1. **Explain multithreading in Java?**

Ans. 1. Multithreading provides better interaction with the user by distribution of task  
2. Threads in Java appear to run concurrently, so it provides simulation for simultaneous activities.  
The processor runs each thread for a short time and switches among the threads to simulate simultaneous execution (context-switching) and it make appears that each thread has its own processor. By using this feature, users can make it appear as if multiple tasks are occurring simultaneously when, in fact, each is   
running for only a brief time before the context is switched to the next thread.  
3. We can do other things while waiting for slow I/O operations.  
In the java.iopackage, the class InputStream has a method, read(), that blocks until a byte is read from the stream or until an IOException is thrown. The thread that executes this method cannot do anything else while awaiting the arrival of another byte on the stream.

1. **Can constructors be synchronized in Java?**

Ans. No. Java doesn't allow multi thread access to object constructors, so synchronization is not even needed.

1. **Java Concurrency – yield (), sleep () and join () methods**

**yield ():** Suppose there are three threads t1, t2, and t3. Thread t1 gets the processor and starts its execution and thread t2 and t3 are in Ready/Runnable state. Completion time for thread t1 is 5 hour and completion time for t2 is 5 minutes. Since t1 will complete its execution after 5 hours, t2 must wait for 5 hours to just finish 5 minutes job. In such scenarios where one thread is taking too much time to complete its execution, we need a way to prevent execution of a thread in between if something important is pending. yield () helps us in doing so.  
**yield ()**basically means that the thread is not doing anything particularly important and if any other threads or processes need to be run, they should run. Otherwise, the current thread will continue to run.

**Use of yield method:**

* Whenever a thread calls java.lang.Thread.yield method, it gives hint to the thread scheduler that it is ready to pause its execution. Thread scheduler is free to ignore this hint.
* If any thread executes yield method, thread scheduler checks if there is any thread with same or high priority than this thread. If processor finds any thread with higher or same priority, then it will move the current thread to Ready/Runnable state and give processor to other thread and if not – current thread will keep executing.

**Sleep ():** This method causes the currently executing thread to sleep for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.  
 **Note**:

* Based on the requirement we can make a thread to be in sleeping state for a specified period
* Sleep () causes the thread to stop executing for a given amount of time; if no other thread or process needs to be run, the CPU will be idle (and probably enter a power saving mode).
* **yield () vs sleep ()**
* **yield:()** indicates that the thread is not doing anything particularly important and if any other threads or processes need to be run, they can. Otherwise, the **current thread will continue to run.**
* **sleep ()**: causes the thread to stop executing for a given amount of time; if no other thread or process needs to be run, **the CPU will be idle** (and probably enter a power saving mode).

[**join ():**](https://www.geeksforgeeks.org/joining-threads-in-java/) The join () method of a Thread instance is used to join the start of a thread’s execution to end of other thread’s execution such that a thread does not start running until another thread ends. If join () is called on a Thread instance, the currently running thread will block until the Thread instance has finished executing.  
The join () method waits at most this much milliseconds for this thread to die. A timeout of 0 means to wait forever.

Note:

* If any executing thread t1 calls join () on t2 i.e. t2.join() immediately t1 will enter into waiting state until t2 completes its execution.
* Giving a timeout within join (), will make the join() effect to be nullified after the specific timeout.

1. **What is the difference between time slicing and preemptive scheduling?**

Ans. In preemptive scheduling, highest priority task continues execution till it enters a not running state or a higher priority task comes into existence. In time slicing, the task continues its execution for a predefined period and reenters the pool of ready task

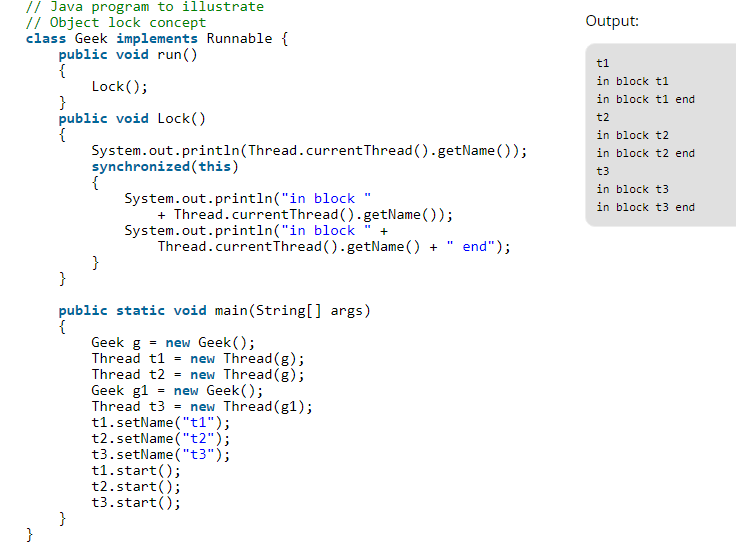
# Object level and Class level locks in Java

**Synchronization :**[Synchronization](https://www.geeksforgeeks.org/synchronized-in-java/) is a modifier which is used for method and block only. With the help of synchronized modifier, we can restrict a shared resource to be accessed only by one thread. When two or more threads need access to shared resources, there is some loss of data i.e. data inconsistency. The process by which we can achieve data consistency between multiple threads it is called Synchronization.

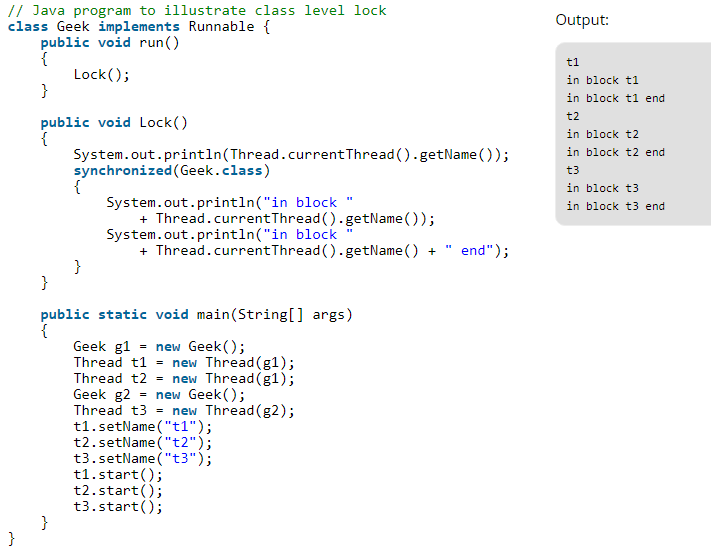
**There are two types of locks on threads:**

* 1. **Object level lock:**Every object in java has a unique lock. Whenever we are using synchronized keyword, then only lock concept will come in the picture. If a thread wants to execute synchronized method on the given object. First, it must get lock of that object. Once thread got the lock then it can execute any synchronized method on that object. Once method execution completes automatically thread releases the lock. Acquiring and release lock internally is taken care by JVM and programmer is not responsible for these activities. Let’s have a look on the below program to understand the object level lock

**Object level lock** is mechanism when we want to synchronize a **non-static method** or **non-static code block** such that only one thread will be able to execute the code block on given instance of the class. This should always be done **to make instance level data thread safe**.



* 1. **Class level lock:**Every class in java has a unique lock which is nothing but class level lock. If a thread wants to execute a static synchronized method, then thread requires class level lock. Once a thread got the class level lock, then it can execute any static synchronized method of that class. Once method execution completes automatically thread releases the lock. Let’s look on the below program for better understanding:



**6. What is Volatile keyword used for?**

Ans. Volatile is a declaration that a variable can be accessed by multiple threads and hence shouldn't be cached.