1)Find out the difference between Thread start and thread pool way of implementation for multithreading.

🡪

* Thread start Class should be used when you need the ability to cancel your asynchronous operation. [General Definition: Threading enables your C# program to perform concurrent processing so you can do more than one operation at a time.]
* Thread Pool Class should be used when you need to schedule asynchronous operation and do not need return value or/and ability to cancel your operation [General Definition: A *thread pool* is a collection of threads that can be used to perform several tasks in the background. This leaves the primary thread free to perform other tasks asynchronously.]

Easy Way to Implement The Techniques

There can be many ways in which Threading, ThreadPooling & Delegates can be implemented (depending on programmers), but in this section few of the popular implementation methods (which seemed very easy to me) are mentioned.

* Threading start:

Initialize and call the function, which you want to run using a new thread. Example,

Thread MyNewThread = new Thread (MyFunction);  *// Kick off a new thread*

MyNewThread.Start();  *// running MyFunction()*

* By the way, do not forget to insert (import) the following namespace: 'using System.Threading;'
* Also have a look at this easy to understand article: [Threading in C# [Jump](http://www.albahari.com/threading/)](http://www.albahari.com/threading/)

ThreadPooling:

You can implement ThreadPooling as mentioned in the MSDN article ([ThreadPooling [Jump](http://msdn.microsoft.com/en-us/library/h4732ks0.aspx)](http://msdn.microsoft.com/en-us/library/h4732ks0.aspx" \o "ThreadPooling" \t "_blank)) itself:

public void DoWork()

{

// Queue a task.

System.Threading.ThreadPool.QueueUserWorkItem(

new System.Threading.WaitCallback(SomeLongTask));

// Queue another task.

System.Threading.ThreadPool.QueueUserWorkItem(

new System.Threading.WaitCallback(AnotherLongTask));

}

private void SomeLongTask(Object state)

{

// Insert code to perform a long task.

}

private void AnotherLongTask(Object state)

{

// Insert code to perform a long task.

}

\*IMPLEMENTATION OF MULTITHREADING

🡪

A thread is an independent stream of instructions in a program. A thread is similar to a sequential program. However, a thread itself is not a program, it can't run on its own, instead it runs within a program's context.

The real usage of a thread is not about a single sequential thread, but rather using multiple threads in a single program. Multiple threads running at the same time and performing various tasks is referred as Multithreading. A thread is considered to be a lightweight process because it runs within the context of a program and takes advantage of resources allocated for that program.

With the task manager, you can turn on the Thread column and see the processes and the number of threads for every process. Here, you can notice that only cmd.exe is has a single thread whereas all other applications use multiple threads.