THREADS

Is parallel universes possible in computer?

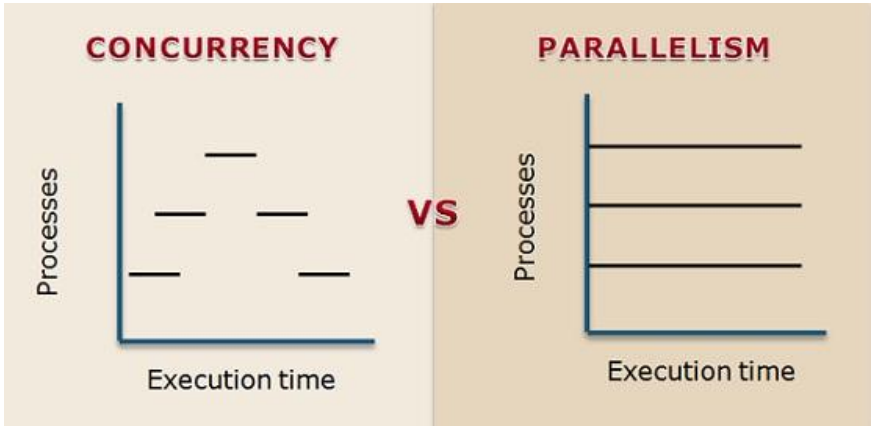
Sequential vs. Parallel programming

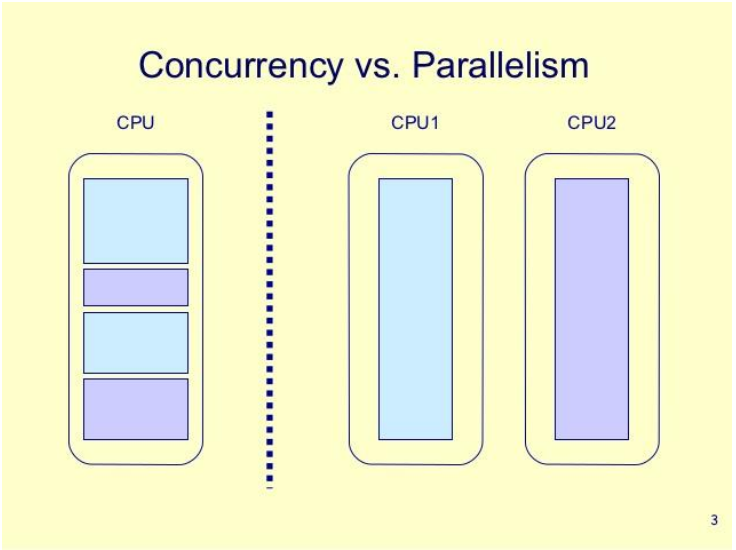
In **sequential** composition, different **program** components execute in sequence on all processors. In **parallel** composition, different **program** components execute concurrently on different processors. ... In **parallel** composition, different modules execute concurrently on disjoint sets of processors.

Multitasking vs. multithreading

The basic difference between Multitasking and multithreading is that **Multitasking** allows CPU to perform multiple tasks (program, process, task, threads) simultaneously whereas, **Multithreading** allows multiple threads of the same process to execute simultaneously.

Parallel and concurrent execution





Implementation in java

Let’s write some code

question

How many threads are executed in this code?

Answer : 5 threads

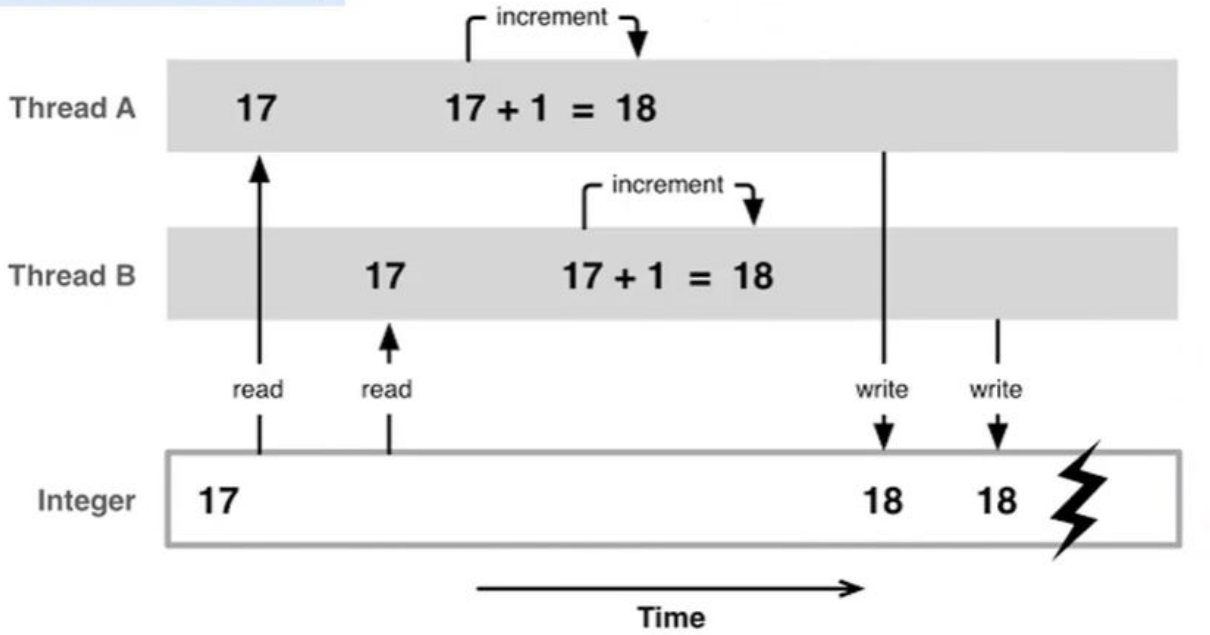
```
class T extends Thread {
    public void run() {
        for (int i = 1; i <= 100; i++)
            System.out.println(i);
    }
}
class R implements Runnable{
    public void run() {
        for (char c = 'A'; c < 'Z'; c++)
            System.out.println(c);
    }
}
public class Threading{
    public static void main(String[] args) {
        new Thread(new R()).start();
        new T().start();
        new Thread(new R()).start();
        new T().start();
        for (char c = 'a'; c < 'z'; c++)
            System.out.println(c);
```

Critical section

A **critical section** is a **section** of code that is executed by multiple **threads** and where the sequence of execution for the **threads** makes a difference in the result of the concurrent execution of the **critical section**.

Critical section

Int x = 17;
ThreadA : x++;
ThreadB : x++;



Synchronize

```java
public class Bank {
    int money;
    public synchronized int deposit(int x){
        money += x;
        return money;
    }
    public synchronized int withdraw(int x){
        money -= x;
        return money;
    }
}
```

```java
List<String> names;
...
synchronized(names){
  names.add("ali");
}
```