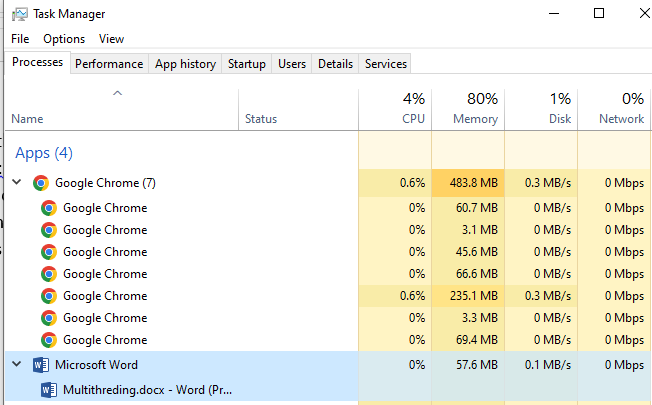
1. Multitasking vs multithreading
   1. Multitasking : Multiple Processes/Application running in a machine. It is OS based. Each process/application is independent having its own separate resources(memory/cpu).
   2. Multithreading: It is process based. Multiple thread of single process run simultaneously. They will share the resources of the process.
      1. 
2. Sequential VS Parallel Execution
   1. Sequential Execution: Every method is executed one after the another. One method call will wait for the another method to return.

**public** **class** Test {

**public** **static** **void** main(String[] args)

{

*a*();

*b*();

*c*();

}

**static** **void** a()

{

//100 lines

}

**static** **void** b()

{

//200 lines

}

**static** **void** c()

{

//5 lines

}

}

* 1. Parallel Execution : Every method is executed parallely/simultanously. One method call will not wait for another method to return, instead they will be called parallel. This parallel implementation is possible using Threads

1. To implement parallel execution:
   1. Create the thread object
   2. Assigning the task to the thread. This task will be executed parallelly
   3. In java application, there is by default one thread always created by JVM. This thread is the main thread to execute the main method of the application. Other than this main thread we can create more thread using following techniques.
      1. By extending Thread class
      2. By implementing Runnable interface
      3. In both the cases the implementation of the thread should be provided in the run() method
      4. We never call the run() method directly. Instead we will start the thread first by using strat() then, the run() will be called automatically.
      5. If we call run() then it will execute sequentially
      6. Runnable interface is a functional interface, it defines only one method to be implemented by any class.
   4. In below code, if we call run() explicitly then it will executes sequentially (by extending Thread class)

**class** CustomThread **extends** Thread

{

@Override

**public** **void** run() {

//this function will contain the

//logic to be executed in parallel with main()

System.***out***.println("Run Method of CustomThread class");

**for**(**int** i=1;i<=5;i++){

System.***out***.println(Thread.*currentThread*().getName() + " " + i);

}

}

}

**public** **class** ExtendsThreadClass {

**public** **static** **void** main(String[] args) {

System.***out***.println("Main Method");

//Create the object of the thread class

//New stage of the thread

CustomThread th1 = **new** CustomThread();

CustomThread th2 = **new** CustomThread();

//if we call the run, rather than running

//the run() method parallel to main()

//the run() will be executed sequencially

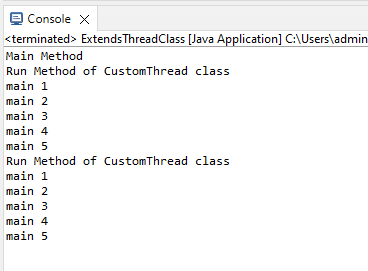
th1.run();

th2.run();

}

}

Output :



* 1. Executing thread using start() for parallel execution

**class** CustomThread **extends** Thread

{

@Override

**public** **void** run() {

//this function will contain the

//logic to be executed in parallel with main()

System.***out***.println("Run Method of CustomThread class");

**for**(**int** i=1;i<=5;i++){

System.***out***.println(Thread.*currentThread*().getName() + " " + i);

}

}

}

**public** **class** ExtendsThreadClass {

**public** **static** **void** main(String[] args) {

System.***out***.println("Main Method");

//Create the object of the thread class

//New stage of the thread

CustomThread th1 = **new** CustomThread();

CustomThread th2 = **new** CustomThread();

**th1.setName("thread1");**

**th2.setName("thread2");**

**//always start your thread using start() method**

**th1.start();**

**th2.start();**

//if we call the run, rather than running

//the run() method parallel to main()

//the run() will be executed sequencially

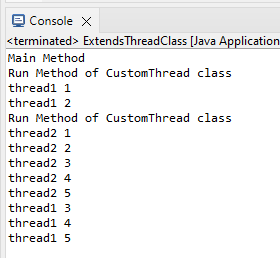
//th1.run();

//th2.run();

}

}

Output :



* 1. Creating thread by implementing Runnable interface

//Runnable interface is a functional interface

//It contain only one method i.e. run()

**class** ThreadImpl **implements** Runnable{

@Override

**public** **void** run() {

**for**(**int** i=1;i<=5;i++){

System.***out***.println(Thread.*currentThread*().getName() + " "+ i);

}

}

}

**public** **class** ImplementsRunnableInterface {

**public** **static** **void** main(String[] args) {

//Create the object of the thread

//will move the thread in new stage

Thread th1 = **new** Thread(**new** ThreadImpl());

th1.setName("thread1 : ");

th1.start();

//Since Runnable interface is a functional interface

//we can provide the implementation of the run() directly without creating class

Thread th2 = **new** Thread(**new** Runnable() {

@Override

**public** **void** run() {

**for**(**int** i=1;i<=5;i++){

System.***out***.println(Thread.*currentThread*().getName() + " " + i);

}

}

});

th2.setName("thread2 : ");

th2.start();

//Since Runnable interface is a functional interface

//we can provide the implementation of the run()

//method using lambda expression

Thread th3 = **new** Thread(() -> {

**for**(**int** i=1;i<=5;i++){

System.***out***.println(Thread.*currentThread*().getName() + " " + i);

}

});

th3.setName("thread3 : ");

th3.start();

}

}