of data that can go into the collection.

Now, if we try to add an integer, no compile time issues, if I try to add a string, the compiler

will catch it right at compile time and throw an error.

-> Type Erasure is the process of the compiler erasing this type for runtime, once it does, it checks the

compiler will erase this type because of compatibility reasons.

-> Because we want the versions, the older version code probably the code written on Java one four or Java one three

to run on the new JVM's

-> So that is the reason for backward compatibility reasons runtime Generics has been compromised or runtime

generics is not present in Java.

-> It is all compile time Generics. So type erasure implements compile time Generics by erasing the type for

runtime so that the older versions of Java code, which is compiled on JDK one four or one three, can still

run on the newer JVMS.

COLLECTIONS

1. What are the different collection Types

-> What are some of the important interfaces and classes in the collections API?

-> We have list set queue blocking queue and map. list allows duplicates and the implementation classes

of a list interface are array list and linked list. set does not allow duplicates.

-> We have hashset linked hashset.

We also have a sorted set that is implemented by tree set, which sorts all the objects or data that

is stored into that data structure.

-> Then we have first in, first out data structure, which is the queue, which is also implemented by

a priority queue.

-> We also have a blocking queue that helps us implement the producer consumer pattern very easily.

We then have the map, which allows us to store data as key value pairs, the most used one is the hash

map.

-> We also have a linked hash map and a tree map, which is a sorted map.

1. ArrayList vs LinkList

-> What is the difference between an array list and a linked list?

-> An array list uses an array for its internal implementation, whereas a linked list has its own custom,

note type with a previous and next pointers, it is like a doubly linked list.

-> Since an array list uses an array

for adding an element randomly at any location, it can get very expensive all the way to big O of n,

because we'll have to shift the current element in the array to insert an element at that location,

whereas for Linked list it is very easy.

-> We simply create a new node and point the previous and next pointers appropriately.

But when it comes to random access, arraylist is super quick because it is backed by an array, we

can access any element by using the index, whereas accessing elements can get expensive in case of

linked list will have to traverse through the doubly linked list.

-> So usually in applications that are very read intensive, array lists are the way to go.

If we have a lot of read operations with a few right operations, then arraylists are the way to go.

If not, you can use Linked lists.

1. Vector vs ArrayList

-> What is the difference between vector and classes like arraylist and linkedlist and hash table versus

hash map?

-> These hash table and vector classes are from previous versions of Java, where all the methods on them are

synchronized, meaning if one thread accesses one of the methods on these objects, no other thread will

be able to access them until the first thread finishes that will degrade the performance of our application.

-> They are thread safe, but they can be very slow. That's the key difference. These classes are not synchronized.

Their methods on these are not automatically synchronized.

-> We have to take care of thread safety as per our application needs.

1. HashMap vs LinkedHashmap

-> What is the difference between a regular hash map and the linked hash map, the linked hash map maintains

the order of elements in which we add them to the hash map, whereas the hash map doesn't do it.

1. How to create a Generic Class

-> How can we define our own generic class to create our own generic class We create a class just like any other class right next to the class name.

We specify a placeholder for the generic type within less than and greater than symbol.

-> This can be any alphabet, but by convention we use T for type once We have that placeholder.

-> We can use that placeholder to define fields, parameters to constructors,

parameters to methods or

even return types of a method

-> And to use this generic class, you create a instance of that generic class.

And when you do that, you specify the type you want to use. This is very specific.

-> The generic type which will be replaced here, that T will be replaced with whatever type you want to use

and you will pass in the appropriate data and use that generic type and you can use the same generic type

with different data types or different types of data.

1. Failfast vs Failsafe Iterators

-> What is the difference between fail fast and fail safe iterators, traditional collection classes like

array lists, array set provide us the implementation of failed fast iterators.

-> That is when we invoke the iterator method on these collections.

-> The iterator that is returned back is a fail fast iterator.

-> If we use that iterator and loop through the elements, try to access those elements and at the same time,

if we try to modify that collection, add, delete, etc on that collection, whether it is in a single

threaded environment or a multithreaded environment, will get a concurrent modification exception fail

fast.

-> Iterators will not allow parallel modification to happen, will get a concurrent modification exception.

But when we use concurrent collection classes like copy on write Array list or copy on write array set,

these classes provide us with the implementation of failsafe iterators.

-> They Return a Failsafe iterator, which will allow parallel modification, whether it is a single threaded environment

or a multithreaded environment, will not have any issues iterating through and modifying

the collection.

-> At the same time, we see no exceptions.Those are called failsafe iterators.

1. Producer Consumer Pattern

-> Which collection class would you recommend to implement the producer consumer pattern?

-> Blocking Queue is the collection class that should be used to implement the producer consumer pattern,

the blocking queue has a put method that can allow the producer to add work to the queue, and this method

will block if there is too much work on the queue and if the consumer has not already consumed a lot

of work.

-> On the other hand, the consumer will use the take method that will block on the queue.

If there is no more work to do, it will wait for the work to come in from the producer.

-> So blocking queue is the collection class that will make it super easy to implement the producer and

consumer pattern.

1. Comparable vs Comparator

-> What is the difference between comparable and comparator interfaces?

-> The comparable interface provides a class the ability to define natural or default, ordering of its

objects, comparable interface is from Java dot lang package.

-> When a class implements this interface, it needs to provide the implementation for the compare to

method that will be used to compare tow objects of that class.

And this methods Logic should be in such a way that it returns a negative value.

-> If object one has to come before object two a positive value object one has to come after object two and zero

if both these objects are same.

-> So we define our own logic and return these values depending on how we want the comparison to happen.

So Comparable provides the natural or the default ordering.

-> And if we want to create any number of custom ordering for how the object should be compare, then we

use the comparator interface, create our own comparator. This comparator interfaces from Java.util package.

-> And when we create a comparator class, we need to provide the implementation for the compare method

from this interface, which takes two objects, compares them.

-> And even here we return negative value if object One has to come before object two positive object

-> Two has to come before object one and zero if both are same.

When we add our objects to collections, that is where these two are very powerful.

-> If you use collections like treeset tree map, etc. and when you add your objects to them.

If you do not provide a comparator, then the default ordering or the natural ordering provided by the

comparable interface.

-> If your class implements the comparable interface and gives a comparitive method, that ordering will be

used.

-> And if you pass in a custom comparator you create, then that comparator will be used and to compare

then compare method inside it will be invoked to figure out how your objects should be sorted.

1. What are concurrent collections

-> What are concurrent collections when we use collection classes like array list. hash set, hash map in

a multithreaded environment, when the first thread gets a lock on one of these objects, no other thread

will be able to access the entire object unless the lock is released.

-> Also, these collection classes implement fail fast iterators.

-> That is when we use one of the iterators written by these classes to iterate through the elements and

also parallel try to modify that collection will get a concurrent modification exception.

-> That is where the concurrent collection classes like Copy on write array list, copy on write array set come in.

And these classes allow multiple threads to access the same collection and modify it at the same time

by creating a copy of that collection. And they will sync up the copy with the original collection later on.

-> And these collection classes, the concurrent collection classes will implement failsafe iterators, that is if we get

that iterator from one of these collections.

-> Iterate through the elements, we can also modify the collection at the same time, all those problems

with the traditional collection glasses are gone in case off a map we have the concurrent hash map,

which allows fine grained locking instead of locking the entire map.

-> when there are multiple threads the locking happens at a very fine grained or a bucket level as required.

Multi Threading

1. How to create threads

-> What are the different ways in which a thread can be created?

-> There are two different ways in which we can create a thread.

-> The first is where our class will extend the thread class and override the run method.

The threading logic will go inside in this run method.

-> We then create a instance of our thread class and invoke the start method on it.

The second way is to implement the Runnable interface instead of extending the thread class.

-> This gives us the capability of extending any other Java class if required in case of extending a thread

class.

-> We cannot extend any other class if required in our application because Java doesn't support multi class

inheritance or multiple inheritance.

-> So in case of implementing a Runnable interface, we will override or provide the implementation for

the run method from this Runnable interface. That is where the logic leaves.

-> Then you create an instance of this class and pass that instance to the

thread instance or object.

You create instance of the thread.Pass that class instance to the thread instance invoke thread dot start.

1. What is Synchronization

-> What is thread synchronization when we have multiple threads accessing the same object or data?

-> There is a chance that these threads might corrupt each other's data or objects.

That is where synchronization comes in.

-> And once we mark our methods using the synchronized keyword, the first thread that enters the

method will attain a lock on that method, as well as all the synchronized methods in that thread class so

that no other thread will be able to access those methods until the first thread finishes its job.

-> We can also have synchronized key word on static methods and at a block level as well

1. What are class level locks

-> What is a class level lock, every class in Java has a unique lock associated with it.

-> When a thread executes a synchronized static method, it will attain that class level lock.

-> All the other threads will have to wait to access this static, synchronized method and any other synchronized

static methods on that class.

1. What are synchronized blocks

-> What are synchronized blocks?

-> When we use synchronized key word on a method, the entire method will be locked and can be used only

by one thread at a time.

-> If we know exactly which lines of code within that method needs locking, instead of locking the entire method

we can use a Synchronized Block instead.

-> And when we use the synchronized block, only those few lines of code within the block will be locked.

-> And when we do that, we can pass in an object or a class on which we want to attain the lock as well.

1. How do threads Communicate

-> How do threads communicate with each other?

-> Threads communicate with each other using wait, notify and notify all methods when a thread invokes a wait

method, it gives up the lock on a particular object and an another thread can take over that lock and do

its work.

-> Once the other thread finishes its work, it will use the notify method to notify the other thread

which gave up the lock that it has finished its work and it can continue doing its job.

-> It can also use notify all method if there are multiple threads waiting for that lock and they gave

up the lock for this thread, it can use the notify all method which will notify all those threads to

use this wait, notify and notify all methods the threads should be in a synchronized context ,either in a

synchronized block or a synchronized method.

-> Otherwise, a illegal monitor state exception will be thrown.

Java 8

1. Features

-> What are some of the java 8 features you have used?

-> I have used lambda expressions, functional interfaces default methods, predicates functions and streaming

API.

1. What is a Lambda

-> What are lambda expressions and why use them?

-> Lambda expressions bring the functional programming syntax to Java, they are like anonymous functions

or closure's where we don't have to use a function name, a return type or even access modifiers like

public, private, etc. What typically needs a lot of lines of code can be easily achieved through a

lambda expression.

-> Lambda expression starts with parentheses, followed by a arrow symbol hyphen and greater than symbol,

followed by the body of the lambda code.We want to execute within the lambda expression.

-> We can also pass parameters to the lambda function, just like how we pass parameters to a function.

-> The advantages of using lambda functions are very few with very few lines of code.

We can achieve a lot of things very easy to implement.

-> Anonymous inner classes wherever we want, anonymous inner classes.

We can create them on the fly using lambda expressions and they can be even passed as parameters to

other functions.

1. What are Functional Interface

-> What are functional interfaces if an interface has only one abstract method, then that interface is

called functional interface.

-> Examples of inbuilt functional interfaces are Runnable, which has only the run method comparator, which

has only one abstract method, which is compare to.

-> A functional interface can have any number of default methods that were introduced in Java eight, but

it can have only one abstract method.

-> We can mark a functional interface using @ functional interface annotation once we use this annotation

on an interface, if we try to add more than one abstract method to the interface, we'll see compile

time errors.

-> Functional interfaces can be expressed as lambda expressions.

That is a rule that the interface should follow to be expressed as a lambda expression.

It has to be a functional interface.

-> If we want to express that interface as a lambda expression, it should have only one abstract method.

1. What is the Use Lambda

-> Can you explain us with an example on how you have used lambdas to simplify coding applications?

-> One of the examples is when we spun off multiple threads within our application, we typically create

a class that implements the runnable interface and provide a implementation for the run method within

which the multithreaded logic will live, will then create an instance of this class when pass it to

the threads constructor are we.And we invoke thread dot start.

-> It involves so much coding just to get a thread working.

Instead, once you start using lambda expressions, it will become so simple.

-> You will say Runnable are is equal to will start the lambda syntax the run method within the Runnable

interface does not take any parameters.So we start with empty brackets.

-> Then the arrow hyphen greater than symbol than the body of the

Function expression, lambda expression where the logic, formality threading will live, we can then simply

pass this runnable instance to a thread and use thread dot start just like before.

So all this creation of class will go away with lambda expression.

1. What is a Predicate

-> What is a predicate, a predicate is a function with a single argument that returns a boolean, true

or false back. To create a predicate, we use the functional interface predicate that takes a generic

type, and it has a single method called test that takes the generic type and returns a boolean value

back.

-> Since it is a functional interface, we can express it using lambda expressions.

Here is one example, we use the predicate passing the generic type integer, and on the right side

we use a lambda expression to express this functional interface.

-> The implementation is very simple.

I is the parameter to the lambda expression or the function, and the logic here is I is greater

than 20.

-> So that will return

True, if I is greater than 20, but is less than 20 or equal to 20, it will return false.

-> And here we invoke p dot test on that predicate.

We invoke the test method.

-> This here will be used as the logic, as the implementation for the test method.

We get the appropriate result back predicate can work with different data types.

-> Here is an example of string predicate, which checks if the string length is greater than five, always a

predicate returns a boolean value within the predicate.

-> We can do whatever logic we want with the input, but it has to return a boolean value back.

1. What are Predicate Joins

-> What are Predicate Joins?

-> Predicate joining allows us to use more than one predicate together, and we can also negate the result of

a predicate here, have an example where a function takes a predicate as a parameter and a integer

array, and it applies this predicate on each of the array elements and it will print the array value

only if the predicate returns True.

-> And up top here in the main method, we have an array defined and two predicates, one that checks if

a number is greater than 10 and another one that checks if a number is even if we simply apply the first

predicate which checks, if the number is greater than 10, then we will get all the numbers in this

array which are greater than 10 on the console.No surprises.

-> And when we apply the second predicate, we'll get all the even numbers.

No surprises there.

-> But if we want to reverse this first predicate, if we want to get all the numbers that are less then 10

you can simply use P1 dot negate and we invoke that method.

It will reverse the predicates logic or it will use false instead of true however you look at it.

-> So to the method here below, you will pass P one dot negate which will send

you back the opposite results.

-> Instead of greater than 10, you will see less than 10 on the console.

And if you want to apply both these predicates on the array, then you can use P1 and P2.

-> That is, it will return all those numbers or it will display all those numbers on the console, which

are both greater than ten and which are even if you use the end method on the predicate to accomplish

that.

-> And finally, if you want or you simply use P1 or instead of P1 and so greater than 10 or even either

of those.

1. What is a Function

-> What is the function? Function is just like a predicate, except for it can return any type of value,

not just a boolean value we can pass in what type of data it will accept and the return type as well.

-> And we can express it as a lambda expression because it has only one method called apply it is a functional

interface with a method called apply.

-> Whatever logic we implement using the lambda expression will be used and that value will be returned

back instead of a boolean value.

1. What are Default methods on interfaces

-> What are default methods in interfaces and why do we need them?

-> When we create an interface and that interface is implemented by several classes in the application,

all those classes will have to provide the implementations for the abstract method in that interface.

-> Which is fine, but in the future, if we change this interface and add more abstract methods, all

those classes will have to provide a default implementation for those new abstract methods that are

added to the interface.Otherwise they will not compile.

-> All of those classes should be marked as abstract.

That is where the default method that was introduced in Java eight comes in inside an interface.

-> When we define the new method, instead of making it to abstract method, we will define it as a default

method.

-> Default methods can have implementations.

-> It is where we'll put the default implementation so that the other the classes that implement this

interface, if they want to, they can override this method.Otherwise, they will not.

-> But the code will not be broken, all those classes will still compile because we have the default

implementation in the interface itself.So default interfaces were introduced.

-> Do not break the classes in the application when the interfaces change or a new abstract methods get

added to the interface.

-> And if the other classes do not want to use those new methods.

They can go with the default behavior.

-> The classes, which aren't all right, can always provide their own implementation for these default

methods.

1. Can a class implement two interfaces with the same default method

-> A class implement two interfaces which have the same exact default method signature.

-> It is a conditional yes here I have two interfaces interface a an interface x which have the same exact default

method default void

-> m1 now if Class B implements those two interfaces.

Let's see what happens.

-> Class B implements A comma X, we see a compilation problem immediately.

And the error is duplicate default methods named M1 with no parameters are inherited from type X and A.

-> We go towards the classical diamond problem to avoid this will have to provide a over ridden version of this

default method in this class B. Click here say over ride, the default method and do provide your own

implementation here and the error is gone.

-> So you can't just implement two interfaces and then stay calm.

If you have the same default method in both of them, you'll have to provide a over ridden version of that

default method.

1. How to use Stream Filter

-> How can you filter out the even numbers in a given list using streams?

-> It is super simple to do that we take the given list, invoke the stream method on it, the first step

to use streams is configuration.

-> In this case, since we want to filter out, we will use the filter method in the configuration step

pass it a predicate in this case.

-> The predicate is to check for even if it is even, it will return true.

If not, it will return false.

-> So the filter will filter out all the even numbers from the list and in the processing step we will

collect all of them using that dot collect method and to the collect method we pass in a collectors

dot to list that will give us back a new list that will have all the filtered elements which are even.

1. Other Methods on Stream

-> Have you used any other methods on the stream other than just the filter method?

-> Yes, I have used filter dot count that will give us the count of the filter objects instead of collecting

them to another list.

-> We can simply get the count.

-> We can sort the given stream using the Sorted method.

-> We can pass in a comparator to that sorted method which it will use.

-> We can easily create a competitor using a lambda expression and we pass that comparator to the sorted

method in the configuration step and in the processing step we will collect the sorted list into a new

list.

-> I have also used the max which will return the maximum element and we give it the comparator.

-> It will compare and it will return back the maximum element in a given list.

Similarly, the minimum. Will give me the minimum, the smallest element in a given list.

1. Map vs Filter

-> What is the difference between the filter and map method while working with streams?

-> Well, the filter method simply filters out based on a predicate we pass in the map method takes the

given list and it will convert it into a list with different content all together.

-> It will act on the given content and it will transform them based on the logic we pass into the map

method using a lambda expression.

Java 9

1. **What are private methods in interfaces**

-> Can we define private methods in interfaces?

-> Yes, starting Java 9 we can have private methods on interfaces, the reason or the advantage of

it is to reuse the code across the default methods and the static methods we can have an interfaces.

-> If we want to reuse some code across the default methods that we define, we

define a regular private

method, not a static one, and then invoke it inside the default methods.

-> But if you want to reuse the code across static methods, default methods, etc, then this private

method also needs to be static.

-> Only then they will be able to use it in other static methods on the interface.

1. **What are Immutable Collections**

-> How can we create un modifiable or immutable collections in Java. Prior to Java nine

We have to use the collections utility class and the methods available on that class.

-> But starting Java nine, we have the off method on every collection like list and set.

-> Have the off method which will give us back a immutable or un modifiable collection

when this method is invoked.

1. **Stream API Updates**

-> Three new methods are added to the streaming API on the stream, we can now use the take while method

which will take the elements from the stream as long as the predicate that is passed into the take while

method returns.

-> True, it will stop as soon as the predicate fails or returns false.

-> It will not take the rest of the elements from the stream. Drop

while is the opposite drop while will keep dropping the elements as long as the predicate returns

True, and when it returns false, it will take all those elements.

-> And one more useful method is of nullable on the stream which we can use to ignore the null values while

using methods like flat map.

-> So take while drop while and of nullable are the three enhancements that were done to the streaming API in Java nine.

1. **Enhancements to try with resource**

-> Private the Resource Block was a feature that was introduced back in Java six, if we define any resource within

that try block, automatically that resource will be closed, provided that resource should implement

the auto closeable interface and override that close method.

-> That resources close method will be automatically invoked for us when we use the try with the resource block

in Java nine, it has been enhanced so that we don't have to define the resource inside the try block

earlier we had to define it like this in here, but we no longer have to do that.

-> We can define or declare the resource itself outside and simply specify the variable name of that resource

in the try block.

Java 10

-> What are some of the changes that happened when it comes to release starting Java 10. Starting Java

10 Oracle promises a six month release before Java 10 up to Java 9

-> We had to wait for a long time in between each release and each release Used to have a lot of features in them.

-> But starting Java 10 will master only a few features at a time, which will be very easy for the developer.

-> Java 10 introduces us two such cool features using var to define variables and also some API updates

in the Collectors' API.

1. **var**

-> What is the use of var that was introduced in Java 10. Var can be used to define inferred types.

-> Once we define a variable using a var the type of that variable will be inferred from the type of data

or value we assign to that variable.

-> No, it is not like JavaScript var in Java.

-> Once we assign a type of value to a variable defined by var we cannot assign a different type to

it later on in the code that is possible in JavaScript but not in Java.

-> And also VAR is not a keyword.

-> It was left out from the keyword list on purpose because we don't want to break someone else's code.

-> If someone had something like this in their code, for example, int var in earlier versions of Java,

we don't want that to break their code.

-> So for backward compatibility reasons var is not a keyword.

-> Hopefully in the future they might add it to the list, but for now it is not a keyword var is of great

use when we define complex connections.

-> So on the left hand side, we don't have to specify the entire collection type

again, making it much more readable and we can also use it in loops, etc.

When we are using complex

collection classes, we cannot use VAR to define a lambda expression.

-> When we express lambda expressions on functional interfaces, we cannot use lambda expressions.

It needs to have an explicit type and also we cannot define fields at a class level using the var keyword.

-> We can use var's inside the lambda if we want to, but not on the lambda expression itself.

1. **Collectors API updates**

-> While using the streaming API when we use the filter methods, etc., and create new collections from

existing collections, how to create a unmodifiable list or a set as a result of the collect

method to do that in Java 10

-> we have the collectors dot to unmodifible

list to a modifiable set to modifiable map, etc. These are the new api methods that are

introduced in Java 10 we can use them and whatever comes back, we cannot change them.

-> It is unmodifiable if we try to change it, we will get a exception as follows.

Will immediately see a unsupported operations, exception on immutable collections, so we get our immutable

collection back when we use these to unmodifiable list etc.

Java 11

1. **Introduction**

-> What are some of the updates that happened in Java 11, Java 11 adds new methods to the string API.

-> It also adds new api s to the file IO packages. it is added is empty on the optional class, which is very helpful.

And with every release, we'll have some depreciations and removals.

1. String API Updates

-> What are the new API methods that are added to the string class in Java 11, the first method is the

is blank method, which we can use to check for blanks within a given string, it will return.

True, the complete string is blank.

-> We can have any number of blanks.It will return True

-> In that case, if the string has some characters, it will return false.

-> Next is the streaming update that happened, string dot lines method will take a string which has

new lines, and it will return as a stream by splitting the string based on the new lines.

-> We can then collect, use the collect method on that stream and create a list out of it.

-> Next is the update when it comes to spaces and unicode spaces, not just the regular spaces, when we

use a string dot trim method, which is in earlier versions of Java, it will only trim the normal

characters, not the unicode spaces.

-> So the Unicode spaces represented as follows backward slash u 2000 it is one way of representing a Unicode space.

-> We can trim and clean up the unicode spaces

Using the strip method that was introduced in Java 11, Strip method will remove both the spaces at

the start of the string and at the end of the string that is leading and trailing.

-> But if you want to be very specific, you have strip leading and strip trailing.

And the last of the methods is the repeat method that will repeat a given string as many number of times

as you want.

1. File API Updates

-> Java 11 also makes it easy to write text or string data to a file, we can use the update that

happened on the files class in Java dot neo dot File package, file start. write.

-> String is a method that takes the path to the file and the content that we want to write to that file

the string content and we'll write it to the file.

-> Similarly, files dot read string will take the path and return as the string data from the files so write

string and read string make it super easy to work with strings while dealing with files.

1. isEmpty method

-> Java 11 also adds a is empty method on the optional class that is very useful when we work with reactive programming

in prior versions of Java, we used to have optional dot is present where we can check if this

optional has some value inside it.

-> But starting Java 11, we can also check if it is empty.

Earlier we had to negate that is present to do this type of check.

-> But starting Java 11 is empty Method will let us know if the optional does not have a value for it doesn't it will return.

True, if not, it will return false.

Java 12

1. String API Updates

-> Java 12 introduces indent and transform methods on the string class, the Indent method can

be used for indenting the string that is adding spaces right at the beginning of the string.

-> It can provide positive value, which will add as many number of spaces as you provide.

And if you do a negative value, if the string has spaces right at the beginning, they will be taken

out.

-> The transform method takes a function and it will apply that function on the given string, the return

type of that function need not be a string.

-> It can return any other data type.So whatever that function returns, that will be the result of the transform.

1. Compact Number Format

What is compact number format that is introduced in Java 12? starting Java 12 the number format class has a method called Get Compact Number instance, which, when used for formatting

numbers, will express them in a shortcut format. thousand, for example, will be represented as one k

So depending on the locale, it will use the appropriate shortcuts.

And we can also pass in style like short, long etc while doing it instead of one m and one k.

If you pass in long, it will express it as one million, one thousand and so on.

1. More Unicode Chars

-> Java 12 add support for several new Unicode characters to represent the characters in the Chinese

language and also the support for representing all those characters on a chess board?

-> Here is an example.If I run this Java code, so these are all the Unicode characters for representing some of the characters

on the chess board like that.

-> You can represent everything on a chess board now using Unicode characters in Java.

1. Collectors API updates

-> Java 12 adds a teeing method to the collectors class, which takes two down streams and merges their

results using the merger we provide.

-> So as we stream through a collection, we can use two down streams.

We invoke the teeing method on the collectors and then you can use two down streams.

-> One downstream here is counting the total number of elements in the collection.

The second down stream is filtering the results and storing them into a collection.

-> Both these results, the counting and this new collection will be stored using this merger class we

provide.

-> So the count will go into this variable and the filter new collection will go into the collection inside this

merger class, that is how that teeing method works on the collectors.

Java 15 Features

1. **What are Sealed Classes**

-> What are sealed classes and interfaces, sealed classes and interfaces is a preview feature that was

introduced in Java 15 using which we can control which classes can implement a particular interface

and which classes can extend a particular class.

-> We do that using the sealed keyword.

-> First, we seal the interface or the class and then we use the permits keyword to specify those classes

which can implement that interface or if it is a class that can extend that class.

1. **Record Enhancements**

Java 15, also enhances the record preview feature, we can use the sealed interfaces and classes

with the record feature as well.

When we define a record, we can implement a sealed interface or extend a sealed class.

Also, we can use custom annotations on the fields that are defined within a record.

When we define a record and define the fields for that record, we can use our own custom annotations

on those fields.

That is also a additional feature that was introduced in Java 15 record is still a preview feature in

Java 15.

Coding Problems

1. Introduction

-> In this section, you will work on some coding problems, when a coding problem is given, it is to

see how Hands-On you are and also to check your logical thinking abilities.So keep it interactive.

-> Instead of directly jumping and trying to write some code, keep it interactive and explain how exactly

you are approaching the problem and how you are coming up with the solution that will tell them how

you can communicate and also how you think.

-> Follow all the standard conventions like naming your classes, methods and variables correctly, make

sure your code is readable and also you can validate the input data for a

particular method throw

the appropriate exceptions.

-> Make sure your code is secured and all that non-functional things should also be followed when you write

the code.

-> There can be multiple ways in solving a problem so you can take one approach and tell them how you have

solved it.

-> And if you can think of any other approach, you can show them that approach as well.

1. Find the Factorial

-> Write a program that will calculate the factorial of a given number, we can do it two ways the non recursive

and the recursive way, let's try both create a new class call.

-> It factorial with that main method finish.

Hit control D to delete that line, start with a static method, public static This guy returns a int

call it factorial and it receives a number of which we need to calculate the factorial.

-> The logic for calculating factorial is starting from one go all the way to the number and multiply each

number with the next number that will give us the factorial to do that.

-> A result start with the result is equal to one starter for loop for a while loop int i is equal to one i is

less than or equal to that given number I plus plus.

-> Within this loop, we will store the result into the result while multiplying the result star, I so

a result will initially be one, and then as the i increments will be multiplying with the result and storing

it in the result, when the next loop comes in or when we go to the next iteration in the loop, the

result will have the next number that will be multiplied with the previous number and that will be stored

in the result all the way to the number and finally will return the result back.

-> So by the time this for loop finishes, will have the factorial inside the reult now invoked, this use sysout

within the sysout use class name factorial dot the factorial method pass in a number five.

-> Run that program. We should get 120 in the result There we go.

-> So that's the non recursive way to do the recursive way.

Public static int recursive factorial int number recursive function always starts with the exit condition.

-> In this case, the exit condition is when the number.

Becomes zero, num is equal to zero.

-> We will stop and return one semicolon right there.

Else we will recursively invoke the same function return.

-> Within brackets, inStar recursive factorial n minus one, or num minus one.

This should be num not n we are calling it num num state recursive factorial num minus one.

-> So initially when it is five, it will come here.

It will check.

-> If it is equal to zero, it is not.So it will go here

-> Return five star recursive factorial of five minus one four next it will become four star recursive

factorial three and so on.

-> And at the end when it comes to zero, one will be return.

-> One will be multiplied with whatever the next number is, two, three and so on.

That's how the recursion works.

-> Instead of regular factorial invoke the recursive factorial function Pass in five run the program, we

should still get the same result.

-> Let me see what the problem is recursive factorial, I have saved it right here, I have an error, num, I forgot a semicolon added.

-> Run we should still see 120, so that's the recursive way of doing it.

Remember, any time you use recursion, there will be a exit condition then the recursive invocation

of the function.

1. **Generate Fibonacci Series**

-> Write a program to display the given number of Fibonacci series numbers. So if I give you fifteen, it should display 15 Fibonacci series numbers.

-> To do that, go to your id, create a new class call it fibonacci series with a main method,

finish double click to maximize control D to the delete that you can create a static method and write all this

logic there or simply put it in the main method.

-> Int Num one is equal to zero comma.Number two is equal to one comma the fibonacci series

Next number. A fibbonacci series starts with zero, then one.

-> And the following number will be the sum of the previous two numbers that is zero and one, when added,

will become the third number, which is one.

-> The following number will be one plus one, which will be two the following number will be one plus two which

is three and so on.

-> So first will start by displaying the first two numbers sysout.

-> We will use the print statement, we don't need println here, num one, plus add a space in between the number

plus num two, this will display the first two numbers in the Fibonacci series, which is zero and

one.

-> Then we'll take the count, the range which we want to go to int count, say, 15.

We want to display 15 Fibonacci series numbers sys out not sysout.

-> We need a loop for loop for int i is equal to zero, i is less than count I plus plus.

And the logic is super simple.

-> First, Will Calculate, the next number in the Fibonacci series using the fib variable is equal to num

one.Plus num two.

-> use a sysout print will not use println and all the fibbonacci series numbers will come on the same line will

add space before each number.

-> So Space plus fib will print the next number once we print it.

Next, we need to switch these numbers.

-> Now, num one is equal to num two.

The next number will be coming into num two.

-> Num two is equal to what FIB holds right now so that the next time the loop comes in, it will add

the previous two numbers.

-> So this should be in the Fibonacci series for a score run, the Java application and see the output

zero one one one plus one two two plus one three three plus two five five plus three eight and so on.

-> That is how we generate Fibonacci series.

1. **Reverse a String**

-> Right, a program to reverse a given string.

Create a new class. Call it string reverser with a main methosd, finish.

Double click to maximize.

-> You can start by telling the interviewer that there are classes like string buffer and string builder

which can take a string.

-> Say, ABC, and they can right away reverse them, there is a method on these classes which we can use

out of the box.

-> That is not what the interviewer is expecting, but what you prove your knowledge.

-> He wants you to just use the string class and do the reverse.

So get rid of this and create a new method.

-> Public static it returns back a string reverse is the method name.

It takes a string input.

-> Start by defining a new string result, which will initially be blank, then we are going to loop through

use a for Loop int I will start at the end of the string is equal to input dot length minus one because

the index starts from zero all the way to the length minus one.

-> You will see that in a second.Why we are doing lenth minus one i is greater than or equal to zero i minus minus.

-> I'm starting at the end of the given input string and coming all the way to the first element.

And within this, I will say the result is equal to result plus input dot character @ character @ method

and given index will give the element at that particular index in the string character@ i will give

me the last element

-> Because we are starting with the last element length minus one.

And as this loop goes through, I will get one element at a time from the back of the string, I keep

adding that to the result.

1. Check if a String is Palindrome

-> Write a program to check if a given string is a palindrome or not, a palindrome is a string, when

reversed will exactly be the same as the original string.

-> For example, step on No pets when we reverse it.

It will still be a step on no pets.

-> Similarly, Malayalam as South Indian language spoken in the state of Kerala when you reverse it will

still be Malayalam.

-> So let's write a program for it.Go to your id.

-> Create a new class call it Palindrome Checker palindrome Checker with a main method.

Let me zoom that in.

-> Create a new method, public static so that it's a utility method.

We are going to check and return a false so boolean

False or true, is palindrome is the method string.It receives the input string.

-> First thing is to validate if the inputs that are coming in are valid, if input double equals null, and also

check for blank.

-> If we want to throw new illegal argument exception here, you can ask the interviewer if you want to throw

an exception or do you want me to return a false if the input is null with the double quotes, you

can say input cannot be null.

-> Hit control one add a written statement.

-> For now, it will return false.

That's OK.

-> We'll change it at the end.

Next, we need to reverse the string.

-> The easiest way to reverse a string is to use the string builder class string builder hit control one

assign statement to a new local variable.

-> You can pass the input to the string builder constructor.

We create a string builder using the string.

-> Next you can invoke string builder dot reverse method that will reverse the string for us to hit control

one as instatement to our local variable. This is the reverse string.

-> Call it reversed, and we need to compare the original input, with a reversed string, we can do all

that in a single statement, return reversed dot to string.

-> We need to invoke to string method to convert the string builder into a string dot equals pass in the original

input.

-> So that comparison will happen. If it is equals, it will return true.

-> If not, it will return false to invoke it right here, use a sys out Sysout.

-> Invoke the method, it is a static method so you can use the class name Palindrome Checker Dot is

palindrome.

-> Let me pass in Malayalam all lowercase.

Right, click run as.

-> Java application, and it says true in the output here, the have missed a couple of checks, you can

add two more additional checks in your code.

-> One, you can convert whatever input comes in to.

All lowercase are all uppercase.

-> Let's see what happens.

We are passing M capital for malayalam.

-> And run the program, it says false because here we are doing a equals comparison instead of equals.

Ignore case.

-> either we do equals ignore case or you convert the entire input into lowercase or uppercase.

-> That method is available on the string classes so we can either use that or here when we do the comparison,

ignore the case.Now it returns True.

1. Find the numbers

-> Given an array of integers, find out if any two elements are numbers within this array, add up to

the given target number.

-> In this case, you can see that 14 and 18 add up to the given target number, which is 32.

-> These elements need not be sequential one after the other they can be anywhere in the array if found,

return a boolean.

-> True, if not found, then return a boolean false.

-> Create a new class, call it target finder.

With a main method.

-> Control D to delete that public static boolean find numbers is the method that takes a integer array

arr , int target number.

-> Follow the coding along with me and then I'll explain the logic to you, I'm going to use a hashset

here as we process the array elements.

-> I'm going to store some of the elements into this hashset he will realize the reason by the end of

this program, assign the statement to a new local variable, call it processed numbers.

-> And this is of type integer use a for loop.

hit controls space, iterate over a array and we will start from index zero, go all the way to that array dot length

and hit control one on the method add a written statement.

-> By default it will return false within this for loop.

-> I'm going to follow a logic where I will subtract the current array element from this target number.

I'll explain you int required number is equal to the target number minus array

Of I, it might look confusing, but once we finish, you will see that it's very simple and easy.

-> So here I'm subtracting the current array element from the target we are looking for.

So in this case here, 32 minus 30, that will yield two now

-> My responsibility is to check if two exists anywhere in this array to save us some time instead of looping

through continuously.

-> I'm going to do the check right now using this hash set.

So here the logic comes in.

-> If the processed numbers dot contains the difference, which is the required number, then we have found

our number return true else put the current array number into the hashset.

-> So here I will say processed numbers dot add arr of I.

So this way, we need not keep looping through as we put the processing numbers here.

-> We will check if the difference is already there in this processed numbers and if it exists, that means

at some point that there was a number in the aray which, when added to the current number, will result

in the target number.

-> Thats it, that's a simple logic.

If not, by the time this for loop finishes, if this return is not executed, we return a boolean.

False.

-> Let's test it.

-> Go to the main method, use a sysout target finder, dot, find numbers, pass in integer array, new int

array within flower brackets, say 20 30 came out of it.

-> So let me pass the target number for, say, 40 is my target number, and then I will go back to the

array here and initialize it 20.

-> For some reason now I go back 20, 30, 10, 20, 50.

-> So here I have five elements and the target number is 40, this 20 and this

20 should add up to 40, so

we should definitely get a result true Run as Java application. There we go.

-> We get a result true.So when this array is passed here, we loop through the array.

We have a hashset.

-> The target number is 40. So we go here 40 minus the first array element, which is 20.

So the difference is 20.

-> We check if the process numbers has a 20 right now, it will have nothing because we have just started.

So it will go here.

-> Put the 20 this 20 inside that processed numbers hashset.

The same will go on with 30 and 10.

-> When it finds this guy here, this 20, it will again do this check process numbers dot contains target minus

this 20 will be 20 when this check happens.

-> We already have a 20 in the processed numbers.

At that point it will return a true change this to something else instead of

40.

-> If you say 80, we have numbers that add up to 80, which is 50 and 30.

So it should still return a boolean.

-> True yes it does, let's take it to a very high number 150 that should fail it should return false back.

You might be asked to return the exact numbers that have added up to the target number.

-> You simply need to take the current number here.

-> When this is true, you have those numbers, you can put it in a list, you can put it in an array and

you can return them back.

-> You have the required number and you have the current array of i those two other numbers that will add

up to the given target number whenever this condition here is true.

-> So you can return those two back or simply display them here.

Let's do a sysout for now before the return statement.

-> Let me cut it paste it before that and display the required number.

And do a sysout of the array of I.

-> Run the program again, not 150, let me take a positive case say 50.

Run it and it shows me 20 and 30 will add up to 50.