Java Interview Question and Answers

➢ Topic – Java Multithreading:

Thread and Multithreading Interview Questions in Java with Answers

---

1. What is Thread in Java?
Ans: A thread is a single independent path of execution of a group of statements. It is used to perform more than one task simultaneously.

2. What is a process in Java?
Ans: A process is a program that executes as a single thread. A thread is a subset (part) of process. One process can contain multiple threads.

3. What is the difference between thread and process in Java?
Ans: The difference between a thread and a process is as follows:

- A process is a program that is executing whereas, a thread is the smallest unit (piece) of executable code within a process.
- A process can have more than one thread.
- A process is considered as heavyweight while a thread is called lightweight.
- Threads are easy to create but processes are not easy to create.
- Threads within a process can communicate directly while processes do not communicate so easily because they need an operating system, files, or network to communicate.
- Processes are heavily dependent on system resources available whereas threads need minimal amounts of resources.
- Every individual process has its own separate memory address space but all threads share the same memory address space.

4. What is a single-threaded program and a multithreaded program?
Ans: When a program contains a single flow of control, it is called single-threaded program.

When a program contains multiple flows of control, it is called multithreaded program.

5. Which thread always runs in a Java program by default?
Ans: Main thread

6. Why threads are lightweight process in Java?
Ans: Threads are also known as lightweight because they can be executed in the same memory space. All the threads in the main application program share the same address space in the memory so that they can easily communicate among themselves.

Thus, they also take less space in memory and less processor time.

7. What is the use of threads in Java?
Ans: Threads can be used for multiple purposes. Some advantages of using threads are as follows:

- Threads are mainly used in server-side programs where we need to handle multiple clients on network or internet simultaneously.
- Another important use of threads is in creating games and animations.
- Generally, threads can be used to perform more than one task simultaneously.

8. What is Multithreading in Java? Why it is needed?
Ans: The process of executing multiple threads simultaneously (concurrently) is called multithreading in Java.

Multithreading is needed because of the following reasons:

a) Multithreading makes the program more responsive and interactive.
b) It enhances the performance of the application.
c) Multithreaded programs can run faster than single-threaded programs in some cases even on single processor system.
d) It makes maximum utilization of CPU and keeping the idle time of CPU to minimum.

9. Explain the concept of context switching of thread.
Ans: The process of switching from one thread to another thread by CPU is called context switching.

10. Explain the time-sharing of thread in Java.

Ans: In a single processor system, multiple threads share CPU time that is known as time-sharing.

11. What are the advantages of Multithreading in Java?

Ans: The advantages of using multithreading programming concept are as follows:

- Multithreading helps to reduce computation time.
- Multithreading technique improves the performance of the application.
- Threads share the same memory address space. Hence, it saves memory.
- Multithreaded program makes maximum utilization of CPU and keeping the idle time of CPU to minimum.
- Context switching from one thread to another thread is less expensive than between processes.
- In a multithreaded application program, different parts of the application are executed by different threads. The entire application does not stop even if an exception occurs in any of the threads. It does not affect other threads during the execution of the application.

12. What are the drawbacks of multithreading in Java?

Ans: The drawbacks of multithreading in Java are as follows:

- Increased complexity.
- Synchronization of shared resources.
- In the multithreading programming concept, debugging is difficult. At times, result is unpredictable.
- Potential deadlocks.
- Programming complications may occur.

13. How can multiple threads run in a single-processor system concurrently (simultaneously)?

Ans: In single-processor systems, multiple threads share the CPU time, known as time-sharing.

14. What is multitasking in Java? What are the two types of multitasking?

Ans: The process of executing one or more tasks concurrently or at the same time is called multitasking. It is the ability of an operating system to execute multiple tasks at once.

There are two types of multitasking or multitasking can be implemented in two ways:

- Process-based multitasking (Multiprocessing)
- Thread-based multitasking (Multithreading)

15. What is the difference between process-based multitasking (multiprocessing) and thread-based multitasking (multithreading)?

Ans: The difference between multiprocessing and multithreading are as follows:

1. The process of executing multiple programs or processes at the same time (concurrently) is called process-based multitasking or multiprocessing.	1. The process of using multiple threads to perform one or more tasks at the same time in a program by the processor is called thread-based multitasking.
2. In process-based multitasking, several programs are executed at the same time by microprocessor.	2. In thread-based multitasking, a program uses multiple threads to perform one or more tasks at the same time by a processor.

16. What are the advantages of thread-based multitasking over process-thread multitasking?

Ans: The main advantages of thread-based multitasking as compared to process-based tasking are:

- Threads share the same memory address space.
- Context switching from one thread to another thread is less expensive than between processes.
- The cost of communication between threads is relatively low.
- Threads are lightweight as compared to processes (heavyweight). They utilize the minimum resources of the system. They take less memory and less processor time.

17. What is the difference between single-tasking and multi-tasking?

Ans: When only one task is executed at a time, it is called single-tasking.

When several tasks are executed at a time, it is called multi tasking.

In single tasking, the CPU time is wasted, but we can utilize the CPU time in an optimum way in multi-tasking.

18. What are the ways to create a new thread in Java?
or How do you implement thread in Java?

Ans: There are two ways to create or implement a new thread in Java. They are:

- **By extending Thread class:** Extend the Thread class and override the run() method.
- **By implementing Runnable interface:** Implement Runnable interface and implement the run() method.

19. When to use Runnable interface vs Thread class in Java? Which one is better?

Ans: Implement Runnable interface is better to use than extends Thread class because Java programming language does not support multiple inheritance through class, but it always supports multiple inheritance through the interface.

20. What is the difference between "extends Thread" and "implements Runnable" in Java?

Ans: extends Thread class and implements Runnable interface both have the same function. But when we extend Thread class, there is no scope to extend another class because Java does not support multiple inheritance through classes.

21. What is a runnable object?

Ans: In Java, each task is an object of the Runnable interface, also known as runnable object.

22. Which method calls the run() method?

Ans: start() method.

23. How a thread is executed in Java?

Ans:

24. What is the difference between start() and run() method of Thread class?

Ans: start() method is used to start a newly created thread. It internally calls run() method for execution of statements inside run() method because there is a difference between calling the run() method directly and though start() method.

When you will call run() method directly, it will be treated as a normal method by JVM and called in the same thread. No new thread will be stared.

25. What are the states of a thread in Java? or what are the thread states in Java?

or **What are the different stages of Thread lifecycle?**

Ans: There are five states of thread in Java. They are:

- **New:** When a thread is started, it goes to New state.
- **Runnable:** When run() method is called, the thread enters into runnable state and thread is ready for execution of a particular task.
- **Running:** When the thread executes a particular task, it is in running state.
- **Wating/Blocking state:** When a thread is made to wait for a certain period of time, it goes to the waiting state.
- **Dead:** When a thread completes the execution of statements inside run() method, it exits from run() method and terminated or dead.

26. What does start() method of Thread class do?

Ans: The Thread class start() method puts the thread into ready state (runnable state) and makes the thread eligible to run. It automatically calls the run() method.

27. What are the various methods of Thread class used to manage threads?

Ans: The important methods provided by Thread class are as follows:

a) **start():** Initiate thread and calls run() method internally.
b) **run():** Task to be performed is declared within a run() method.
c) **sleep():** It makes the running process to wait for a particular period of time.
d) **yield():** Pauses the execution of current thread and allows another thread of equal or higher priority that are waiting to execute.
e) **stop():** Stop the execution of thread permanently.
f) **join():** Make a thread wait for another thread to terminate its process.
g) **isAlive():** Check the thread is alive or not. It returns a boolean value (true or false) that indicates thread is running or not.
h) **setPriority():** Set the priority of thread.

28. What is the difference between sleep() and yield() method?

Ans: When sleep() method is called on a thread, the thread will return to its waiting state. When yield() method is called on a thread, the thread returns to the runnable state (ready state).

29. What is the fundamental difference between sleep() and wait() methods?

Ans: The main difference between sleep() and wait() methods is as follows:

a) The wait() method is an Object class method, whereas sleep() is a static method provided by Thread class.

b) wait() method is called on objects whereas, sleep() method is called on threads, not objects.

c) A waiting thread can be waked up by another thread by calling notify() method on the monitor which is being waited on. But a sleeping thread cannot be woken up.

d) When we call wait() method, the current thread releases the monitor or lock and goes from running state to waiting state can return to runnable state only when notify() or notifyAll() method is invoked on that object.

In the case of sleep(), the current thread does not release the lock. It just sleeps for some pre-defined time period and returns to runnable state when sleep time is up

e) sleep() method does not require any object lock whereas, wait() method needs object lock before it is called.

f) If wait() method is invoked without getting object lock, IllegalMonitorStateException is thrown at runtime. But sleep() method never throws such an exception.

g) wait() method must be called from synchronized block whereas sleep() method can also be called from outside synchronized block.

30. What is thread scheduler in Java?

Ans: Thread scheduler in Java is the component of JVM that determines the execution order of multiple threads on a single processor (CPU). It decides the order in which threads should run. This process is called thread scheduling.

31. What is time-slicing in Java?

Ans: The process of allocating time to threads is known as time slicing in Java. Time-slicing is based on non-priority scheduling.

32. What is thread priority? How do you set a priority for a thread?

Ans: Thread priority in Java is a number assigned to a thread that is used by Thread scheduler to decide which thread should be allowed to execute. It is represented by a number from 1 to 10.

The thread with the highest priority is selected by the scheduler to be executed first.

We set the priority of a thread using the setPriority() method of Thread class. This method accepts an integer value as an argument and sets that value as priority of a thread through which it is called.

33. What is the default priority of a thread in Java?
Ans: The default priority of a thread is 5 (NORM_PRIORITY).

34. What are the different priorities that can be set on a Thread in Java?
Ans: Thread class provides three priorities that can be set on a Thread object in Java. They are as:

a) **MIN_PRIORITY:** It is the minimum priority that can be assigned to a thread. The value of MIN_PRIORITY is 1.
b) **NORM_PRIORITY:** It is the default priority that a thread can have. The default priority of a thread is 5.
c) **MAX_PRIORITY:** This is the maximum priority that is assigned to a thread. The value of MAX_PRIORITY is 10.

35. How to get priority of current thread in Java?
Ans: We can get priority of current thread by using getPriority() method of Thread class. The getPriority() method returns the priority of a thread through which it is called.

36. How to stop a thread in Java?
Ans: Java provided some control methods such as stop(), suspend(), and resume() in JDK 1.0. But these methods deprecated in later releases due to potential deadlock threats.

We know that thread stops automatically as soon as they finish the execution of run() method.

To manually stop, there are two ways through which we can easily stop a thread in java program. They are:

- By using boolean variable.
- By using isInterrupted() method

37. Can a thread is again alive when it goes into the dead state?

Ans: No, once a thread is terminated or moves into dead state, it cannot alive again. If we try to it by calling start() method, we will get an IllegalThreadStateException exception.

38. Can we call run() instead of start() method on a thread in Java?
Ans: Yes. We can call but it will not work as a separate thread. It will just work as a normal object in main thread. So, there will not be context switching between threads.

39. Is it possible to start a thread two times in Java?
Ans: No. There is no possibility to start a thread twice because we can call start() method only once on a thread in Java. If we call it twice, it will throw an exception.

40. In which scenarios can we interrupt a thread in Java?
Ans: In Java, we can interrupt a thread if we want to wake it up from the sleep or wait state.

41. What does sleep() method when it is called on a thread?
Ans: The sleep() method is a static method provided by the Thread class. It is used to sleep a thread for a specified amount of time.

When it is called on a thread, sleep() method pauses the current thread for a given period of time. The current thread is become out of the queue and enters into a blocked (or non-runnable state) for a specified amount of time.

When the specified amount of time is elapsed, the thread goes into the runnable state (ready state).

42. How to sleep current thread in Java?
Ans: Refer to this tutorial: Java Thread sleep.

43. What is yield in Java?
Ans: When a currently executing thread goes to the runnable state from running state, this process is called yield in Java.

44. What happens when yield() method is called on a thread object?
Ans: When the yield() method is called on a thread object, it does not move the currently running thread into sleeping, waiting, or blocking state. It sends the thread into the runnable state (ready state).

45. What does join() method in Java?
or, **What is the purpose of join() method in Thread class?**

Ans: The join() method is an instance method that is always called by an object of Thread class. It waits for a thread to die. In other words, it causes to wait for the current thread until the thread that has called the join() method completes its execution.

46. Which exception is thrown by join() method when it is interrupted?
Ans: InterruptedException. It must be enclosed in a Java try-catch block.

47. What is the fundamental difference between yield() and sleep() methods?
Ans: The main differences between yield() and sleep() methods are as follows:

a) When yield() method is called on a thread, it gives an indication to the thread scheduler that current thread is willing to yield its current use of a processor.

When sleep() method is called, it pauses the current thread for a specified amount of time.

b) When sleep() method is invoked on a thread, the thread goes from running state to waiting state and can return to runnable state when sleep time is over.

When yield() method is called, the thread goes from running state to runnable state, not in waiting state.

c) yield() method stops the thread for an unpredictable amount of time that depends on the thread scheduler. But sleep() method sleeps thread for the specified amount of time.

d) yield() method does not need to catch or throw any exception. But sleep method must catch or thrown InterruptedException.

48. What is the minimum number of threads in a Java program?
Ans: JVM executes each Java program within the main process that starts with java.exe. Therefore, each Java program has at least one thread.

49. What is synchronization in Java?
Ans: Synchronization in Java is the ability (or feature) to control the access of multiple threads to any shared resource. It is used:

- To prevent thread interference.
- To prevent consistency problem.
- To prevent deadlock between multiple threads.

50. What is thread synchronization (or thread-safe) in Java?
Ans: When a thread is already accessing an instance of a class, and preventing any other thread from acting on the same instance is called 'thread synchronization in Java' or ' Thread safe' in Java.

51. What is a synchronized object in Java?
Ans: The object on which a thread is synchronized is called synchronized object.

52. What is the purpose of thread synchronization in Java?
Ans: The purpose (objective) of thread synchronization is to control the use of any shared resource.

53. What is a monitor in Java?
Ans: A monitor is an object lock that can block and resume threads. It allows only one thread to use the shared resource (object) at a time.

54. How to acquire an object lock on a thread in Java?
Ans: To acquire an object lock on a thread, we need to call a method or a block with the synchronized keyword.

55. What is static synchronization in Java?
Ans: We define a static method as synchronized in Java, the lock will be on the class, not on object.

56. How can we achieve Synchronization in Java?
Ans: There are two ways by which we can achieve or implement synchronization in Java. They are:

- Synchronized Method
- Synchronized Block

For more detail, go to this tutorial: Synchronized method in Java

57. Can we synchronize static method in Java?
Ans: Yes, a static method can also be synchronized. In this case, the lock is placed on the class, not on the object.

58. What is a Race condition?
Ans: A race condition is an unwanted situation in which multiple threads access the same object and modify the state of object inconsistently.

Race condition can be solved with the help of synchronization mechanism in which when one thread is accessing the state of object, the other threads will wait to access the same object at a time until their come turn.

59. What is a Deadlock in Java?
Ans: A deadlock is a situation that occurs when two or more threads are waiting on each other to release a resource. Each thread is waiting for a resource that is held by the other waiting thread.

60. What happens when thread deadlock occurs in a program?
Ans: When thread deadlock occurs in a program, the further execution of the program will stop. We should take care to avoid deadlock while coding.

61. What are the conditions to occur deadlock situation in a program?
Ans: A deadlock situation in a program may occur due to the following conditions. They are as follows:

- Mutual Exclusion
- Hold and Wait
- No Preemption
- Circular Wait

62. How to avoid deadlock in Java program?
Ans: To prevent (or avoid) a deadlock from occurring in a program at least one condition for a deadlock should be removed:

- Mutual exclusion
- Resource holding
- No preemption
- Circular wait

For more detail, go to this tutorial: Deadlock in Java | Realtime Example

63. How can we detect a deadlock situation in Java program?
Ans: Use ThreadMXBean.findDeadlockedThreads() method to detect deadlock situation in Java program.

64. Which is better to use: synchronized method or synchronized block, if we need to synchronize a certain block of code?
Ans: Synchronized block is better to use for synchronization on a certain block of code or statements inside the method.

65. Can we declare a constructor as synchronized in Java?
Ans: No. We cannot declare a constructor as synchronized. This will lead to a compiler time error.

Hope that this tutorial has covered all types of **multithreading interview questions in java** with the best possible answers. Interview questions based on Java multithreading will be regularly updated further.
All the best!!!

Java