

Java Multithreading Interview Questions and Answers

Last Updated : 17 Oct, 2024



Java Multithreading lets developers run multiple tasks at the same time, making apps faster and more responsive. Java is used by over 10 million developers on 15 billion devices, from Big Data apps to everyday gadgets like phones and DTH boxes. Big companies like **Uber**, **Airbnb**, **EA**, **Google**, **Netflix**, and **Amazon** use multithreading to handle many tasks at once and make their apps work better.



This top 20 Java Multithreading interview questions guide is perfect for freshers and professionals. It interview guide covers key topics like **multithreading basics**, **synchronization**, **thread lifecycle**, **thread pooling**, and more to help you to upgrade your Java multithreading skills. Let's dive into the most frequently asked questions with detailed answers!

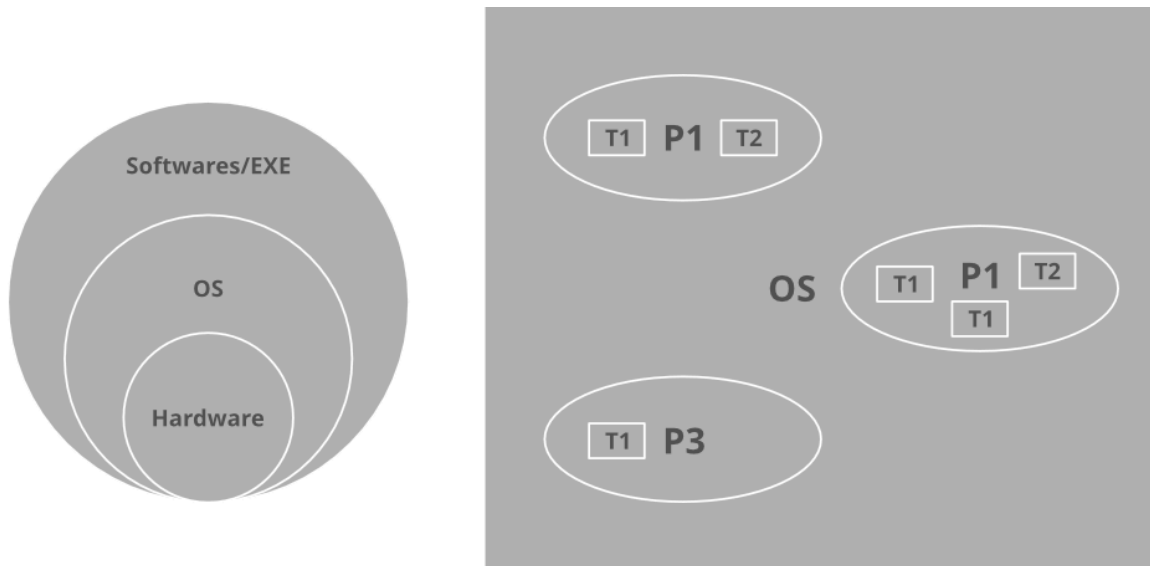
Java Multithreading Interview Questions

1. What is multitasking?

A multitasking operating system is an operating system that gives you the perception of 2 or more tasks/jobs/processes running at the same time. It does this by dividing system resources amongst these tasks/jobs/processes and switching between the tasks/jobs/processes while they are executing over and over again. Usually, the CPU processes only one task at a time but the switching is so fast that it looks like the CPU is executing multiple processes at a time. They can support either preemptive multitasking, where the OS provides time to applications (virtually all modern OS), or cooperative multitasking, where the OS waits for the program to give back control (Windows 3.x, Mac OS 9, and earlier), leading to hangs and crashes. Also known as Timesharing, multitasking is a logical extension of multiprogramming.

Multitasking programming is of two types which are as follows:

1. Process-based Multitasking
2. Thread-based Multitasking



Note: Performing multiple tasks at one time is referred to as *multithreading in java* which is of two types namely *Process-based multithreading* and *Thread based multithreading*.

2 How can you identify the process?

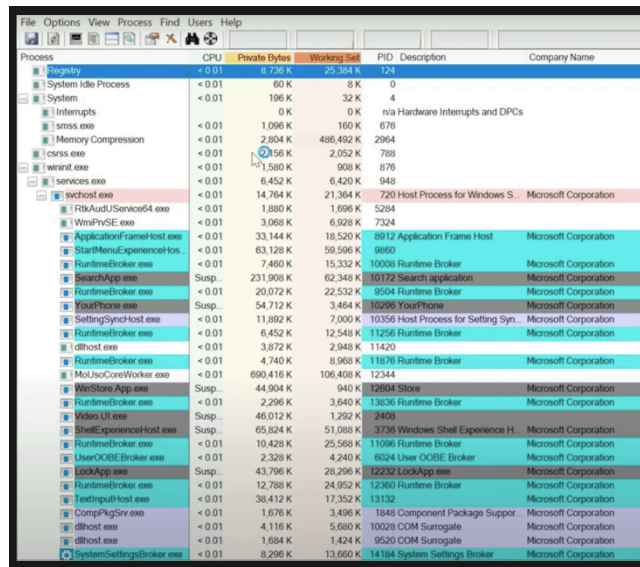
Any program which is in a working state is referred to as a process. These processes do have threads that are single dispatchable units.

3 How do you see a thread?

In order to see threads status let us take windows as an operating system, it illustrates then we'd have ProcessExplorer where you can see GUI shown below for windows operating systems.

This PC > OS > Users > GeeksforGeeks > Downloads > ProcessExplorer

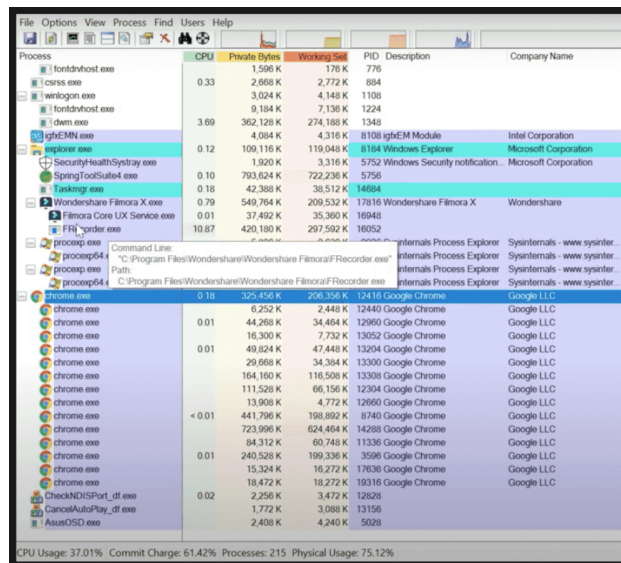
ProcessExplorer is illustrated below in the windows operating systems



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
System Idle Process	< 0.01	60 K	8 K	0		
System	< 0.01	196 K	32 K	4		
smss.exe	< 0.01	1,096 K	0 K	676	ntia Hardware Interrupts and DPCs	
Memory Compression	< 0.01	2,804 K	486,492 K	2964		
csrss.exe	< 0.01	2,556 K	2,052 K	788		
wininit.exe	< 0.01	1,580 K	908 K	876		
services.exe	< 0.01	6,452 K	6,420 K	948		
svchost.exe	< 0.01	14,764 K	21,364 K	720	Host Process for Windows S...	Microsoft Corporation
lsass.exe	< 0.01	1,880 K	1,896 K	5294		
Winlogon.exe	< 0.01	3,096 K	6,928 K	7324		
ApplicationFrameHost.exe	< 0.01	33,144 K	18,520 K	8912	Application Frame Host	Microsoft Corporation
StartMenuExperienceHost.exe	< 0.01	63,128 K	59,596 K	9680		
RuntimeBroker.exe	< 0.01	7,460 K	15,332 K	10008	Runtime Broker	Microsoft Corporation
SearchApp.exe	Susp...	231,908 K	62,348 K	10172	Search application	Microsoft Corporation
RuntimeBroker.exe	< 0.01	20,072 K	22,532 K	9504	Runtime Broker	Microsoft Corporation
YousPhone.exe	Susp...	54,712 K	3,484 K	10296	YousPhone	Microsoft Corporation
SettingSyncHost.exe	< 0.01	11,892 K	7,000 K	10356	Host Process for Setting Syn...	Microsoft Corporation
RuntimeBroker.exe	< 0.01	6,452 K	12,548 K	11256	Runtime Broker	Microsoft Corporation
dlhost.exe	< 0.01	3,872 K	2,948 K	11420		
RuntimeBroker.exe	< 0.01	4,740 K	8,968 K	11676	Runtime Broker	Microsoft Corporation
MsUserCoreWorker.exe	< 0.01	690,416 K	106,408 K	12344		
WinStoreApp.exe	Susp...	44,904 K	940 K	12604	Store	Microsoft Corporation
RuntimeBroker.exe	< 0.01	2,296 K	3,640 K	13636	Runtime Broker	Microsoft Corporation
VideoUI.exe	Susp...	46,012 K	1,292 K	2408		
ShellExperienceHost.exe	Susp...	65,324 K	51,088 K	3736	Windows Shell Experience H...	Microsoft Corporation
RuntimeBroker.exe	< 0.01	10,428 K	25,568 K	11096	Runtime Broker	Microsoft Corporation
UserOOBEBroker.exe	< 0.01	2,328 K	4,240 K	8024	User OOBEBroker	Microsoft Corporation
LockApp.exe	Susp...	43,796 K	28,296 K	12232	LockApp.exe	Microsoft Corporation
RuntimeBroker.exe	< 0.01	12,788 K	24,952 K	12360	Runtime Broker	Microsoft Corporation
TextInputHost.exe	< 0.01	38,412 K	17,352 K	13132		
CompPkgSrv.exe	< 0.01	1,676 K	3,496 K	1848	Component Package Support...	Microsoft Corporation
dlhost.exe	< 0.01	4,116 K	5,680 K	10028	COM Surrogate	Microsoft Corporation
dlhost.exe	< 0.01	1,684 K	1,424 K	8520	COM Surrogate	Microsoft Corporation
SystemSettingsBroker.exe	< 0.01	8,296 K	13,660 K	14184	System Settings Broker	Microsoft Corporation

Note: All of them as listed in the above media are the processes as shown above where at a time many are running in parallel to each other henceforth illustrating multiprocessing in the Jwindows operating system.

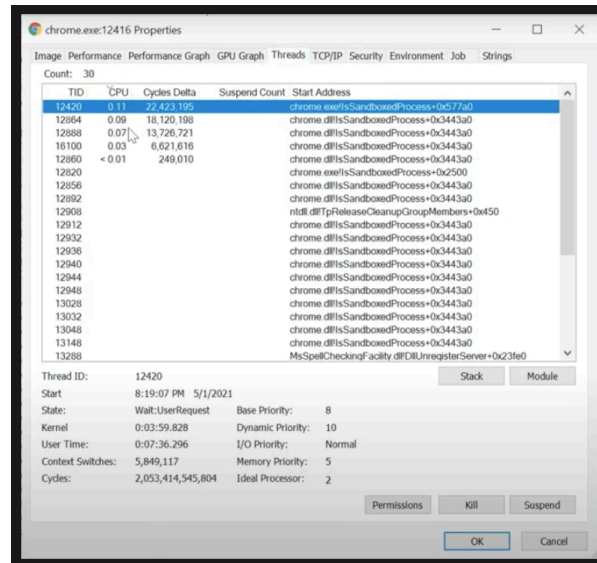
As we have seen threads do reside in a single process so we have to deep dive into a specific process to see them in order to show users how multithreading is going on in the computers at the backend. For example: let us pick a random process from the above media consisting of various processes say it be 'chrome'. Now we need to right-click over the process and click the properties' menu.



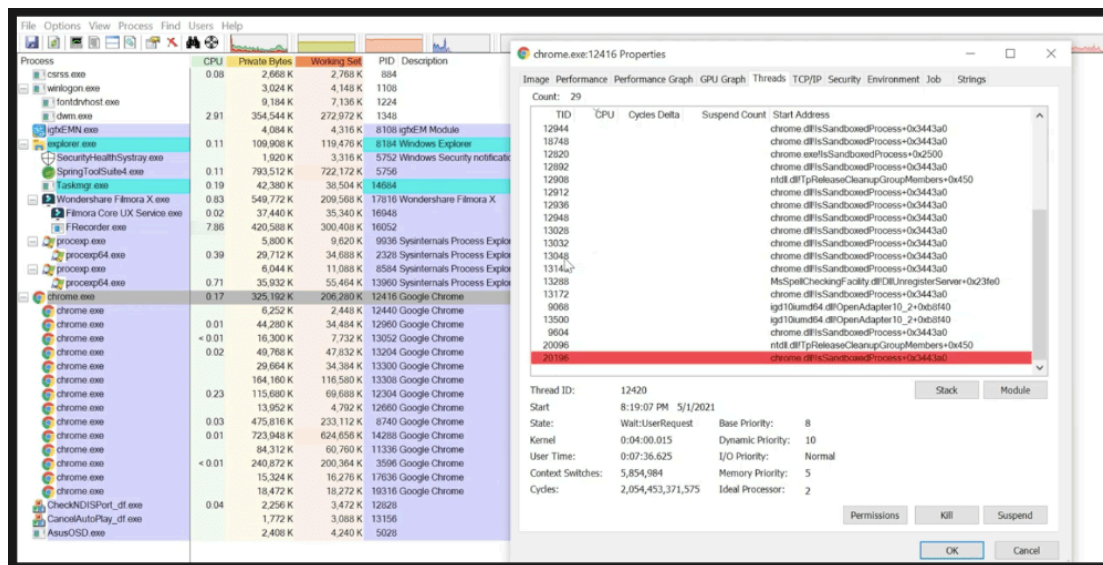
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
fontdrvhost.exe	0.33	1,596 K	176 K	776		
csrss.exe		2,668 K	2,772 K	884		
winlogon.exe		3,024 K	4,148 K	1108		
fontdrvhost.exe		9,184 K	7,136 K	1224		
dm.exe	3.69	362,128 K	274,188 K	1348		
igfxEMN.exe		4,084 K	4,316 K	8108	igfxEMN Module	Intel Corporation
explorer.exe	0.12	109,116 K	119,048 K	8184	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,920 K	3,316 K	5752	Windows Security notification...	Microsoft Corporation
SpringToolSuite4.exe	0.10	793,624 K	722,236 K	5756		
Taskmgr.exe	0.18	42,388 K	38,512 K	14684		
Wondershare Filmora X.exe	0.79	549,764 K	209,532 K	17816	Wondershare Filmora X	Wondershare
Filmora Core UX Service.exe	0.01	37,492 K	35,360 K	16948		
FIL_order.exe	10.87	420,180 K	297,592 K	16052		
proccomp64.exe					Internals Process Explorer	Sysinternals - www.sysinter...
proccomp64.exe					Internals Process Explorer	Sysinternals - www.sysinter...
proccomp64.exe					Internals Process Explorer	Sysinternals - www.sysinter...
chrome.exe	0.16	325,456 K	206,356 K	12416	Google Chrome	Google LLC
chrome.exe		6,252 K	2,448 K	12440	Google Chrome	Google LLC
chrome.exe	0.01	44,288 K	34,464 K	12960	Google Chrome	Google LLC
chrome.exe		16,300 K	7,732 K	13052	Google Chrome	Google LLC
chrome.exe	0.01	49,824 K	47,448 K	13204	Google Chrome	Google LLC
chrome.exe		29,668 K	34,384 K	13300	Google Chrome	Google LLC
chrome.exe		164,160 K	116,508 K	13308	Google Chrome	Google LLC
chrome.exe		111,528 K	66,156 K	13204	Google Chrome	Google LLC
chrome.exe		13,908 K	4,772 K	12660	Google Chrome	Google LLC
chrome.exe		441,796 K	198,892 K	8740	Google Chrome	Google LLC
chrome.exe		723,996 K	624,464 K	14288	Google Chrome	Google LLC
chrome.exe		84,312 K	60,748 K	11336	Google Chrome	Google LLC
chrome.exe	0.01	240,528 K	199,336 K	3596	Google Chrome	Google LLC
chrome.exe		15,324 K	16,272 K	17636	Google Chrome	Google LLC
chrome.exe		18,472 K	18,272 K	19316	Google Chrome	Google LLC
CheckNDISPort.dll.exe	0.02	2,256 K	3,472 K	12828		
CanoeAutoPlay.dll.exe		1,772 K	3,088 K	13156		
AsusOSD.exe		2,408 K	4,240 K	5028		

From the above media, it is clearly perceived that **chrome** is a process and after proceeding with the steps to figure out threads running inside the chrome

process we go to properties of the process 'chrome' below pictorial output will be generated representing threads running in the process chrome.



Note: If we look scroll way from up to down then it will be seeing some colors against a few of those threads. Here green color threads are associated as the newly created threads and red colors associated threads are representing the closed threads.



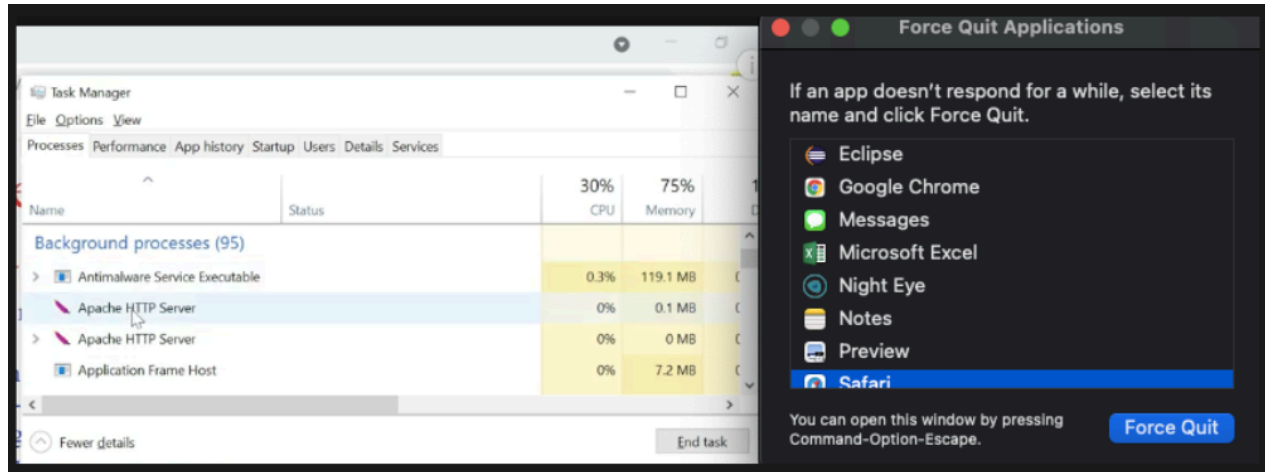
Note: So for chrome to increase the performance by reducing the response time that is referred to as Thread based multitasking.

4 What is Multithreading and How it is Different from Multitasking?

Multithreading is a specialized form of multitasking. **Process-based multitasking** refers to executing several tasks simultaneously where each task

is a separate independent process is Process-based multitasking.

Example: Running Java IDE and running TextEdit at the same time. Process-based multitasking is represented by the below pictorial which is as follows:



Thread-based multitasking refers to executing several tasks simultaneously where each task is a separate independent part of the same program known as a thread. For example, **JUnits** uses threads to run test cases in parallel. Henceforth, process-based multitasking is a bigger scenario handling process where threads handle the details. It is already discussed to deeper depth already with visual aids.

For more details, please refer to [Process-based and Thread-based multitasking in Java](#)



Java Multithreading Interview Questions and Answers

5 Which Kind of Multitasking is Better and Why?

Thread-based multitasking is better as multitasking of threads requires less overhead as compared to process multitasking because processes are heavyweight in turn requiring their own separate address space in memory while threads being very light-weight processes and share the same address space as cooperatively shared by heavyweight processes.

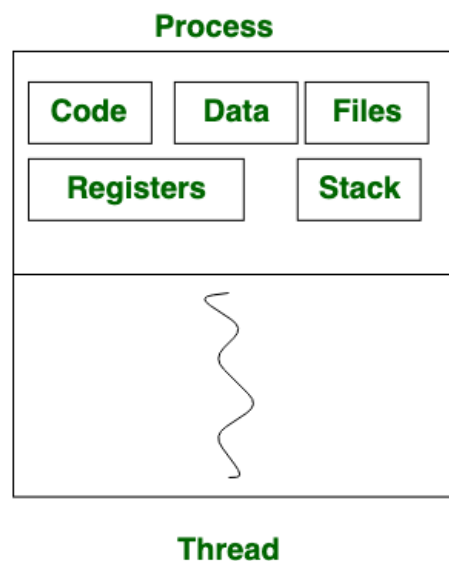
Switching is a secondary reason as inter-process communication is expensive and limited. Context switching from one process to another is cost hefty whereas inter-thread communication is inexpensive and context switching from one thread to another is lower in cost.



Note: However java programs make use of process-based multitasking environments, but this feature is not directly under Java's direct control while multithreading is complete.

6 What is a thread?

Threads are lightweight processes within processes as seen. In java, there are two ways of creating threads namely via Thread class and via Runnable interface.

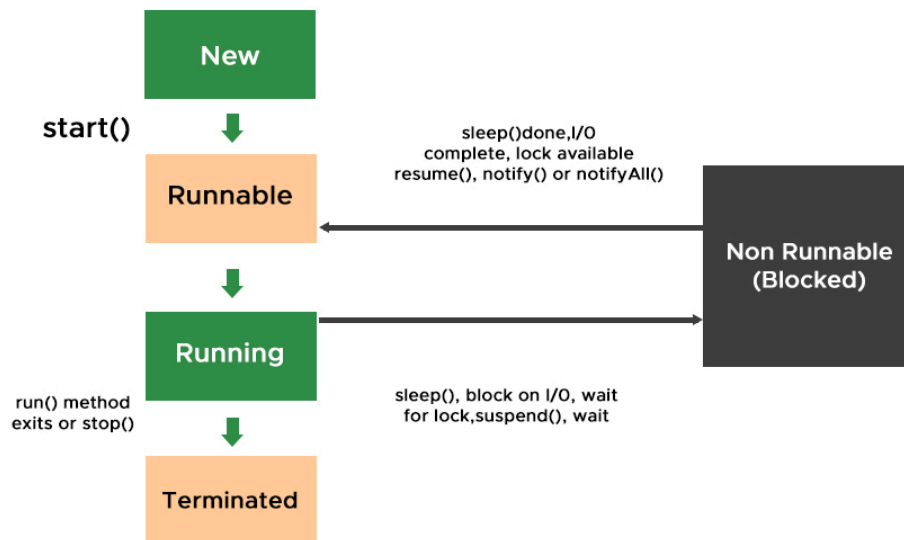


To read more about this, please refer [Thread class in Java](#), [Runnable interface in Java](#)

7 What are the different states of a thread, or what is thread lifecycle?

A thread in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

1. New
2. Runnable
3. Blocked
4. Waiting
5. Timed Waiting
6. Terminated



To read more about this, please refer [Lifecycle and States of a Thread in Java](#)

8 What is the task of the main thread?

All Java programs have at least one thread, known as the **main** thread which is created by JVM at the program start when the `main()` method is invoked with the main thread as depicted from the output perceived from pseudo-code illustration.

Illustration:

```
System.out.println("Mayank Solanki");
```

Output: Mayank Solanki

```
System.out.println(Thread.currentThread().getName());
```

Output: main

9 What are the Different Types of threads in Java?

There are two types of threads in Java as follows:

- User thread
- Daemon thread

User threads are created by java developers for example Main thread. All threads are created inside the main() method are by default non-daemon thread because the 'main' thread is non-daemon. **Daemon thread** is a low-priority thread that runs in the background to perform tasks such as garbage collection, etc. They do not prevent daemon threads from exiting when all user threads finish their execution. JVM terminates itself when all non-daemon threads finish their execution. JVM does not care whether a thread is running or not, if JVM finds a running daemon thread it terminates the thread and after that shutdown itself.



10 How to Create a User thread?

As discussed earlier when the JVM starts it creates a main thread over which the program is run unless an additional thread is not created by the user. The first thing “Main” thread looks for ‘*public static void main(String [] args)*’ method to invoke it as it acts as an entry point to the program. All other threads created in main acts as child threads of the “Main” thread.

User thread can be implemented in two ways listed below:

1. Using Thread class by extending [java.lang.Thread class](#).
2. Using [Runnable Interface](#) by implementing it.

11 How to set the name of the thread?

We can name a thread by using a method been already up there known as [setName\(\)](#) replacing default naming which was 'Thread-0', 'Thread-1', and so

on.

```
thread_class_object.setName("Name_thread_here");
```

12 What is thread priority?

Priorities in threads is a concept where each thread is having a priority which in layman's language one can say every object is having priority here which is represented by numbers ranging from 1 to 10.

- The default priority is set to 5 as excepted.
- Minimum priority is set to 1.
- Maximum priority is set to 10.

Here 3 constants are defined in it namely as follows:

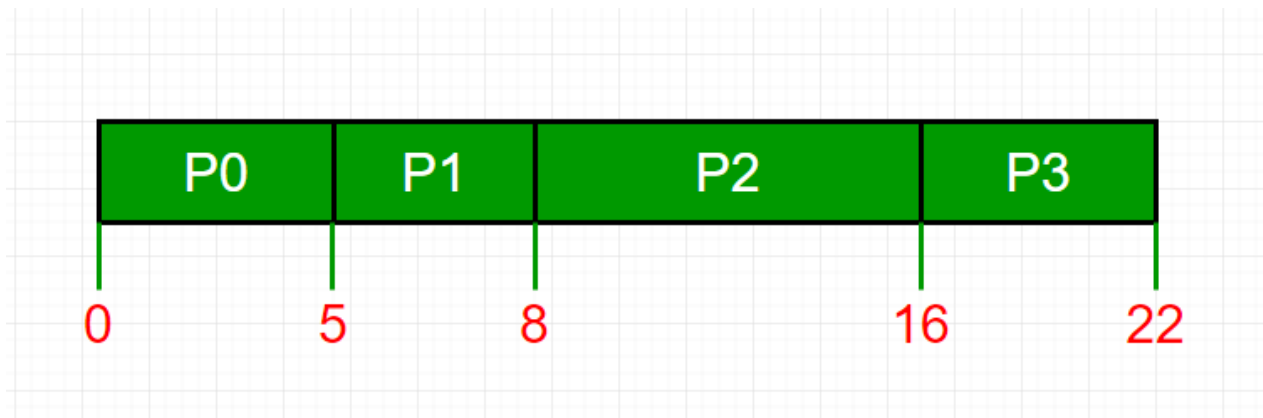
1. public static int NORM_PRIORITY
2. public static int MIN_PRIORITY
3. public static int MAX_PRIORITY

To read more about this, please refer [Thread Priority in Multithreading in Java](#)

13 How deadlock plays a important role in multithreading?

If we do incorporate threads in operating systems one can perceive that the process scheduling algorithms in operating systems are strongly deep-down working on the same concept incorporating thread in **Gantt charts**. A few of the most popular are listed below which wraps up all of them and are used practically in software development.





- First In First Out
- Last In First Out
- Round Robin Scheduling



Now one Imagine the concept of **Deadlock in operating systems with threads** by now how the switching is getting computed over internally if one only has an overview of them.

Figure - 1

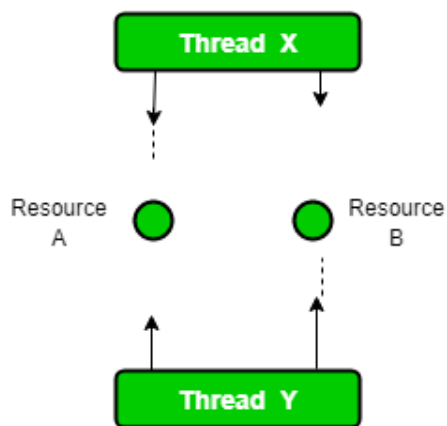
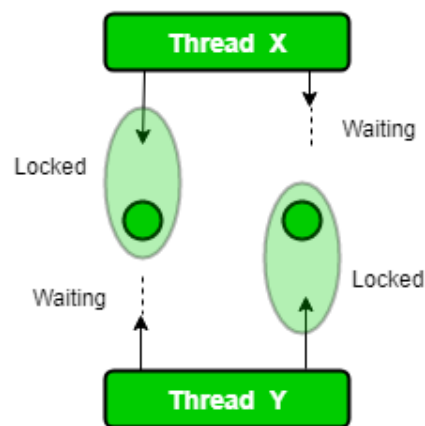


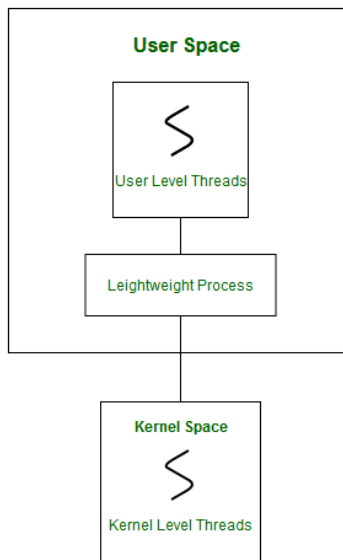
Figure - 2



14 Why output is not ordered?

Scheduling of threads involves two boundary scheduling,

- Scheduling of user-level threads (ULT) to kernel-level threads (KLT) via lightweight process (LWP) by the application developer.
- Scheduling of kernel-level threads by the system scheduler to perform different unique os functions.

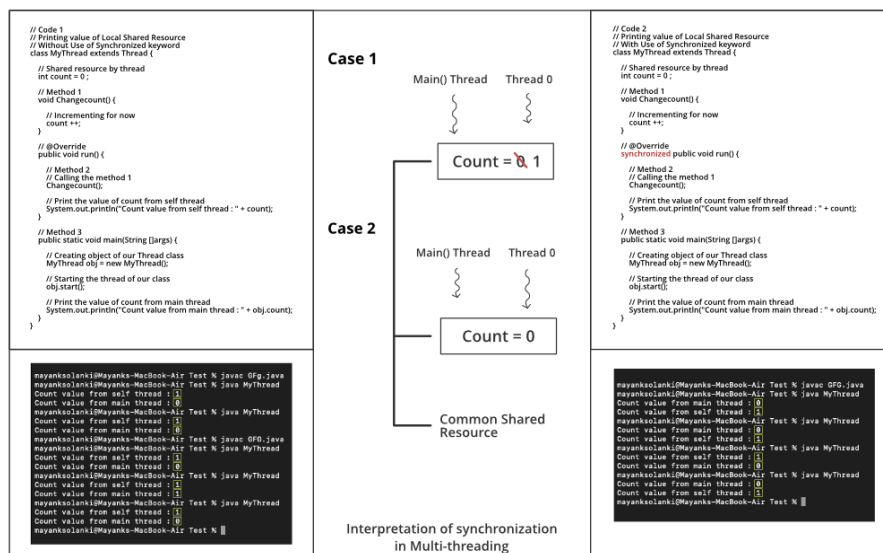


If multiple threads are waiting to execute then thread execution is decided by “ThreadScheduler” which is a part of JVM hence its vendor dependent resulting in unexpected execution of output order.

Note:

- In multithreading, the guarantee of order is very less where we can predict possible outputs but not exactly one.
- Also, note that synchronization when incorporated with multithreading does affect our desired output simply by using the keyword 'synchronized'.

It is as illustrated in the below illustration which is as follows:



15 What is Daemon Thread in Java and explain their properties?

Daemon thread is a low-priority thread that runs in the background to perform tasks such as garbage collection. It does possess certain specific properties as listed below:

- They can not prevent the JVM from exiting when all the user threads finish their execution.
- JVM terminates itself when all user threads finish their execution
- If JVM finds a running daemon thread, it terminates the thread and after that shutdown itself. JVM does not care whether the Daemon thread is running or not.
- It is an utmost low priority thread



Note: The main difference between user thread and daemon thread is that JVM does not wait for daemon thread before exiting while it does wait for the user thread.

To read more about this, please refer to [Daemon Thread in Java](#)

16 How to Make User Thread to Daemon Thread?

It is carried out with the help of two methods listed in 'Thread class' known as **setDaemon()** and **isDaemon()**. First, the setDaemon() method converts user thread to daemon thread and vice-versa. This method can only be called before starting the thread using **start() method** else it is called after starting the thread with will throw **IllegalThreadStateException** After this, isDaemon() method is used which returns a boolean true if the thread is daemon else returns false if it is a non-daemon thread.

17 What are the tasks of the start() method?

The primary task of the **start() method** is to register the thread with the thread scheduler, so one can tell what child thread should perform, when, and how it will be scheduled that is handled by the **thread scheduler**. The secondary task is to call the corresponding run() method of the threads.

18 What is the difference between the start() and run() method?

First, both methods are operated in general over the thread. So if we do use `threadT1.start()` then this method will look for the **run() method** to create a new thread. While in case of `theadT1.run()` method will be executed just likely the normal method by the "Main" thread without the creation of any new thread.

Note: If we do replace `start()` method with `run()` method then the entire program is carried by 'main' thread.



19 Can we Overload run() method? What if we do not override the run() method?

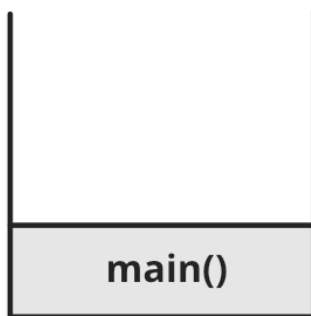


Fig 1.1 : Runtime stack for main thread



Fig 1.2 : Runtime stack for thread named as t

Yes, it is possible to overload `run()` by passing parameters to it and also keeping a check over to comment down `@override` from the `run()` method.

It should be as good as a thread wherein thread we do not have any arguments, so practice to overload is to comment on the call out for overloaded `run()` method. Now, so we need to acknowledge the same whether the output is the same or not if we have not overloaded it.

If we have overloaded the `run()` method, then we will observe that output is always the main method as can be perceived from the stack call from the above image. It is because if we debug the code as provided in the link below we see

as soon as the start() method is called again, run() is called because we have not overridden the run() method.

For more refer to the [Implementation part of how the run\(\) method is overloaded](#)

Conclusion: We can overload the run() method but start() method will call no argument run() only. Hence, it will be of no help to us and is considered as bad practice.



The compiler will simply execute the run() method of the Thread class, keeping a check that the run() method of the Thread class must have an empty implementation. Hence, it results out in no output corresponding to the thread. As we have discussed above already, if we try to do so, then the Thread class run() method will be called and we will never get our desired output.

Note: Geek initially we are requesting to create a thread for us and later the same thread is doing nothing for us which we have created. So it becomes completely meaningless to us by writing unwanted operations to our code fragments. Hence, it becomes useless not to override the run() method.

20 Can we Override the start() method?

Even if we override the start() method in the custom class then no initializations will be carried on by the Thread class for us. The run() method is also not called and even a new thread is also not created.

Note: We can not restart the same thread again as we will get [IllegalThreadStateException](#) from java.lang package. Alongside we can not do this indirectly with usage of 'super.start()' method.

Conclusion

Java Multithreading is an essential skill for any Java developer, providing the ability to create responsive, high-performance applications capable of handling multiple tasks concurrently. Understanding core concepts such as thread creation, synchronization, and inter-thread communication is crucial for tackling complex problems and optimizing resource utilization. By mastering these

interview questions and answers, both freshers and experienced professionals can confidently show their knowledge and expertise in Java multithreading.

[Comment](#)[More info](#) ▼[Advertise with us](#)[Next Article](#) >

Top 100 Data Structure and
Algorithms DSA Interview Questions
Topic-wise



Similar Reads

30 OOPs Interview Questions and Answers [2025 Updated]

Object-oriented programming, or OOPs, is a programming paradigm that implements the concept of objects in the program. It aims to provide an easier solution to real-world problems by implementing real-world entities...

🕒 15 min read

C++ Interview Questions and Answers (2025)

C++ - the must-known and all-time favourite programming language of coders. It is still relevant as it was in the mid-80s. As a general-purpose and object-oriented programming language is extensively employed...

🕒 15+ min read

Top 100 C++ Coding Interview Questions and Answers [2025 Updated]

C++ is one of the most popular languages in the software industry for developing software ranging from operating systems, and DBMS to games. That is why it is also popular to be asked to write C++ Programs in...

🕒 15+ min read

Python Interview Questions and Answers

Python is the most used language in top companies such as Intel, IBM, NASA, Pixar, Netflix, Facebook, JP Morgan Chase, Spotify and many more because of its simplicity and powerful libraries. To crack their Online...

🕒 15+ min read

Java Interview Questions and Answers

Java is one of the most popular programming languages in the world, known for its versatility, portability, and wide range of applications. Java is the most used language in top companies such as Uber, Airbnb, Google,...

🕒 15+ min read

Java Collections Interview Questions and Answers

Java Collection Framework was introduced in JDK 1.2 which contains all the collection classes and interfaces. Java Collection is a framework that provides a mechanism to store and manipulate the collection of objects. It...

🕒 15+ min read

Java Multithreading Interview Questions and Answers

Java Multithreading lets developers run multiple tasks at the same time, making apps faster and more responsive. Java is used by over 10 million developers on 15 billion devices, from Big Data apps to everyday...

🕒 12 min read

Top 100 Data Structure and Algorithms DSA Interview Questions Topic-wise

DSA has been one of the most popular go-to topics for any interview, be it college placements, software developer roles, or any other technical roles for freshers and experienced to land a decent job. If you are...

🕒 3 min read

Top 50 Array Coding Problems for Interviews

Array is one of the most widely used data structure and is frequently asked in coding interviews to the problem solving skills. The following list of 50 array coding problems covers a range of difficulty levels, from...

🕒 2 min read

Most Asked Problems in Data Structures and Algorithms | Beginner DSA Sheet

In this Beginner DSA Sheet for Data Structures and Algorithms, we have curated a selective list of problems for you to solve as a beginner for DSA. After learning the fundamentals of programming, choosing a...

🕒 2 min read



Corporate & Communications Address:

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)



Registered Address:

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305



Advertise with us

Company

Languages

About Us
Legal
Privacy Policy
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Python Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD

Interview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles



System Design Bootcamp

Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects



@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved