**Multithreading**:

What is a thread? – when you want to perform a big task and this task can be broken into small tasks which are called threads.

Multi tasking? – At a time we can run multiple applications ex: you can document something on word, browse a website on browser, watch a movie on a media player, play a game listening to online songs etc..

Now a days we have Multi core cpu’s which support which means we have multiple cores available to handle multiple applications. Ex: when you are writing something on word document we have other features as well like spell check, editor, fonts, formatting and many other things

Another example: driving a car and we have opponents, navigation actions related to game.

Navigate to task manager on your laptop/desktop and you can check how many cores are available. Below screenshot provides more information: how many threads, when you open an application some threads get created and when we close some applications some will get closed.

Table

Description automatically generated

How can we understand this in programming way?

Let’s check the below code: 2 classes with similar methods inside printing 5\*hello and 5\*Hi

First it prints hello and then hi as we are maintaining the sequence.

Graphical user interface, application, Teams

Description automatically generated

Our machine takes sometime to print the output. Let’s say it takes 5 seconds to print 5 hellos and 5 seconds to print 5 hi’s in total 10 seconds. This machine is using one core for this even though we have multiple cores and it will still use only one core.

* Now we want to run the hello and hi simultaneously which means at the same it, can we run the two functions at a time on **different cores** or on the same core simultaneously.
* Remember by default every execution has one thread, so even if you are not creating a thread by yourself, you do have one thread and that thread is known as ***main thread.*** The previous execution is done by main thread, but we don’t want to work with main thread. We want to print hello and hi but then with the help of two different threads.
* Let’s do it by making hello as sub class of thread class as shown below: to use thread class we need to import threading package import \* as shown below

Diagram

Description automatically generated with medium confidence

***main thread*** gets created by default which will execute all the statements and it will creates two separate threads for **t1.run() and t2.run() which prints hello\*5 and hi\*5—**but when we run the below code it gives the same output as before.

Graphical user interface

Description automatically generated with medium confidence

* To make it work we need use **t1.start() and t2.start()** methods. Let’s observe the Thread class(ctrl+left click) and it have the run method inside of it. Hence when we say start() it automatically calls run method behind the scenes.
* Change the code as shown below and observe the output

Teams

Description automatically generated with medium confidence

We are still getting the same output – what’s happening is these two threads are running simultaneously to prove it we will print the hello\*500 and hi\*500 as shown below

Graphical user interface, application, Teams

Description automatically generated

* It is happening in parallel however the system is so fast that it is executing them at the same point, there is a collision. In our system we have a concept called schedulers(which will give specific time to execute). We are executing it to print one hello/hi to print at a time however it is so smart that it is printing 10 or more times in that small time gap. Now we will regulate it or increase the gap or slow down. How do we do it? – check the code below.
* We need to use sleep which can be used by importing time package. Sleep for 1 second

Graphical user interface

Description automatically generated

Why sleep? Here we are printing a simple hello/hi. However in real world application there will be huge methods which require certain time to execute hence we need to know the use of sleep method.

* When we re run the code we are getting the collision as shown below. Reason is two threads are coming to the cpu at the same time, hence it is printing hihello at a time. So sleeping is fine however, when they wake up they are going to the cpu at the same time hence collision.

A person wearing glasses

Description automatically generated with medium confidence

* Let’s address the collision issue as shown below by making use of between the two methods as shown below.

A picture containing graphical user interface

Description automatically generated

* When we say start it will create 3 threads in total 1. Main thread 2. T1 and 3.t2

Text

Description automatically generated with medium confidence

* Now let’s say we want to print bye at the end of the code. Usually it should print after all the hi and hello once they are finished it should print bye. However, it is not the case. It will print it bye after the first hello hi.
* Question is who is responsible to print bye is it t1 thread/t2 thread or the main thread?
* Graphical user interface, application, Teams

  Description automatically generated
* It is the main thread that is responsible to print the bye here. First t1 print hello second t2 print hi third then comes the main thread which prints the bye and from there t1 and t2 threads print the remaining hello and hi’s.

Now we need to tell main thread –“hey main thread wait until the t1 and t2 threads to ***join***. How do we do this? – check below screenshot.

A person wearing glasses

Description automatically generated with low confidence

We are saying the main thread that let t1 and t2 join then only we will print the bye.