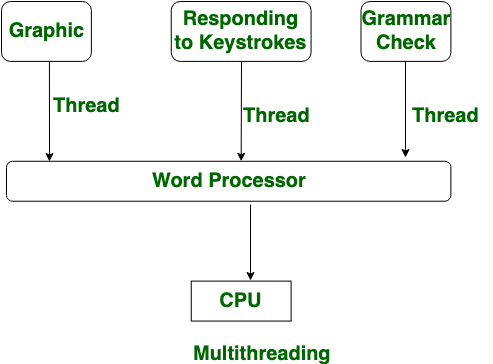
**MULTITHREADING IN JAVA**

Multithreading in Java is the act of executing a complex process using virtual processing entities that are independent of each other.

**What does this mean?**

Consider an example where a chef is cooking 4 dishes in a kitchen alone. The best strategy for the chef is to use 4 gas stoves to cook food simultaneously. This way, each stove can cook a different dish, leading to 4 cooking processes happening at the same time.

Similarly, in multithreading, different threads can work independently of each other, allowing a program to perform multiple tasks simultaneously, such as downloading files, playing music, or processing data.



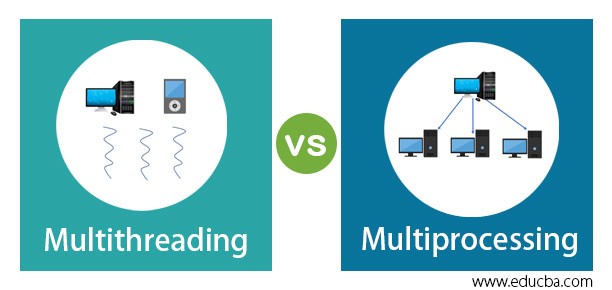
**Real-life scenario in CS of multithreading**

When using a mobile phone, you might be browsing the internet, listening to music, and have a book reading app open in the background. Meanwhile, you are receiving notifications from your SMS app and someone is calling you. Each of these activities can be considered as different tasks and processes, managed by different threads, all happening on one device.

**Why do we use multithreading?**

Multithreading helps minimize execution time and maximize CPU utilization by executing multiple tasks simultaneously.

**In Java, multithreading can be achieved with the help of two processes:**



1. **Multiprocessing:** This is purely based on the number of processors available. Every process initiated by the user is sent to the CPU.

|  |  |
| --- | --- |
| METHODS | DESCRIPTION |
| Start() | Initiates the execution |
| Current Thread() | Return the reference of the current executing thread object |
| Run() | The run method triggers an action for the thread |
| isAlive() | It is invoke to verify whether the thread is dead or alive |
| Sleep() | Suspend the thread temporarily |
| Yield() | The yield method is used to send the currently running thread to standby and runs with different sets of thread on higher priority |
| Suspend() | Instantly suspend |
| Resume() | resume |
| Interrupt() | May triggers interruption to the current executing thread |
| Destroy | Destroy execution in group of thread |
| Stop() | Stops the execution |
|  |  |

1. **Multithreading:** In Java, this approach is similar to multiprocessing, but there are fundamental differences between the two. Instead of physical processes, multithreading involves virtual and independent threads. Each thread executes independently, and if one thread is terminated unexpectedly, the entire process does not come to a halt.

METHODS of multithreading in java