JAVA means DURGA SOFT

JAVA FAQ`S

Most important Questions In Multi Threading

DURGA M.Tech
(Sun certified & Realtime Expert)
Ex. IBM Employee
Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Multi Threading FAQs

Q1. What is Multitasking?
Ans. Executing several task simultaneously is called multitasking.

Q2. What is the difference between process-based and Thread-based Multitasking?
Ans.**Process-based multitasking:-** Executing several task simultaneously where each task is a separate independent process such type of multitasking is called process based Multitasking. Example:-While typing a program in the editor we can listen MP3 audio songs. At the same time we download a file from the net. all these task are executing simultaneously and each task is a separate independent program. hence it is process based multitasking. It is best suitable at operating system level. **Thread-based multitasking:-** Executing several task simultaneously where each task is a separate independent part of the same program is called Thread-based multitasking. and every independent part is called a thread. This type of multitasking is best suitable at programmatic level.

Q3. What is Multithreading and explain its application areas?
Ans. Executing several thread simultaneously where each thread is a separate independent part of the same program is called multithreading. Java language provides inbuilt support for multithreading by defining a reach library, classes and interfaces like Thread, ThreadGroup, Runnable etc. The main important application area of multithreading are video games implementation, animation development, multimedia graphics etc.

Q4.What is advantage of Multithreading?
Ans. The main advantage of multithreading is reduces response time and improves performance of the system.

Q5. When compared with C++ what is the advantage in java with respect to Multithreading?
Ans.Java language provides inbuilt support for multithreading by defining a reach library, classes and interfaces like Thread, ThreadGroup, Runnable etc. But in c++ there is no inbuilt support for multithreading.

Q6. In how many ways we can define a Thread? Among extending Thread and implementing Runnable which is recommended?
Ans. We can define a Thread in the following two ways:

1. by extending Thread class or
2. by implementing Runnable interface.

Among the two ways of defining a thread implementing Runnable mechanism is always recommended. In the first approach as our Thread class already extending Thread there is no chance of extending any other. Hence, we missing the key benefit of oops(inheritance properties).

Q7. What is the difference between t.start() and t.run() method?

Ans. In the case of t.start() method, a new thread will be created which is responsible for the execution of run() method.

But in the case of t.run() method no new thread will be created main thread executes run() method just like a normal method call.

Q8. Explain about Thread Scheduler?

Ans. If multiple threads are waiting for getting the chance for executing then which thread will get chance first decided by Thread Scheduler. It is the part of JVM and its behavior is vendor dependent and we can't expect exact output.Whenever the situation comes to multithreading the guarantee behavior is very- very low

Q9. If we are not overriding run() method what will happened?

Ans.If we are not overriding run() method then Thread class run() method will executed which has empty implementation and hence we will not get any output.

Q10.Is overloading of run() method is possible?

Ans.Yes, we can overload run() method but Thread class start() method always invokes no-argument run() method only. The other run() method we have to call explicitly then only will be executed.

Q11.Is it possible to override start() method?

Ans. Yes it is possible. But not recommended.

Q12.If we are overriding start() method then what will happen?

Ans. It we are overriding start() method then our own start() method will be executed just like a normal method call. In this case no new Thread will be created.

Q13. Explain life cycle of a Thread?

Ans. Once we create a Thread object then the Thread is said to be in New/Born state once we call t.start() method now the Thread will be entered into ready/Runnable state that is Thread is ready to execute. If Thread Scheduler allocates CPU now the Thread will entered into the Running state and start execution of run() method. After completing run() method the Thread entered into Dead State.

Q14. What is the importance of Thread class start() method?

Ans. Start() method present in Thread class performing all low level joining formalities for the newly created thread like registering thread with Thread Scheduler etc and then start() method invoking run() method.As the start() method is doing all low level mandatory activities, Programmer has to concentrate only on run() method to define the job. Hence, start() method is a big assistant to the programmer.Without executing Thread class start() method there is no chance of starting a new Thread.

Q15. After starting a Thread if we trying to restart the same thread once again what will happen?

Ans. After starting a Thread restarting of the same Thread once again is not allowed violation leads to Runtime Exception saying IllegalThreadStateException.

Q16. Explain Thread class constructors?

Ans. There are eight constructors are available in Thread class:

1. Thread t=new Thread();
2. Thread t=new Thread(Runnable r);
3. Thread t=new Thread(String name);
4. Thread t=new Thread(Runnable r, String name);
5. Thread t=new Thread(ThreadGroup g, String name);
6. Thread t=new Thread(ThreadGroup g, Runnable r);
7. Thread t=new Thread(ThreadGroup g, Runnable r, String name);
8. Thread t=new Thread(ThreadGroup g, Runnable r, String name, long stacksize);

Q17. How to get and set name of a Thread?

Ans. For every Thread in java there is a name. To set and get the name of a Thread we can use the following methods. All methods are final.

1. Public final void setName(String name); - To set the name of a Thread
2. Public final String getName(); - To get the name of a Thread.

Q18. What is the range of Thread priority in java?

Ans. The valid range of a Thread priority is 1-10. (1 is least priority and 10 is highest priority)

.

Q19. Who uses Thread priority?

Ans. Thread Scheduler uses priorities while allocating CPU. The Thread which is having highest priority will get chance first for execution.

Q20. What is the default priority of the Thread?

Ans. The default priority only for the main thread is 5 but for all remaining threads default priority will be inheriting from parent to child. Whatever priority parent thread has the same will be inherited to the child thread.

Q21. Once we created a new Thread what about its priority?
Ans. Whatever priority parent thread has the same will be inherited to the new child thread.
Q22. How to get and set priority of a Thread?
Ans. To get and set priority of a Thread, Thread class defines the following two methods:;

1. Public final int
getPriority();

2. Public final void setPriority(int priority);

Q23. If we are trying to set priority of a Thread as 100 what will happen?
Ans. If we are trying to set priority of a Thread as 100 then we will not get any compile time error but at the runtime we will get Runtime exception IllegalArgumentException. Because the valid range of the Thread priority is (1-10) only.

Q24. If two threads having same priority then which thread will get chance first for execution?
Ans. If two threads having same priority then which thread will get the chance first for execution decided by Thread Scheduler. It is the part of JVM and its behavior is vendor dependent and we can't expect exact output.

Q25. If two threads having different priority then which thread will get chance first for execution?
Ans. If two threads having different priority then the Thread which is having highest priority will get chance first for execution.

Q26 .How we can prevent a thread from execution?
Ans. We can prevent a Thread from executin by using the following methods:

1. Yield()
2. Join()
3. Sleep()

Q27. What is yield() method? Explain its purpose?
Ans. yield() method causes the current executing thread to pause execution and give the chance for waiting thread are same priority. If there is no waiting thread or all the remaining waiting thread have low priority then the same thread will get chance once again for execution. The Thread which is yielded when it will get chance once again for execution depends upon mercy of Thread scheduler.Public static native void yield();

Q28.What is purpose of join() method?
Ans. If a Thread wants to wait until some other Thread completion then we should go for join() method.
Example: if a Thread t1 execute t2.join() ; then t1 will entered into waiting state until t2 Thread completion.

Q29. Is join() method is overloaded?

Ans. Yes join() method is overloaded method.

Public final void join() throws InterruptedException

By using this method thread will wait up to another thread completion .

Public final void join(long ms) throws InterruptedException

By using this method thread will wail upto sometime what we are passing as a argument in millisecond

Public final void join(long ms, int ns)throws InterruptedException

By using this method thread will wait up to sometime what we are passing as a argument in millisecond and nanosecond.

Q30 What is the purpose of sleep() method?

Ans. If a Thread don't want to perform any operation for a particular amount of time then we should go for sleep() method.Whenever we are using sleep() method compulsory we should handle InterruptedException either by using try-catch or by using throws keyword otherwise we will get compile time error.

Q31. What is synchronized keyword? Explain its advantages and disadvantages.

Ans. Synchronized keyword is applicable for method and blocks only. We can't use for variables and classes.

If a method declared as a synchronized then at a time only one Thread is allow to execute that method on the given object.

The main advantages of synchronized keyword are, we can prevent data inconsistency problems and we can provide Threadsafty.

But the main limitation of synchronized keyword is it increases waiting time of Threads and effect performance of the system. Hence if there is no specific requirement it is not recommended to use synchronized keyword.

Q32.Where we can use synchronized keyword?

Ans. Synchronization concept is applicable whenever multiple Threads are operating on the same object simultaneously. But whenever multiple Threads are operating on different objects then there is no impact of synchronization.

Q33. What is object lock? Explain when it is required?

Ans. Every object in java has a unique lock whenever we are using synchronization concept then only lock concept will coming to the picture.

If a Thread wants to execute a synchronized method first it has to get the lock of the object. Once a Thread got the lock then it is allow to execute any synchronized method on that object. After completing synchronized method execution Thread releases the lock automatically.While a Thread executing synchronized method on the given object the remaining Threads are not allow to execute any synchronized method on that object simultaneously. But remaining Threads are allow to execute any non-synchronized method simultaneously. (Lock concept is implemented based on object but not based on method.)

Q34.What is the class level lock? Explain its purpose.

Ans. Every class in java has a unique lock if a Thread wants to execute static synchronized method that Thread has to get class level lock once a Thread got class level lock then only it is allow to execute static synchronized method.

While a Thread executing any static synchronized method then remaining Threads are not allow to execute any static synchronized method of the same class simultaneously. But the remaining Threads are allow to execute the following method simultaneously:

1. Any static non-synchronized method.
2. Synchronized instance methods
3. Non-synchronized instance method.

There is no relationship between object lock and class level lock, both are independent.

Q35. While a thread executing any synchronized method on the given object is it possible to execute remaining synchronized method of the same object simultaneously by any other thread?

Ans. No, it is no possible.

Q36. What is the difference between synchronized method and static synchronized method?

Ans. If a Thread wants to execute a **synchronized method** first it has to get the **lock of the object**. Once a Thread got the lock then it is allow to execute any **synchronized method on that object**.If a Thread wants to execute **static synchronized method**that Thread has to get **class level lock** once a Thread got class level lock then only it is allow to execute **static synchronized method**.

Q37. What is the advantage of synchronized block over synchronized method?
Ans. If very few lines of the code required synchronization then declaring entire method as the synchronized is not recommended. We have to declare those few lines of the code inside synchronized block. This approach reduces waiting time of the Thread and improves performance of the system.

Q38. What is synchronized statement?
Ans. The Statement which is inside the synchronized area (synchronized method or synchronized block) is called synchronized statement.

Q39. How we can declare synchronized block to get class level lock?
Ans. To get the class level lock we can declare synchronized block as follows:

```
synchronized(Display.class)
  {
   }
```


Q40. How two thread will communicate with each other?
Ans. Two Threads will communicate with each other by using wait(), notify(), notifyAll() methods.

Q41.wait(), notify(), notifyAll() method can available in which class?
Ans. These methods are defined in Object class.

Q42.Why wait(), notify(), notifyAll() method defines in object class instead of Thread class?
Ans. These methods are defined in Object class but not in Thread because Threads are calling this method on the shared object.

Q43.If a waiting thread got notification then it will entered into which state?
Ans. It will entered into another waiting state to get lock.

Q44.In which method threads can release the lock?
Ans. Once a Thread calls wait() method it immediately releases the lock of that object and then entered into waiting state similarly after calling notify() method Thread releases the lock

but may not immediately. Except these three methods(wait(), notify(), notifyAll()) method Thread never releases the lock anywhere else.

Q45. Explain wait(), notify(), notifyAll() method uses.
Ans. Two Threads will communicate with each other by using wait(), notify() or notifyAll() methods.

These methods are defined in Object class but not in Thread because Threads are calling this method.

Q46. What is the difference between notify() and notifyAll()?
Ans. To give notification to the single waiting Thread. We use notify() method and to give notification to all waiting thread we use notifyAll() method.

Q47. Once a Thread got the notification then which waiting thread will get chance?
Ans. It is depends on the Thread Scheduler.

Q48. How a thread can interrupt another thread?

Ans. A Thread can interrupt another Thread by using interrupt() method.

Q49. What is DeadLock? Is it possible to resolve DeadLock situation?
Ans. If two Threads are waiting for each other forever such type of situation is called DeadLock.

For the DeadLock, there are no resolution techniques but prevention techniques are available.

Q50. Which keyword causes DeadLock situation?
Ans. Synchronized keyword is the thing to causes of DeadLock. If we are not using properly synchronized keyword the program will entered into DeadLock situation.

Q51. How we can stop a thread explacitly?
Ans. Thread class defines stop() method by using this method we can stop a Thread. But it is deprecated. And hence not recommended to use.

Q52. Explain about suspend() and resume() method?
Ans. A Thread can suspend another Thread by using suspend() method.
A Thread can resume a suspended Thread by using resume() method.

Q53.What is Starvation()? And Explain the difference between Deadlock and Starvation?
Ans. A long waiting Thread is said to be in starvation (because of least priority) but after certain time defiantly it will get the chance for execution. But in the case of Deadlock two Threads will wait for each other forever. It will never get the chance for execution.

Q54. What is race condition?
Ans. Multiple Threads are accessing simultaneously and causing data inconsistency problem is called race condition, we can resolve this by using synchronized keyword.

Q55. What is Daemon Thread? And give an example?
Ans. The Threads which are running in the background are called Daemon Thread.
Example: Garbage collector.

Q56. What is the purpose of a Daemon Thread?
Ans. The main purpose of Daemon Threads is to provide support for non-daemon Threads.

Q57. How we can check Daemon nature of a Thread?
Ans. We can check Daemon nature of a Thread by using **isDaemon()** method.
Q58. Is it possible to change a Daemon nature of a Thread?
Ans. Yes, we can change Daemon nature of a Thread by using **setDaemon()** method.
Q59. Is main thread is Daemon or non-daemon?
Ans. By default main thread is always non-daemon nature.
Q60. Once we created a new thread is it daemon or non-daemon.
Ans. Once we created a new Thread, The Daemon nature will be inheriting from parent to child. If the parent is Daemon the child is also Daemon and if the parent is non-daemon then child is also non-daemon.

Q61. After starting a thread is it possible to change Daemon nature?
Ans. We can change the Daemon nature before starting the Thread only. Once Thread started we are not allow to change Daemon nature otherwise we will get RuntimeException sying IllegalThreadStateException.

Q62. When the Daemon thread will be terminated?
Ans. Once last non-daemon Thread terminates automatically every Daemon Thread will be terminated.

Q63. What is green Thread?
Ans. A green thread refers to a mode of operation for the Java Virtual Machine (JVM) in which all code is executed in a single operating system thread. If the Java program has any concurrent threads, the JVM manages multi-threading internally rather than using other operating system threads.
There is a significant processing overhead for the JVM to keep track of thread states and swap between them, so green thread mode has been deprecated and removed from more recent Java implementations.

Q64.Explain about Thread group?

Ans. Every Java thread is a member of a *thread group*. Thread groups provide a mechanism for collecting multiple threads into a single object and manipulating those threads all at once, rather than individually. For example, you can start or suspend all the threads within a group with a single method call. Java thread groups are implemented by the ThreadGroup api class in the java.lang package.

Q65.What is the Thread Local?

Ans. It's a way for each thread in multi-threaded code to keep its own copy of an instance variable. Generally, instance variable are shared between all threads that use an object; ThreadLocal is a way for each thread to keep its own copy of such a variable. The purpose might be that each thread keeps different data in that variable, or that the developer wants to avoid the overhead of synchronizing access to it.

Q66. In your previous project where you used multithreading concept?