

[DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [Topic-wise Practice](#) [C](#) [C++](#)

Top 20 Java Multithreading Interview Questions & Answers

Difficulty Level : Easy • Last Updated : 12 Jan, 2023

[Read](#)[Discuss](#)[Courses](#)[Practice](#)[Video](#)

Java has been rated number one in TIOBE popular programming developers which are used by over 10 Million developers over 15 billion devices supporting Java. It is used for creating applications for trending technologies like Big Data to household devices like Mobiles and DTH Boxes, it is used everywhere in today's information age.



Multithreading in Core Java(J2SE) is a very important topic from an interview point of view. It can lead you to become a **Java Developer, Java Testing Engineer, Java Architect, Lead Analyst, Java Consultant**, and most important a real good java programmer enabling the confidence to dive in J2EE programming that stands for Java to enterprising edition or in layman language making you fit to work in corporate domain workflow directly. Perks wide varied in India for java developers from **200K to 25000K** for as fresher based upon the level of

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

Start Your Coding Journey Now!

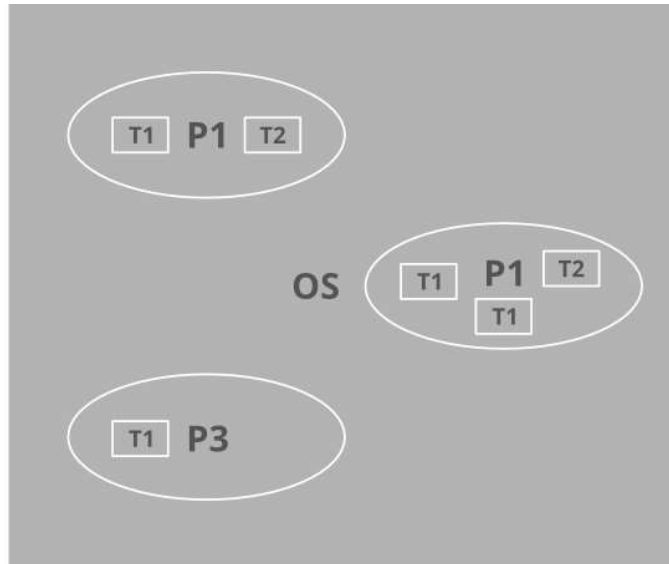
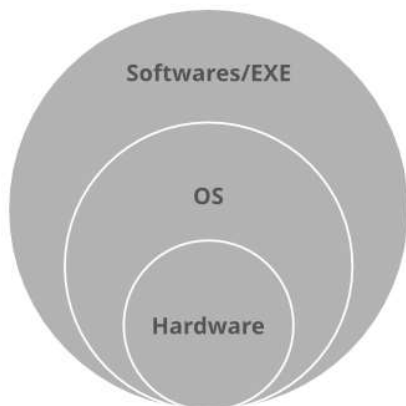
[Login](#)[Register](#)

Q-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.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

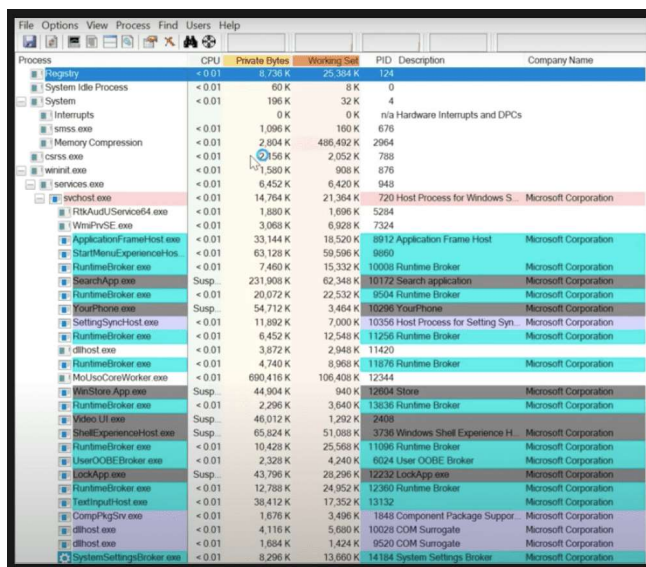
threads that are single dispatchable units.

Q-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



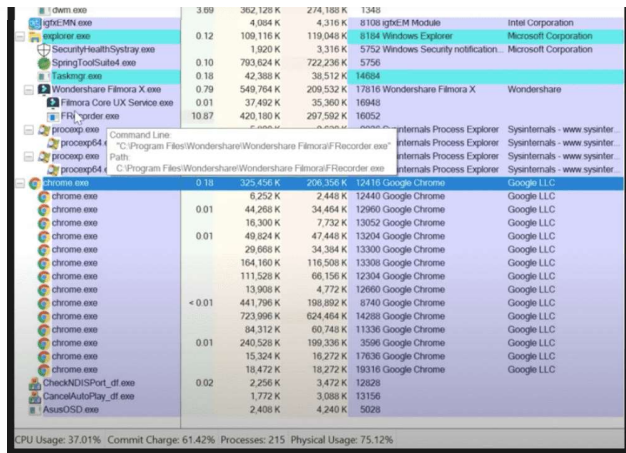
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
System Idle Process	< 0.01	60 K	8 K	0		
System	< 0.01	196 K	32 K	4		
smss.exe	< 0.01	1,096 K	160 K	676		
csrss.exe	< 0.01	2,994 K	486,492 K	2964		
conhost.exe	< 0.01	1,580 K	2,052 K	788		
services.exe	< 0.01	6,452 K	6,420 K	948		
svchost.exe	< 0.01	14,764 K	21,364 K	720	Host Process for Windows S.	Microsoft Corporation
RuntimeBroker.exe	< 0.01	1,880 K	1,696 K	5284		
WmiPsvSE.exe	< 0.01	3,068 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	8960		
RuntimeBroker.exe	< 0.01	7,480 K	15,332 K	10028	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
YourPhone.exe	Susp...	54,712 K	3,464 K	10296	YourPhone	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	11876	Runtime Broker	Microsoft Corporation
MoUsoCoreWorker.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
Video UI.exe	Susp...	46,012 K	1,292 K	2408		
ShellExperienceHost.exe	Susp...	65,824 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
UserCoreBroker.exe	< 0.01	2,328 K	4,240 K	8624	User Core Broker	Microsoft Corporation
LockApp.exe	Susp...	43,796 K	28,296 K	12232	LockApp	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		
CompKpgSvc.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	9520	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.

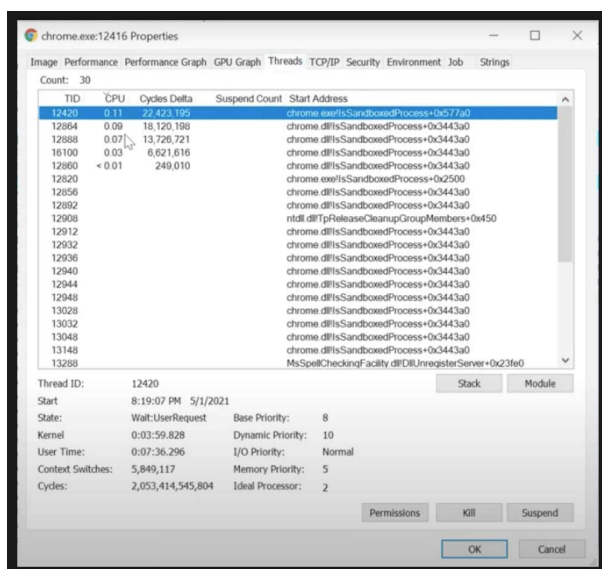
As we have seen threads do reside in a single process so we have to deep dive into a specific process to see them in order to show users how multithreading is going on in the computers at the backend. For example: let us pick a random process from the above media consisting of various processes say it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

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!



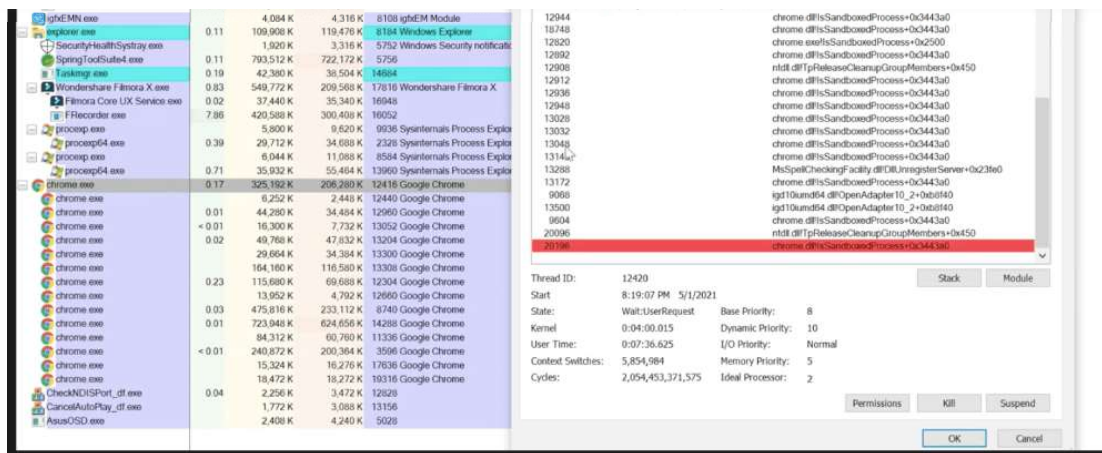
From the above media, it is clearly perceived that **chrome** is a process and after proceeding with the steps to figure out threads running inside the chrome process we go to properties of the process 'chrome' below pictorial output will be generated representing threads running in the process chrome.



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.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

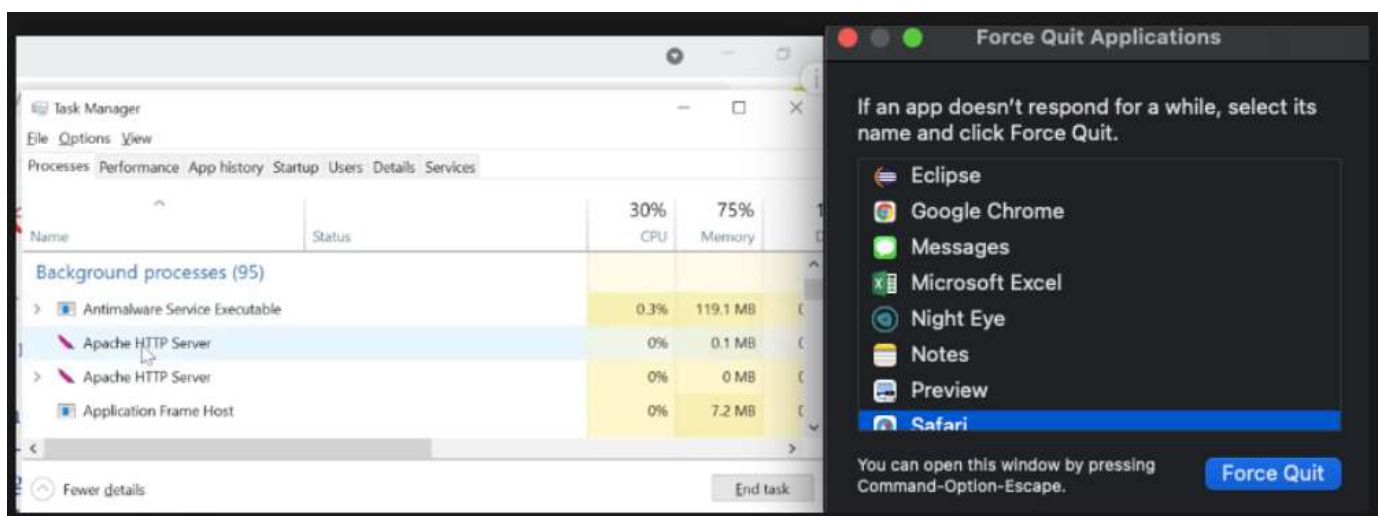


Note: So for chrome to increase the performance by reducing the response time that is referred to as **Thread based multitasking**.

Q-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:



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

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](#)

Q-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.

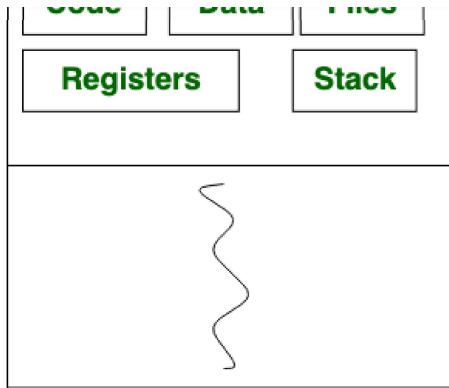
Switching is a secondary reason as inter-process communication is expensive and limited. Context switching from one process to another is cost hefty whereas inter-thread communication is inexpensive and context switching from one thread to another is lower in cost.

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.*

Q-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.

Start Your Coding Journey Now!



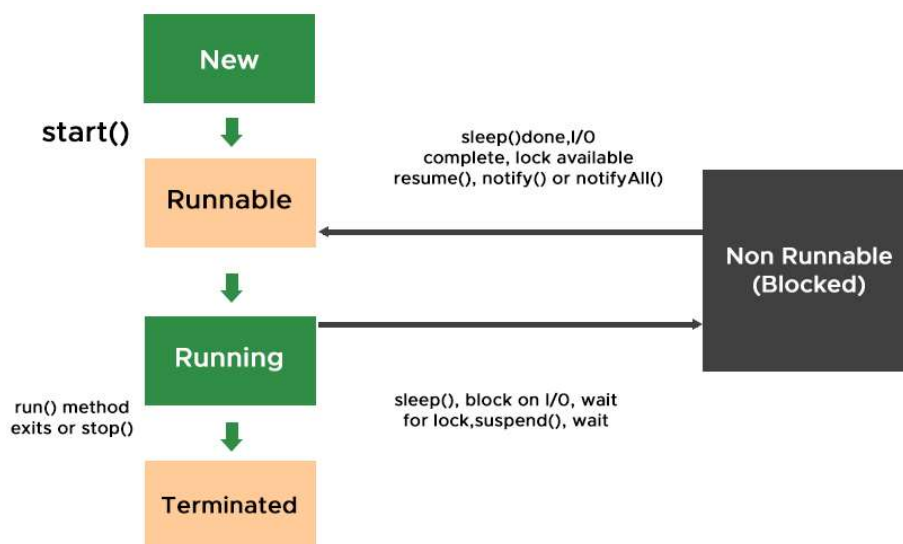
Thread

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

Q-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



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

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.getName().currentthread());
```

Output: main

Q-9 What are 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.

Q-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.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

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

Q-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 0.
- Maximum priority is set to 10.

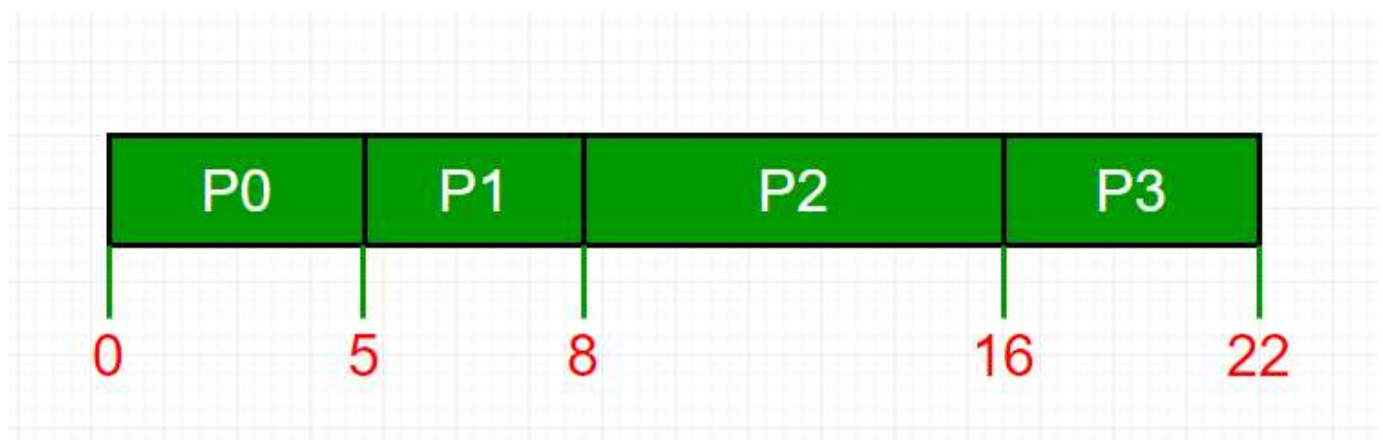
Here 3 constants are defined in it namely as follows:

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](#)

Q-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.



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

- 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

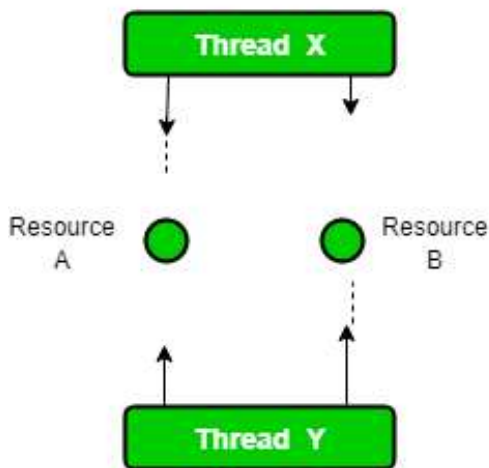
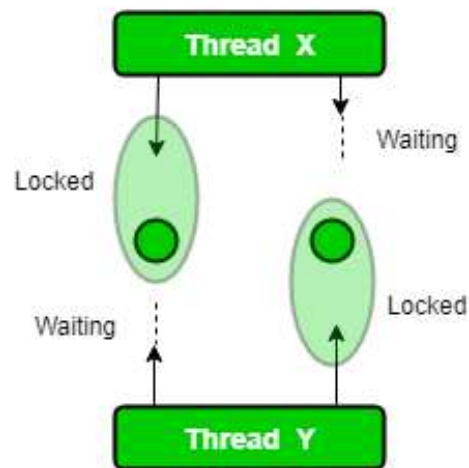


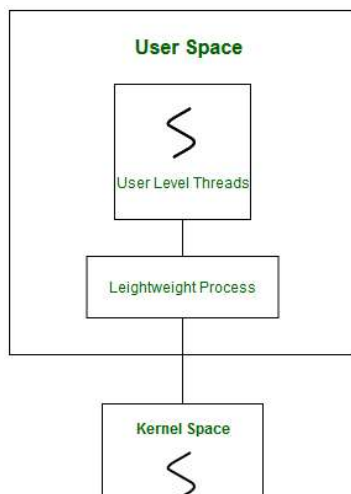
Figure - 2



Q-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.



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

execution of output order.

- *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'.*

[illegible]

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:

- We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Start Your Coding Journey Now!

wait for daemon thread before exiting while it do waits for the user thread.

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

Q-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 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.

Q-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.

Q-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 threadT1.run() method will be executed just like 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.*

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

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!



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.

Start Your Coding Journey Now!

Q-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.

10

Related Articles

1. Top 50 Java Collections Interview Questions and Answers

2. Top 25 Android Interview Questions and Answers For Experienced

3. 30 OOPs Interview Questions and Answers (2023)

4. && operator in Java with Examples

5. Deadlock in Java Multithreading

6. Java Thread Priority in Multithreading

7. Multithreading in Java

8. Java Multithreading Program with Example

9. What does start() function do in multithreading in java?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

[Next](#)

Java Multithreading Tutorial

Article Contributed By :

**solankimayank**

@solankimayank

Vote for difficulty

Current difficulty : [Easy](#)

Easy

Normal

Medium

Hard

Expert

Improved By : [sweetyty](#), [sagar0719kumar](#), [mitalibhola94](#)Article Tags : [Java-Multithreading](#), [Java](#)Practice Tags : [Java](#)[Improve Article](#)[Report Issue](#)A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305feedback@geeksforgeeks.org

Company

[About Us](#)

Languages

[Python](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

[Privacy Policy](#)[Copyright Policy](#)[Third-Party Copyright Notices](#)[Advertise with us](#)

Data Structures

[Array](#)[String](#)[Linked List](#)[Stack](#)[Queue](#)[Tree](#)[Graph](#)

Web Development

[HTML](#)[CSS](#)[JavaScript](#)[Bootstrap](#)[ReactJS](#)[AngularJS](#)[NodeJS](#)

Computer Science

[GATE CS Notes](#)[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)[SQL](#)[R Language](#)[Android Tutorial](#)

Algorithms

[Sorting](#)[Searching](#)[Greedy](#)[Dynamic Programming](#)[Pattern Searching](#)[Recursion](#)[Backtracking](#)

Write & Earn

[Write an Article](#)[Improve an Article](#)[Pick Topics to Write](#)[Write Interview Experience](#)[Internships](#)[Video Internship](#)

Data Science & ML

[Data Science With Python](#)[Data Science For Beginner](#)[Machine Learning Tutorial](#)[Maths For Machine Learning](#)[Pandas Tutorial](#)[NumPy Tutorial](#)[NLP Tutorial](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Start Your Coding Journey Now!

[Company Interview Corner](#)[Experienced Interview](#)[Internship Interview](#)[Competitive Programming](#)[Aptitude](#)[Django Tutorial](#)[Python Projects](#)[Python Tkinter](#)[OpenCV Python Tutorial](#)

GfG School

[CBSE Notes for Class 8](#)[CBSE Notes for Class 9](#)[CBSE Notes for Class 10](#)[CBSE Notes for Class 11](#)[CBSE Notes for Class 12](#)[English Grammar](#)

UPSC/SSC/BANKING

[SSC CGL Syllabus](#)[SBI PO Syllabus](#)[IBPS PO Syllabus](#)[UPSC Ethics Notes](#)[UPSC Economics Notes](#)[UPSC History Notes](#)

@geeksforgeeks , Some rights reserved