1. What is Thread and What is Process? What are differences between them?
   A process is an active program i.e. a program that is under execution. A process generally has a complete, private set of basic run-time resources; in particular, each process has its own memory space.

   A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution that can be managed independently by a scheduler.

Comparison Basis	Process	Thread
Definition	A process is a program under execution i.e an active program.	A thread is a lightweight process that can be managed independently by a scheduler.
Context switching time	Processes require more time for context switching as they are more heavy.	Threads require less time for context switching as they are lighter than processes.
Memory Sharing	Processes are totally independent and don't share memory.	A thread may share some memory with its peer threads.
Communication	Communication between processes requires more time than between threads.	Communication between threads requires less time than between processes .
Blocked	If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.
Resource Consumption	Processes require more resources than threads.	Threads generally need less resources than processes.
Dependency	Individual processes are independent of each other.	Threads are parts of a process and so are dependent.
Data and Code sharing	Processes have independent data and code segments.	A thread shares the data segment, code segment, files etc. with its

Comparison Basis	Process	Thread
		peer threads.
Treatment by OS	All the different processes are treated separately by the operating system.	All user level peer threads are treated as a single task by the operating system.
Time for creation	Processes require more time for creation.	Threads require less time for creation.
Time for termination	Processes require more time for termination.	Threads require less time for termination.

2. How to create threads in Java?
   Extends the Thread class
   Implements the Runnable interface

3. Runnable or Thread, which do you prefer to use? Why?
   Simply put, we generally encourage the use of Runnable over Thread:
   • When extending the Thread class, we're not overriding any of its methods. Instead, we override the method of Runnable (which Thread happens to implement). This is a clear violation of IS-A Thread principle
   • Creating an implementation of Runnable and passing it to the Thread class utilizes aggregation/composition and not inheritance – which is more flexible
   • After extending the Thread class, we can't extend any other class
   • From Java 8 onwards, Runnables can be represented as lambda expressions
4. What are differences between start() and run()?
   When a program calls the start() method, a new thread is created and then the run() method is executed
   When a program directly call the run() method then no new thread will be created and run() method will be executed as a normal method call on the current calling thread itself and no multi-threading will take place

5. What if invoking the start() method of a thread twice? What if invoking the run() method twice?

No. After starting a thread, it can never be started again. If you does so, an IllegalThreadStateException is thrown. In such case, thread will run once but for second time, it will throw exception.

Yes, you can call run() twice.

6. What is the Thread Life Cycle in Java? Explain how to get to each stage.

Five states of thread

- new — just created but not started
- runnable — created, started, and able to run
- running — thread is currently running
- blocked — created and started but unable to run because it is waiting for some event to occur
- dead — thread has finished or been stopped

New->{(start())runnable->(run())running->(sleep or join)blocked}->(exit)dead

7. Explain join()method in Java Thread.

Join — The join method allows one thread to wait for the completion of another. causes the current thread to pause execution until t's thread terminates. Overloads of join allow the programmer to specify a waiting period.

8. What is wait(), sleep(), yield()? What are differences among them?

three methods that can be used to pause a thread in java.

sleep() and yield() methods are defined in thread class while wait() is defined in the Object class,

The key difference between wait() and sleep() is that the former is used for inter-thread communication while later is used to introduced to pause the current thread for a short duration.

when a thread calls the wait() method, it releases the monitor or lock it was holding on that object, but when a thread calls the sleep() method, it never releases the monitor even if it is holding.

Coming back to yield(), it's little different than wait() and sleep(), it just releases the CPU hold by Thread to give another thread an opportunity to run though it's not guaranteed who will get the CPU.

9. What is notify()?

The notify() method is defined in the Object class, which is Java's top-level class. It's used to wake up only one thread that's waiting for an object, and that thread then begins execution. The thread class notify() method is used to wake up a single thread. If multiple threads are waiting for notification, and we use the notify() method, only one thread will receive the notification, and the others will have to wait for more. This method does not return any value.

10. What is a Daemon thread in Java? Why do we need it?

Daemon thread in Java provides service to the user thread which runs in the background. It is considered to be a low priority thread which is used to perform tasks such as garbage collection.

11. What is thread interference? Give an example.

Interference happens when two operations, running in different threads, but acting on the same data, interleave. This means that the two operations consist of multiple steps, and the sequences of steps overlap.

```
class Counter {
    private int c = 0;
    public void increment() { c++;    }
```

```
    public void decrement() { c--; }
    public int value() {       return c;}
}
```

12. What is memory consistency error? Give an example.

Memory consistency errors occur when different threads have inconsistent views of what should be the same data.

Same Counter class as before — Thread A increases the counter by 1, and thread B tries to print the value of counter. Now the value can either be 1 or 0

13. What are ways of Thread Synchronization?

Synchronization can be achieved in two ways

- By using synchronized keyword with method definition
- By using synchronized keyword with any block of code

14. What is Deadlock? How to resolve it?

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

Avoid Nested Locks: A deadlock mainly happens when we give locks to multiple threads. Avoid giving a lock to multiple threads if we already have given to one.

Avoid Unnecessary Locks: We can have a lock only those members which are required. Having a lock unnecessarily can lead to a deadlock.

Using Thread.join(): A deadlock condition appears when one thread is waiting other to finish. If this condition occurs we can use Thread.join() with the maximum time the execution will take.