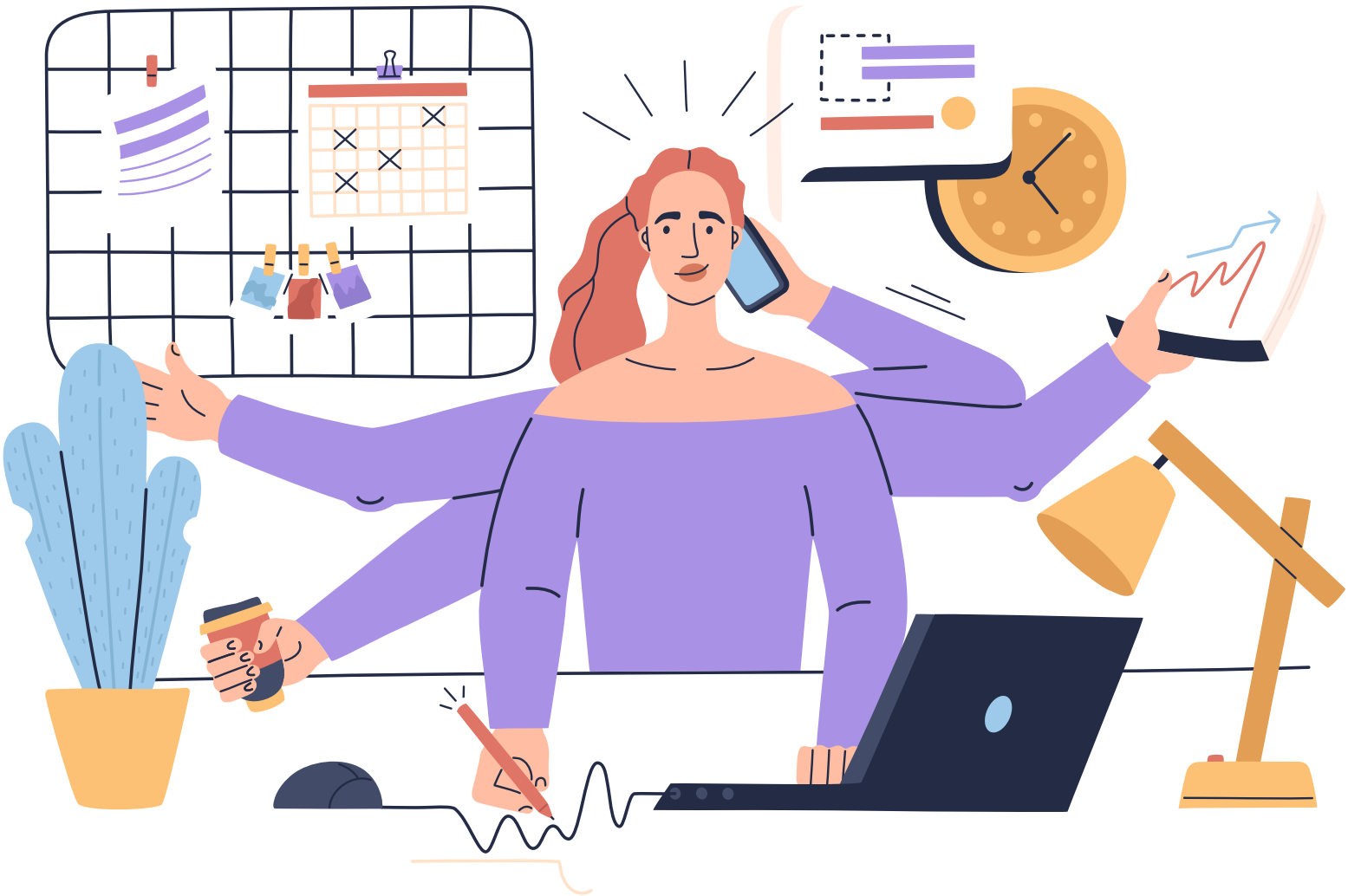
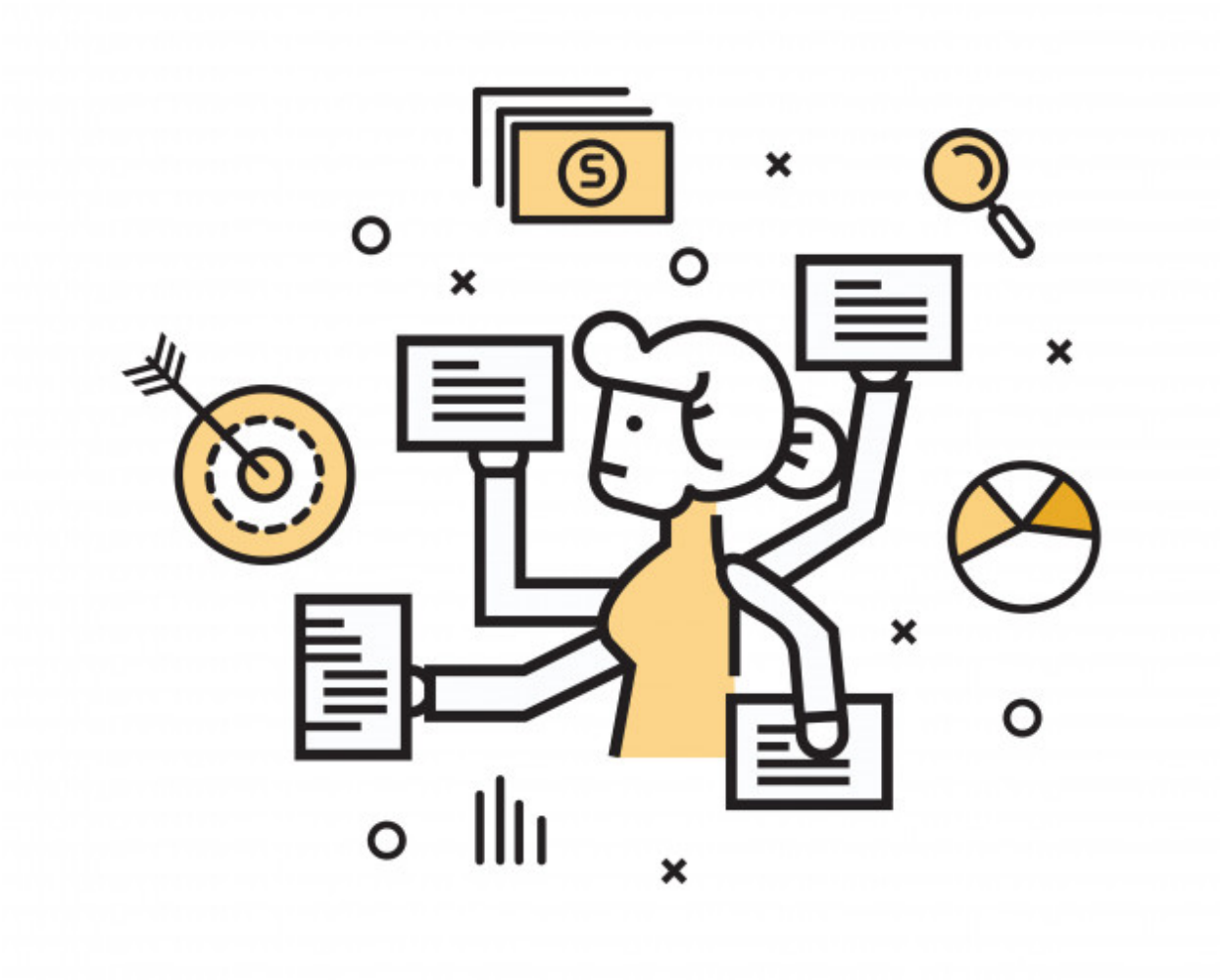
ALL ABOUT THREADING



PANGASA ANKIT



How To Explain Threads

- The thread is an independent path of execution.
- We can create multiple threads in java, even if we don't create any Thread, one Thread at least do exist i.e. main thread.
- Multiple threads run in parallel.
- There are several ways to achieve thread safety in java – synchronization, atomic concurrent classes
- When we use volatile keyword with a variable, all the threads read it's value directly from the memory and don't cache it

Question: What is Thread in Java?

A thread is an independent path of execution. It's a way to take advantage of multiple CPU available in a machine.

- Threads consumes CPU in best possible manner, hence enables multi processing. Multi threading reduces idle time of CPU which improves performance of application.
- Thread are light weight process.
- A thread class belongs to java.lang package.
- We can create multiple threads in java, even if we don't create any Thread, one Thread at least do exist i.e. main thread.

- Multiple threads run parallel in java.
- Threads have their own stack.

Question: Does multi-threading improve performance? How?

For a simple task of iterating 100 elements multi-threading the task will not provide a performance benefit.

Iterating over 100 billion elements and do processing on each element, then the use of additional CPU's may well help reduce processing time.

And more complicated tasks will likely incur interrupts due to I/O for example.

When one thread is sleeping waiting for a peripheral to complete I/O (e.g. a disk write, or a key press from the keyboard), other threads can continue their work.

Question: How to Implement Thread in java?

In Java threads can be implemented by following 2 ways

1. By extending Thread class
2. By implementing Runnable interface.

By Thread Class:

We can create thread by extending the **“Thread ”** class. Then the class

- Must override run() method.
- And Call start() method to start executing the thread object.

```
public class CodeSpaghetti extends Thread{
@override
public void run()
   {
   System.out.println("Write your logic here");
    }
}
```

And we can execute the thread like `public class ThreadDemo { public static void main(String args[]) { CodeSpaghetti.start(); } }`

By Runnable interface:

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run(). public void run(): is used to perform action for a thread.

```
class CodeSpaghetti implements Runnable{
public void run(){
System.out.println("Write your logic here");
}

public static void main(String args[]){
CodeSpaghetti codeSpaghetti=new CodeSpaghetti();
Thread thread =new Thread(codeSpaghetti);
thread.start();
 }
}
```

Question: What is Thread Life Cycle in Java?

Different states in Thread life cycle?

During the lifecycle of a thread it has many states. These states are described below.

1. New

A thread has state new, when we create it using *new* operator At this point, thread is not alive and it's a state internal to Java programming.

2. Runnable

A thread is changed to state Runnable when we call start() method on Thread object. The control is given to Thread scheduler to finish its execution.

3. Running

A thread is changed to Running when thread is executing. Thread scheduler picks one of the thread from the runnable thread pool and change it's state to Running. Then CPU starts executing this thread.

4. Blocked/Waiting

A thread can be waiting for other thread to finish using thread join or it can be waitingfor some resources to become available.

Then its state is changed to Waiting. Once the thread wait state is over, its state is changed to **Runnable** and it's moved back to runnable thread pool.

5. Dead

Once the thread finished executing, it's state is changed to Dead and it's considered to be not alive.

Question: What is the purpose of sleep() method?

sleep() method is used to pause the execution of thread for certain time. Once the thread awake from sleep.

It’s state changes back to runnable and based on thread scheduling, it gets executed.

Question: What is thread priority in java?

We can specify the priority of thread to higher or lower. But it doesn’t guarantee that higher priority thread will get executed before lower priority thread.

Thread priority is an int whose value varies from 1 to 10 where 1 is the lowest priority thread and 10 is the highest priority thread.

Every thread has a priority, usually higher priority thread gets precedence in execution but it depends on Thread Scheduler implementation that is OS dependent.

Question: What is thread scheduler and time slicing in java?

Thread Scheduler is the Operating System service that allocates the CPU time to the available runnable threads.

Once we create and start a thread, it’s execution depends on the implementation of Thread Scheduler.

Time Slicing is the process to divide the available CPU time to the available runnable threads.

Allocation of CPU time to threads can be based on thread priority or the thread waiting for longer time will get more priority in getting CPU time.

Thread scheduling can’t be controlled by java, so it’s always better to control it from application itself.

Question: How we can achieve Thread Safety in Java?

Thread safety means that a method or class instance can be used by multiple threads at the same time without any problems.

There are several ways to achieve thread safety in java like synchronization, atomic concurrent classes, using volatile keyword,implementing concurrent Lock interface, using immutable classes and Thread safe classes

Question: What is Volatile Keyword in Java?

When a variable is declared with volatile keyword, all the threads will read its value directly from memory and will not cache it.

This makes sure that the value read is the same as in the memory.

Question: How threads communicates with each other?

There are three final methods that allows threads to communicate about the lock status of a resource.

These methods are **wait()**, **notify()** and **notifyAll()**.

Question: Why Thread's Sleep() and Yield() methods are Static in Java?

Thread sleep() and yield() methods work on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state.

That’s why these methods are made static so that when this method is called statically.

Question: What are the daemon threads in Java?

The daemon threads are the low priority threads that provides the background support and services to the user threads.

Question: What is a shutdown hook in Java?

The shutdown hook is a thread that is invoked implicitly before JVM shuts down. Its main purpose is to perform some cleanup of resources.

Question: What is a Deadlock in Java?

When two threads are waiting for each other to release the resources a dead lock situation occurs. In simple words each thread is waiting for a resource which is held by another waiting thread.

Question: Can a Thread Started Twice in Java?

Once a thread is started, it can never be started again. Doing so will throw an illegalThreadStateException.

Question: What is Yield method in method?

yield() method causes the currently executing thread object to temporarily pause and allow other threads to execute.

If there is no waiting thread or all the waiting threads have a lower priority than the current thread, then the same thread will continue its execution.

Question: What is Context-Switching in Java?

Context Switching is the process of storing and restoring of CPU state so that Thread execution can be resumed from the same point at a later point of time.

Question: What is ThreadLocal in java?

Java ThreadLocal is used to create thread-local variables. We know that all threads of an Object share it's variables.

So if the variable is not thread safe, we can use synchronization but if we want to avoid synchronization, we can use ThreadLocal variables.

Question: What are Thread Pool? How can we create Thread Pool in Java?

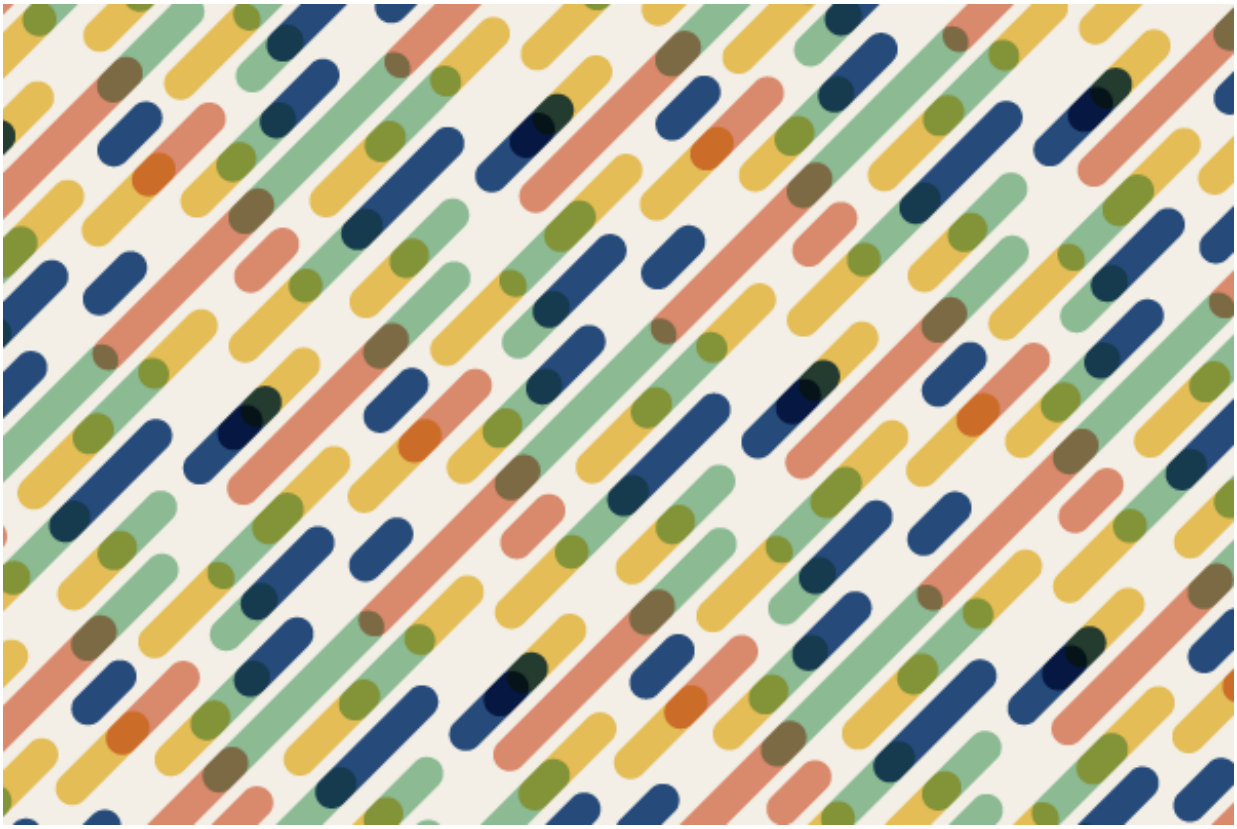
A thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed.

A thread pool manages the collection of Runnable threads and worker threads execute Runnable from the queue.

java.util.concurrent.Executors provide implementation of java.util.concurrent.Executor interface to create the thread pool in java

Important Note: Quite often multithreading questions are also asked in relation to Arrays and Linked lists. Make sure you go through those interview questions as well

Java Concurrency Interview Questions



Question: What is Atomic Operation in Java?

Atomic operations are performed in a single unit of task without interference from other operations.

Atomic operations are necessity in multi-threaded environment to avoid data inconsistency.

Question: What is Lock interface in Java Concurrency API?

Lock interface provide more extensive locking operations than can be obtained using synchronized methods and statements.

They allow more flexible structuring, may have quite different properties, and may support multiple associated Condition objects.

Question: What is Executors Framework?

The Executor framework is a framework for standardizing invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies.

Question: What is BlockingQueue?

java.util.concurrent.BlockingQueue is a Queue that supports operations that wait for the queue to become non-empty.

When retrieving and removing an element, and wait for space to become available in the queue when adding an element.

Question: What are Concurrent Collection Classes in Java?

Java Collection classes are fail-fast which means that if the Collection will be changed while some thread is traversing over it using iterator.

The iterator.next() will throw ConcurrentModificationException.Concurrent Collection classes support full concurrency of retrievals and adjustable expected concurrency for updates.

Major classes are

ConcurrentHashMap

- CopyOnWriteArrayList
- CopyOnWriteArraySet

Question: Why Concurrent Collection Classes are fail-fast in Java?

Java Collection classes are fail-fast which means that if the Collection will be changed while some thread is traversing over it using iterator, the iterator.next() will throw ConcurrentModificationException.

Question: What is an Executors Class?

Executors class provide utility methods for Executor, ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable classes.

Executors class can be used to easily create Thread Pool in java, also this is the only class supporting execution of Callable implementations