# Programming in C#

**Mục tiêu:**
Sau nội dung thực hành này bạn có khả năng:

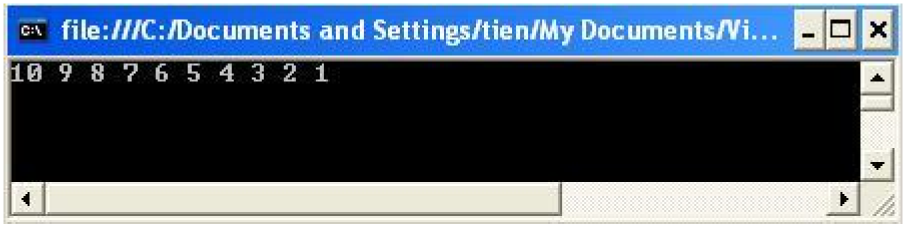
*Làm việc với Thread*

## Phần I: Thực hành từng bước – 45 phút

### Bài tập 1: The creation of threads

```
using System;
using System.Threading;

class MainClass {

      public static void Countdown() {
     for (int i = 10; i > 0; i--) {
      Console.Write(i.ToString() + " ");
    }
  }

  public static void Main() {
    Thread t2 = new Thread(new ThreadStart(Countdown));
    t2.Start();
  }
}
```

Result:

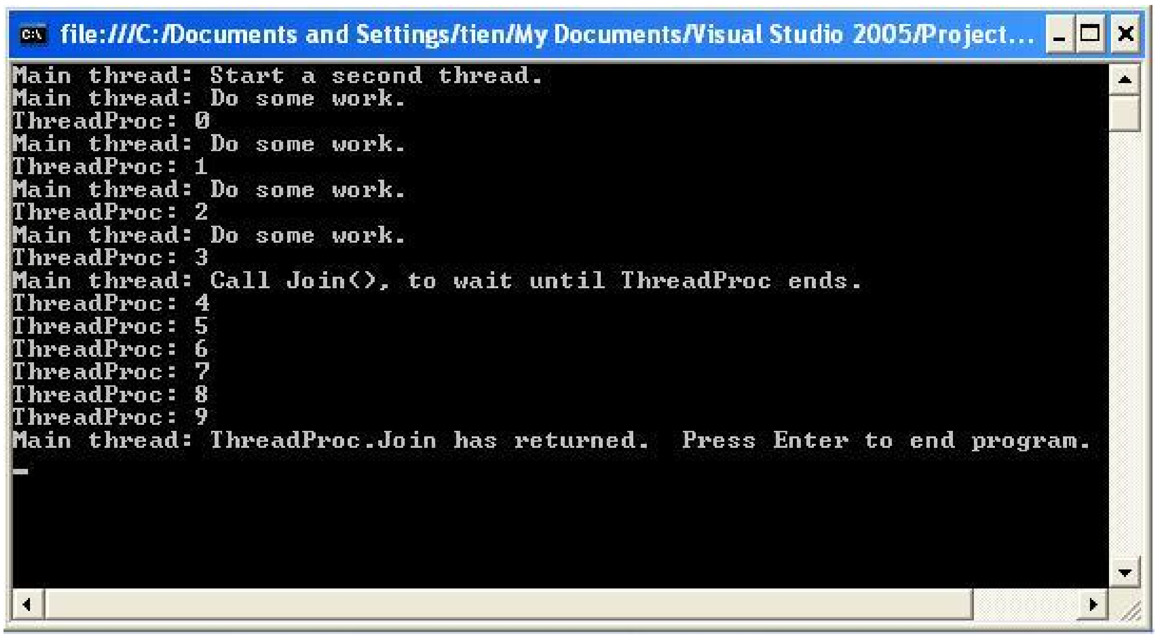


```
10 9 8 7 6 5 4 3 2 1
```

## Bài tập 2: The following code example demonstrates simple threading functionality

```
using System;
using System.Threading;

// Simple threading scenario:  Start a static method running
// on a second thread.
public class ThreadExample{
    // The ThreadProc method is called when the thread starts.
    // It loops ten times, writing to the console and yielding
    // the rest of its time slice each time, and then ends.
    public static void ThreadProc(){
        for (int i = 0; i < 10; i++){
            Console.WriteLine("ThreadProc: {0}", i);
            // Yield the rest of the time slice.
            Thread.Sleep(0);
        }
    }

    public static void Main(){
        Console.WriteLine("Main thread: Start a second thread.");
        // The constructor for the Thread class requires a ThreadStart
        // delegate that represents the method to be executed on the
        // thread.  C# simplifies the creation of this delegate.
        Thread t = new Thread(new ThreadStart(ThreadProc));

        // Start ThreadProc.  Note that on a uniprocessor, the new
        // thread does not get any processor time until the main thread
        // is preempted or yields.  Uncomment the Thread.Sleep that
        // follows t.Start() to see the difference.
        t.Start();
        //Thread.Sleep(0);

        for (int i = 0; i < 4; i++){
            Console.WriteLine("Main thread: Do some work.");
            Thread.Sleep(0);
        }

        Console.WriteLine("Main thread: Call Join(), to wait until
            ThreadProc ends.");
        t.Join();

        Console.WriteLine("Main thread: ThreadProc.Join has returned.
Press Enter to end program.");
        Console.ReadLine();
    }
}
```

Result:



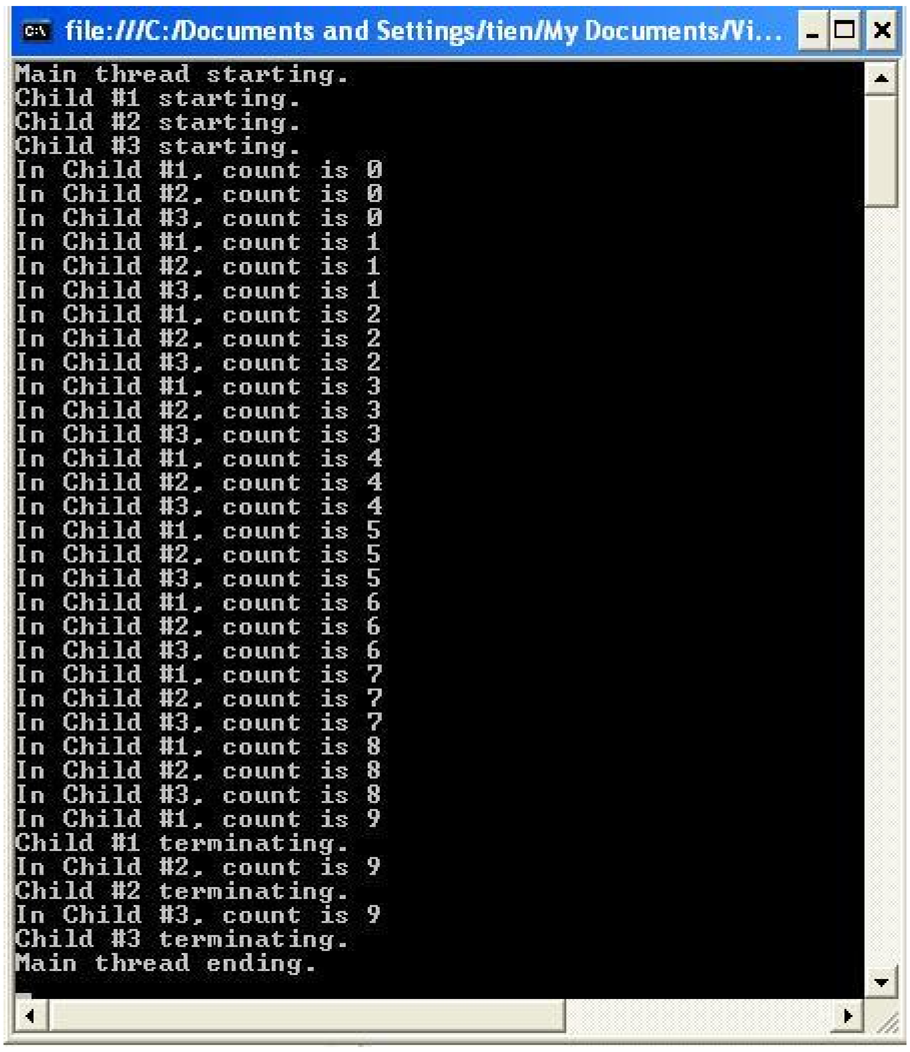
```
file:///C:/Documents and Settings/tien/My Documents/Visual Studio 2005/Project...
Main thread: Start a second thread.
Main thread: Do some work.
ThreadProc: 0
Main thread: Do some work.
ThreadProc: 1
Main thread: Do some work.
ThreadProc: 2
Main thread: Do some work.
ThreadProc: 3
Main thread: Call Join(), to wait until ThreadProc ends.
ThreadProc: 4
ThreadProc: 5
ThreadProc: 6
ThreadProc: 7
ThreadProc: 8
ThreadProc: 9
Main thread: ThreadProc.Join has returned.  Press Enter to end program.
_
```

## Bài tập 3. Create multiple threads of execution

```
using System;
using System.Threading;

class MyThread{
    public int count;
    public Thread thrd;

    public MyThread(string name){
        count = 0;
        thrd = new Thread(this.run);
        thrd.Name = name;
        thrd.Start();
    }

    void run(){
        Console.WriteLine(thrd.Name + " starting.");

        do{
            Thread.Sleep(500); Console.WriteLine("In " + thrd.Name +
                                  ", count is " + count);
            count++;
        } while (count < 10);

        Console.WriteLine(thrd.Name + " terminating.");
    }
```

```
 }


class MainClass{
    public static void Main(){
        Console.WriteLine("Main thread starting.");

        MyThread  mt1  =  new  MyThread("Child
        #1");     MyThread     mt2     =     new
        MyThread("Child  #2");  MyThread  mt3  =
        new MyThread("Child #3");

        Thread.Sleep(10000);

        Console.WriteLine("Main thread ending.");
    }
}
```

Result :

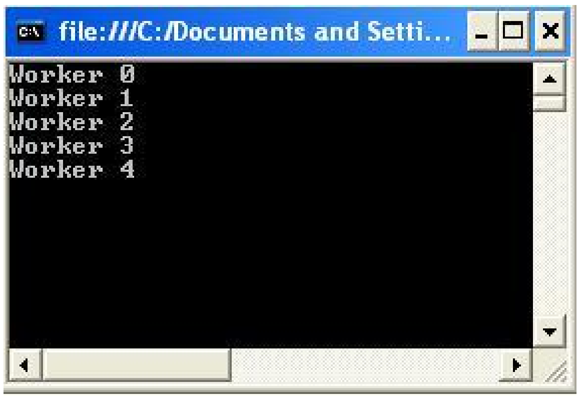


```
Main thread starting.
Child #1 starting.
Child #2 starting.
Child #3 starting.
In Child #1, count is 0
In Child #2, count is 0
In Child #3, count is 0
In Child #1, count is 1
In Child #2, count is 1
In Child #3, count is 1
In Child #1, count is 2
In Child #2, count is 2
In Child #3, count is 2
In Child #1, count is 3
In Child #2, count is 3
In Child #3, count is 3
In Child #1, count is 4
In Child #2, count is 4
In Child #3, count is 4
In Child #1, count is 5
In Child #2, count is 5
In Child #3, count is 5
In Child #1, count is 6
In Child #2, count is 6
In Child #3, count is 6
In Child #1, count is 7
In Child #2, count is 7
In Child #3, count is 7
In Child #1, count is 8
In Child #2, count is 8
In Child #3, count is 8
In Child #1, count is 9
Child #1 terminating.
In Child #2, count is 9
Child #2 terminating.
In Child #3, count is 9
Child #3 terminating.
Main thread ending.
```

## Bài tập 4: Use anonymous delegate as the worker method to create Thread

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Reflection;
using System.Runtime;
using System.Runtime.CompilerServices;
using System.Security;
using System.Text;
using System.Threading;

public class MainClass{
    public static void Main(){
        int threadCount = 5;
        Thread[] threads = new Thread[threadCount];

        for (int i = 0; i < threadCount; i++){
            int idx = i;
            threads[i] = new Thread(delegate()
          { Console.WriteLine("Worker {0}", idx); });
        }

        Array.ForEach(threads, delegate(Thread t) { t.Start(); });
    }
}
```

Result:

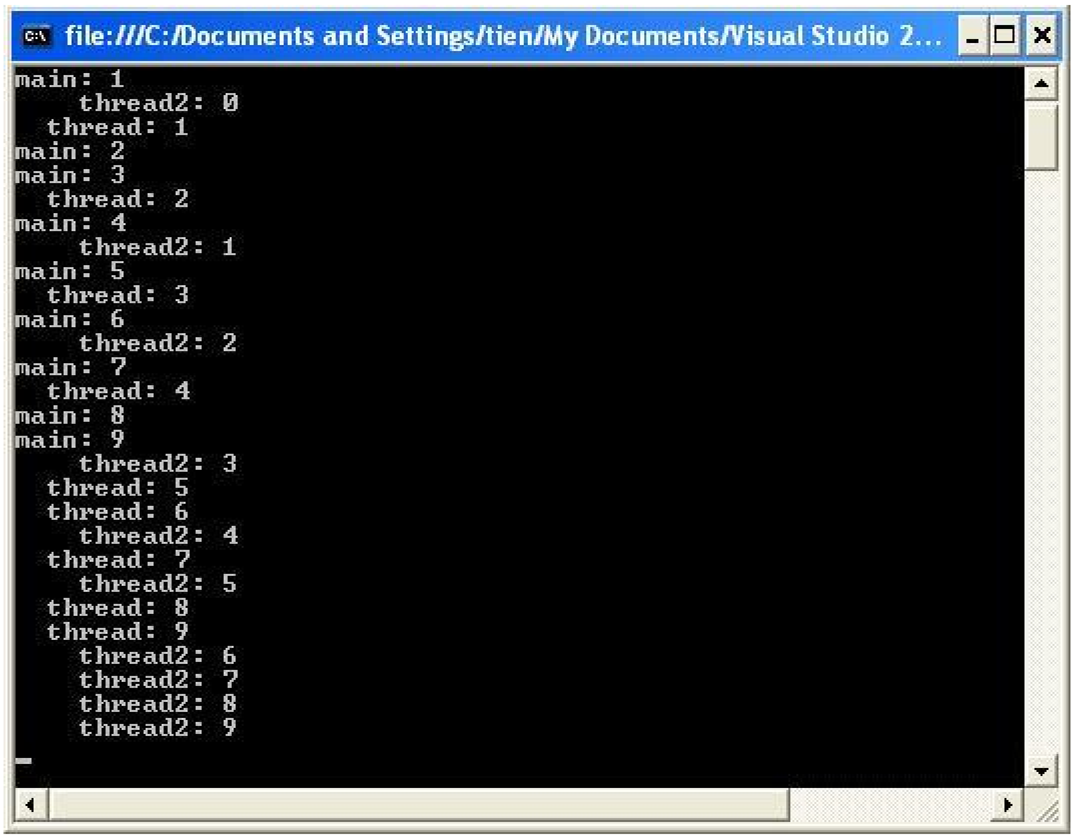


```
Worker 0
Worker 1
Worker 2
Worker 3
Worker 4
```

## Bài tập 5: Thread Pool

```
 using System;
 using System.Threading;

public class ThreadPoolSample{

    public static void Main(){
        ThreadPoolSample tps = new
        ThreadPoolSample(); Console.ReadLine();
    }

    public ThreadPoolSample(){
        int i;

        ThreadPool.QueueUserWorkItem(new
        WaitCallback(Counter));
        ThreadPool.QueueUserWorkItem(new
        WaitCallback(Counter2));

        for (i = 0; i < 10; i++){
            Console.WriteLine("main: {0}", i);
            Thread.Sleep(1000);
        }
    }

    void Counter(object state){
        int i;
        for (i = 0; i < 10; i++){
            Console.WriteLine("  thread: {0}", i);
            Thread.Sleep(2000);
        }
    }

    void Counter2(object state){
        int i;
        for (i = 0; i < 10; i++){
            Console.WriteLine("    thread2: {0}", i);
            Thread.Sleep(3000);
        }
    }
}
```

The result:

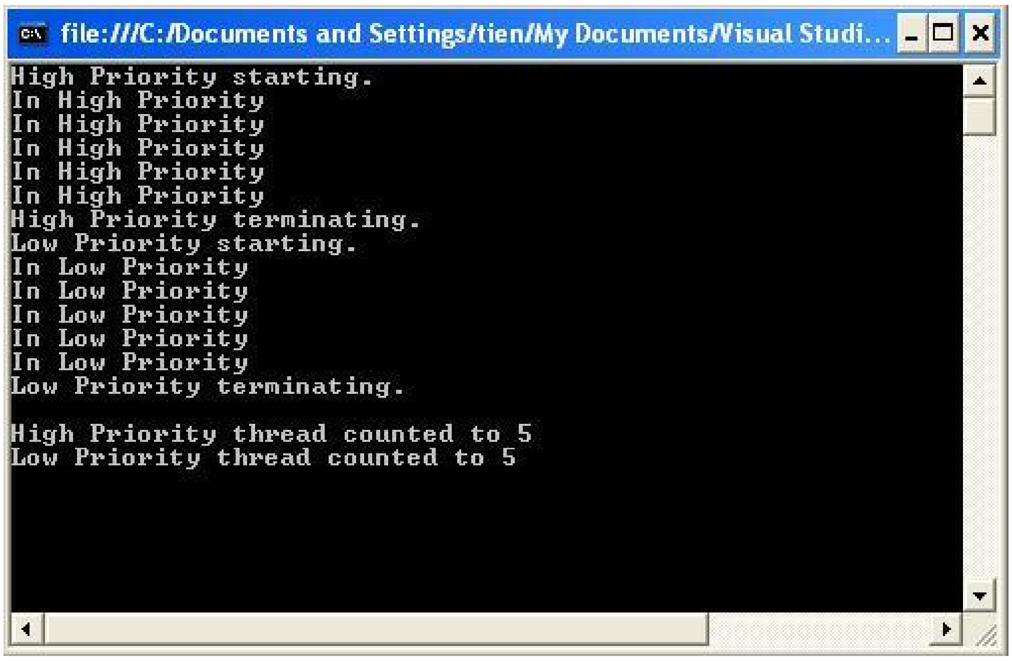

file:///C:/Documents and Settings/tien/My Documents/Visual Studio 2...

```
main: 1
    thread2: 0
  thread: 1
main: 2
main: 3
  thread: 2
main: 4
    thread2: 1
main: 5
  thread: 3
main: 6
    thread2: 2
main: 7
  thread: 4
main: 8
main: 9
    thread2: 3
  thread: 5
  thread: 6
    thread2: 4
  thread: 7
    thread2: 5
  thread: 8
  thread: 9
    thread2: 6
    thread2: 7
    thread2: 8
    thread2: 9
```

## Bài tập 6 : Thread priorities

```
using System;
using System.Threading;

class MyThread{
    public int count;
    public Thread thrd;

    public MyThread(string name){
        count = 0;
        thrd = new Thread(this.run);
        thrd.Name = name;
    }

    void run(){
        Console.WriteLine(thrd.Name + " starting.");
        do
        {
            count++;

            Console.WriteLine("In " + thrd.Name);
        } while (count < 5);

        Console.WriteLine(thrd.Name + " terminating.");
    }
}

class PriorityDemo{
    public static void Main(){
        MyThread mt1 = new MyThread("High Priority");
        MyThread mt2 = new MyThread("Low Priority");

        mt1.thrd.Priority = ThreadPriority.AboveNormal;
        mt2.thrd.Priority = ThreadPriority.BelowNormal;

        mt1.thrd.Start();
        mt2.thrd.Start();

        mt1.thrd.Join();
        mt2.thrd.Join();

        Console.WriteLine();
        Console.WriteLine(mt1.thrd.Name + " thread counted to " +
mt1.count);
        Console.WriteLine(mt2.thrd.Name + " thread counted to " +
mt2.count);
        Console.ReadLine();
    }
}
```

The result:



```
High Priority starting.
In High Priority
In High Priority
In High Priority
In High Priority
In High Priority
High Priority terminating.
Low Priority starting.
In Low Priority
In Low Priority
In Low Priority
In Low Priority
In Low Priority
Low Priority terminating.

High Priority thread counted to 5
Low Priority thread counted to 5
```

## Bài tập 7: Use .NET components in COM

1. The first step is to create a .NET library. Start Microsoft Visual Studio.NET 2005.
2. Click on the 'New Project' button from the startup window. Select 'Visual C# Projects' from the 'Project Types' section,'Class Library' from the 'Templates' section and specify the project name as 'DotNetCOMDemo'.
3. Click 'OK' to proceed.
4. Rename the class file to Hospital.cs.
5. Replace the code in the class as shown below.

```
using System;

class Hospital{
    public abstract class HospitalEmployee{
        private string name;
        private int id;
        private string address;

        public string Name{
            get { return name; }
            set { name = value; }
        }

        public int ID{
            get { return id; }
            set { id = value; }
        }
        public string Addr{
            get { return address; }
            set { address = value; }
        }
        public class Doctor : HospitalEmployee{
            public Doctor(){ }
        }


        public class Nurse : HospitalEmployee{
            public Nurse(){

            }
        }
    }
}
```

6. Compile the code to generate a library.
7. Export a type library for COM clients from this assembly using the TlbExp tool.
8. The COM client (which will be reated later )can use this exported type library unless it is registered. A special registration tool is provided by .NET for this purpose named
'Regasm'. Register the library using the Regasm tool.
9. Write a simple program in Visual Studio 6 to use this .NET component after registering it to Windows registry.

## Bài tập 8: Using a COM component from .NET.

**1.**Assume that the COM component named Calculator having class
CalcClass has already been created and compiled into a DLL.
**2.**Create a Console base C# application uing VS.NET and name it
as Q1.
**3.**Rename Class1.cs to COMDemo.cs
**4.**Copy the COM DLL to the Debug folder of Q1.
**5.** Merely copying it ,will not ensure that it can be used within a .NET application. It needs to be imported into the application. The steps to import the COM DLL so that it can be used and tested with the .NET application are listed:
**6.**Register the COM component using the Regsvr32.exe tool
**7.**Using TblImp to import the COM component
**8.**Open the VS.Net IDE and add a reference to the imported library CalcNet.dll.
**9.**Open the class COMDemo.cs. Replace the code in the Main method with the following code:

```
 public static void Main(string[] args){
     CalcNet.CalcClass c= new CalcNet.CalClass();
     Console.WriteLine("Please enter a number: ");
     short num1 = Convert.ToUInt16(Console.ReadLine());
      Console.WriteLine("Please enter another number:");
      short num2 = Convert.ToUInt16(Console.ReadLine());
      int res = c.Add(num1,num2); Console.WriteLine("The sum
      of the two numbers is {0}",res);
      res = c.Subtract(num1,num2);
      Console.WriteLine("The difference between the two numbers is
      {0}",res);
      Console.ReadLine();
}
```

```
10. Save and Build the application.
```

## Phần II:Tự thực hành – 60 phút

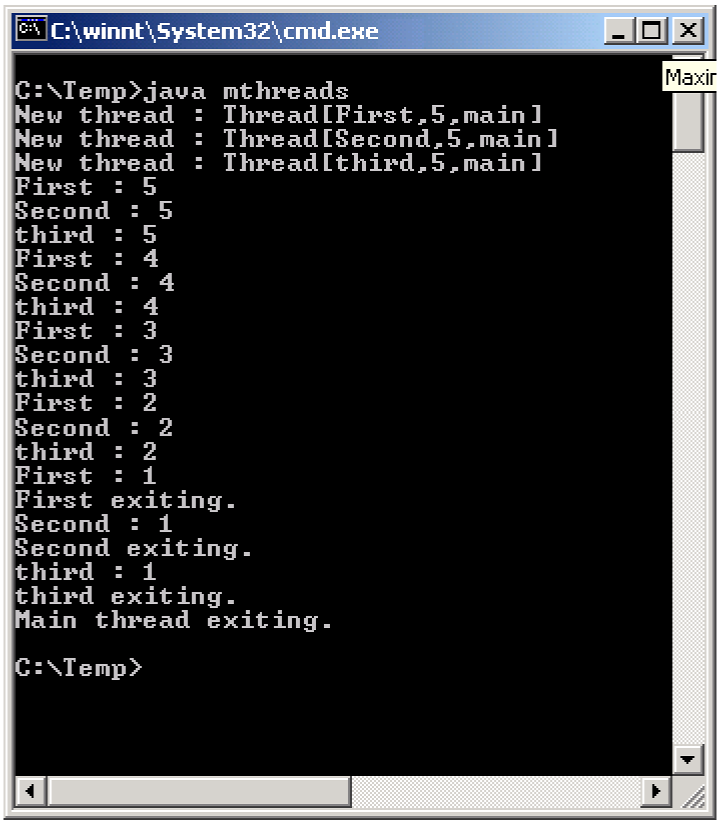
### Bài tập 1. Sử dụng DllImport

Tạo một chương trình Console C# dựa trên chương trình ứng dụng mà bao gồm một phương thức không thể phá hủy **`GetComputerName()`** từ kernel32.dll sử dụng các dịch vụ gọi nền tảng và hiển thị tên máy tính

Create a C# console based application that invokes the unmanaged method GetComputerName() from the kernel32.dll using platform invocation services and displays the computer name.

**Hint:** Use the DllImport attribute.

### Bài tập 2. Viết chương trình chạy thử như sau:



```
C:\Temp>java mthreads
New thread : Thread[First,5,main]
New thread : Thread[Second,5,main]
New thread : Thread[third,5,main]
First : 5
Second : 5
third : 5
First : 4
Second : 4
third : 4
First : 3
Second : 3
third : 3
First : 2
Second : 2
third : 2
First : 1
First exiting.
Second : 1
Second exiting.
third : 1
third exiting.
Main thread exiting.

C:\Temp>
```

Create three threads and the main thread. Execute each thread as simultaneous tasks. Display information when exiting each thread.

## Bài tập 3.3. Tạo một Thread mới in một giá trị trong một khoảng thời gian. Thông tin về một luồng bao gồm: message, time duration, priority.

```
MESSAGE BOARD
=================
=== Number of
messages: 2
Message 1: multithreading
Timeout: 1000
Priority: high
Message 2: multitasking
Timeout: 2000
Priority: medium

Result:
Multithreading


 Multithreading
 Multitasking
 Multithreading
 Multithreading
 Multitasking
 …..
```