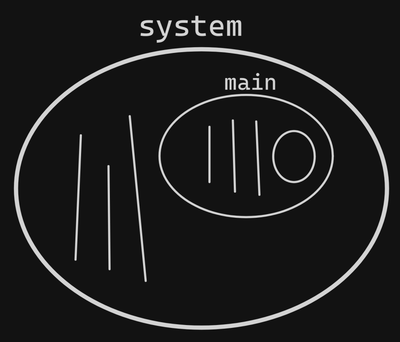
Synchronization

⇒ What is difference between sleep() and wait() method ?

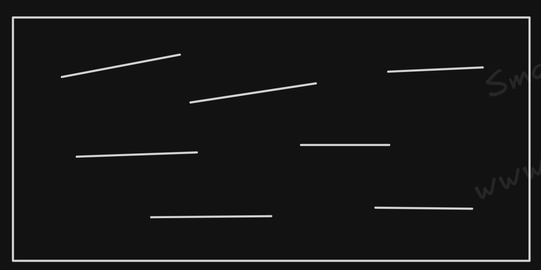
⇒ Disadvantages of Synchronization :-
1. Application becomes slow because every thread is executed one by one
2. Due to synchronization, deadlock condition may occur

⇒ To resolve this deadlock problem, java introduced one new package in JDK 1.5 version
- java.util.concurrent.locks
  - >> Lock interface
  - >> ReenterantLock class

⇒ ThreadGroup class



⇒ ThreadPool:
- → It is the pool of created threads waiting for the task



- → For this java has provided one framework i.e.
  - >> ThreadPool Framework or Executor Framework
    - Executors class
    - ExecutorService interface

Interview Questions

⇒ Lock (Interface)
1. What is the Lock interface in Java?
2. How is Lock different from using the synchronized keyword?
3. What are the main methods provided by the Lock interface?
4. What is the purpose of the tryLock() method?
5. What happens if a thread forgets to call unlock() after acquiring a lock?
6. Can multiple threads acquire the same lock simultaneously? Why or why not?

⇒ ReentrantLock
7. What is ReentrantLock in Java?
8. How is ReentrantLock different from synchronized?
9. What is the meaning of “reentrant” in ReentrantLock?
10. How do you create and use a ReentrantLock in Java?
11. What is the purpose of lockInterruptibly()?
12. What is the difference between lock(), tryLock(), and lockInterruptibly()?
13. How do you make sure a ReentrantLock is released properly?
14. Does ReentrantLock support fairness? How?
15. When would you prefer ReentrantLock over synchronized?

⇒ ThreadGroup
16. What is a ThreadGroup in Java?
17. How do you create a thread inside a ThreadGroup?
18. What is the purpose of activeCount() and activeGroupCount() methods?
19. Can we set the maximum priority of a ThreadGroup? What happens if we do?
20. What happens to a ThreadGroup when all its threads are finished?
21. Is ThreadGroup widely used in modern Java applications? Why or why not?

⇒ ThreadPool
22. What is a thread pool in Java?
23. Why do we need a thread pool instead of creating new threads?
24. How do you create a thread pool in Java?
25. What is the role of the Executor and ExecutorService interfaces?
26. What are the different ways to create a thread pool using Executors?
27. What is the difference between newFixedThreadPool(), newCachedThreadPool(), and newSingleThreadExecutor()?
28. How do you gracefully shut down a thread pool?
29. What happens if you submit more tasks than the pool size?
30. What are the advantages and drawbacks of using a thread pool?