

**CHƯƠNG 3B**

# **LẬP TRÌNH ĐA LUỒNG (MULTITHREADING)**

**LẬP TRÌNH MẠNG CĂN BẢN**



# NỘI DUNG

- Khái niệm về Đa luồng
- Lợi ích của Lập trình Đa luồng
- Đặc điểm, cách thức khai báo Đa luồng trong C#
- Demo

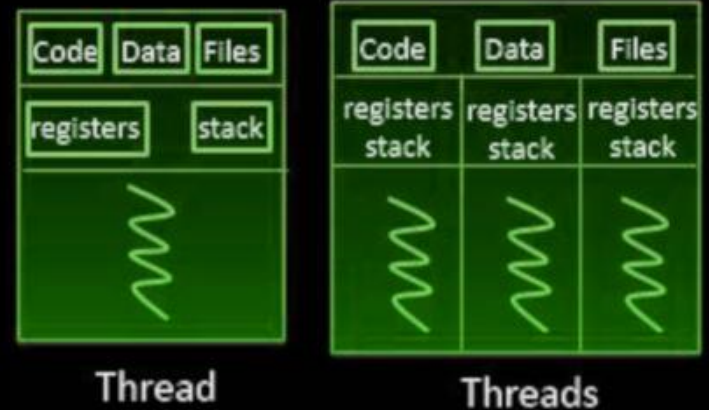
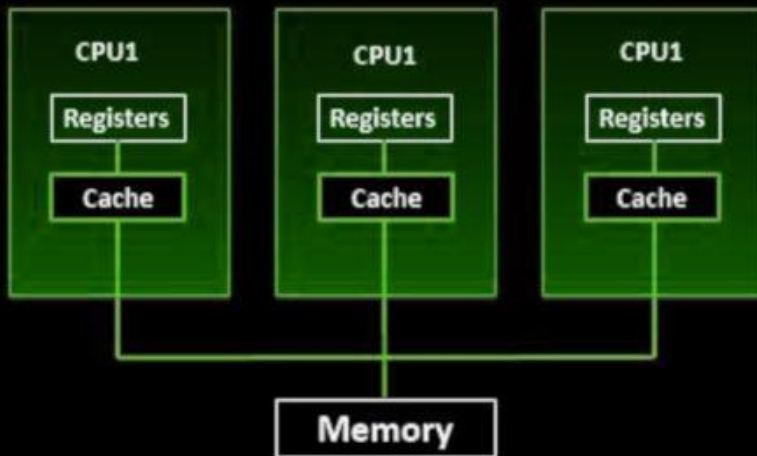


# KHÁI NIỆM VỀ THREAD

- Thread là luồng (còn gọi là tiểu trình)
- Cho phép chương trình thực hiện đồng thời nhiều tác vụ
- Giúp quá trình tương tác với người dùng không bị gián đoạn
- Là kĩ thuật không thể thiếu trong các ứng dụng mạng

# THREAD VÀ PROCESS

## Multiprocessing vs Multithreading



```
class Program
{
    static void Main()
    {
        Thread t = new Thread(new ThreadStart(MethodA));
        t.Start();
        MethodB();
    }

    static void MethodA()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("0");
    }
    static void MethodB()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("1");
    }
}
```

## THREAD – VÍ DỤ

[illegible]

Press any key to continue ...

# TRUYỀN THAM SỐ CHO THREAD

- Chỉ nhận một tham số kiểu object
- Trong phương thức callback, cần phải ép kiểu để sử dụng được đúng kiểu dữ liệu của tham số

```

namespace ThreadExample
{
    class Student
    {
        public string Name { get; set; }
        public DateTime BirthDay { get; set; }
    }

    class Program
    {
        static void Main()
        {
            Thread t1 = new Thread(Print);

            t1.Start(new Student() { Name = "Yin", BirthDay = new DateTime(1989, 10, 17) });

            Console.ReadKey();
        }

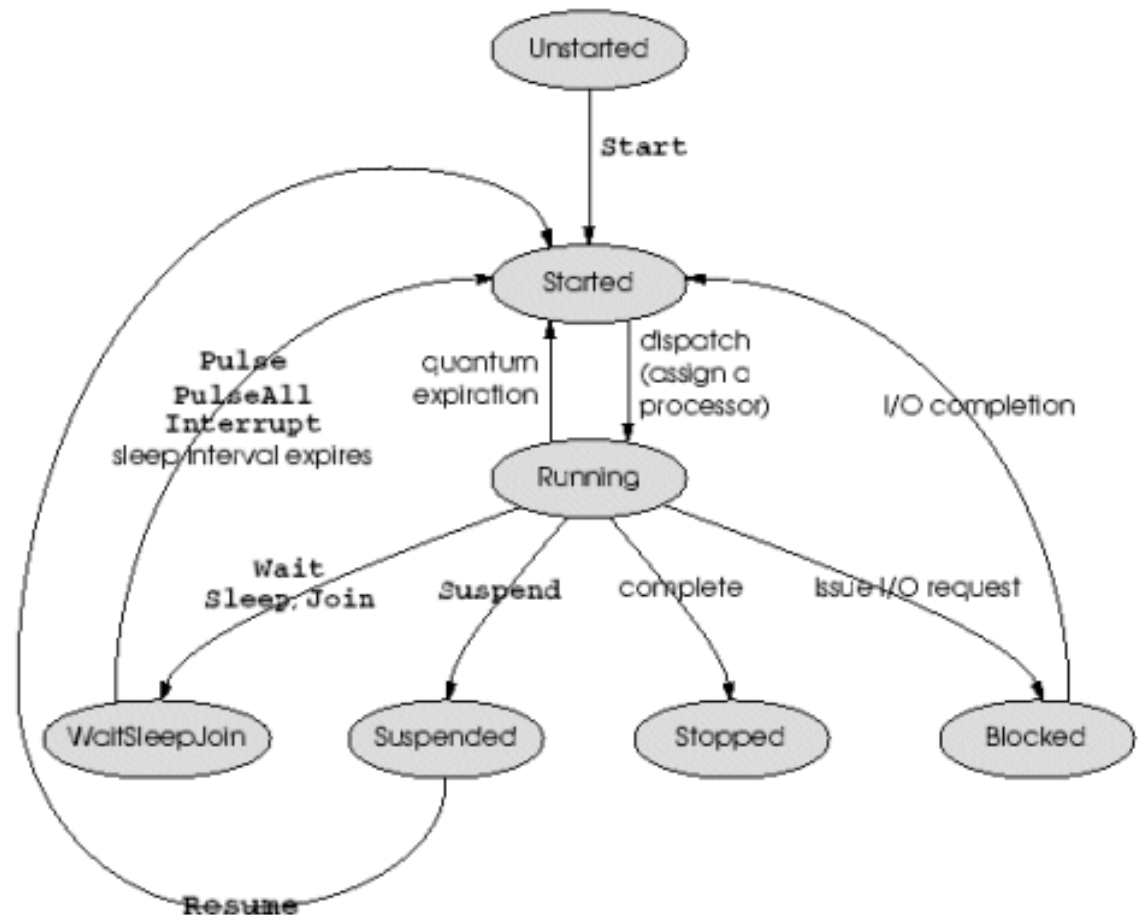
        static void Print(object obj)
        {
            Student st = (Student)obj;
            Console.Write(st.Name + "\t" + st.BirthDay.ToShortDateString());
        }
    }
}

```



# VÒNG ĐỜI CỦA THREAD

- ThreadState:
  - Unstarted
  - Running
  - NotRunning
  - Stopped



# THREADPRIORITY

- ThreadPriority:
  - Xác định mức độ ưu tiên sẽ được thực thi so với các thread khác
  - Mỗi thread khi được tạo ra mang giá trị là Normal
  - Các giá trị gồm: Lowest, BelowNormal, Normal, AboveNormal và Highest

# CÁC PHƯƠNG THỨC THÔNG DỤNG CỦA THREAD

- **Abort()**: hệ thống sẽ đưa ra một ngoại lệ ThreadAbortException để kết thúc thread, ThreadState sẽ chuyển sang giá trị Stopped.
- **await()**: tạm dừng việc thực thi của Thread vô thời hạn đến khi được yêu cầu chạy tiếp tục với phương thức Resume()
- **Sleep()**: dừng thread **hiện tại** trong một khoảng thời gian (milisecond, khi đó thread sẽ chuyển sang trạng thái WaitSleepJoin)

```
class Program
{
    static void Main()
    {
        Thread t = new Thread(MethodA);
        t.Start();
        MethodB();
    }

    static void MethodA()
    {
        Thread.Sleep(500); // sleep for 500 milliseconds
        for (int i = 0; i < 100; i++)
            Console.Write("0");
    }

    static void MethodB()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("1");
    }
}
```

## KẾT QUẢ

[illegible]

# CÁC PHƯƠNG THỨC THÔNG DỤNG CỦA THREAD

- **Join()**: phương thức dùng trong trường hợp muốn thực hiện một tác vụ nào đó sau khi thread đã kết thúc. Phương thức này chỉ dùng sau khi bạn đã chạy Thread. Các tác vụ nằm phía dưới lệnh gọi Join() của một Thread chỉ được thực thi sau khi Thread đó đã hoàn tất

```
class Program
{
    static void Main()
    {
        Thread t1 = new Thread(MethodA);
        Thread t2 = new Thread(MethodB);
        Thread t3 = new Thread(MethodC);

        t1.Start();
        t2.Start();

        t2.Join();

        t3.Start();
    }

    static void MethodA()
    {
        for (int i = 0; i < 100; i++) Console.Write("0");
    }
    static void MethodB()
    {
        for (int i = 0; i < 100; i++) Console.Write("1");
    }
    static void MethodC()
    {
        for (int i = 0; i < 100; i++) Console.Write("2");
    }
}
```

## KẾT QUẢ

[illegible]



# BÀI TẬP

- Nhập vào 1 số  $n$  (số nguyên dương)
- Thêm 2 cách thức đếm, sao cho có thể đếm lúc nào cũng được (Song song, hoặc 1 trước 1 sau)
- Nút sẽ bị tắt cho đến khi đếm xong
- 2 cách thức đếm:
  - Cách 1: đếm tăng dần 1 đơn vị
  - Cách 2: đếm tăng dần 2 đơn vị

# BÀI TẬP (GIAO DIỆN MINH HỌA)

The image shows a Windows application window titled "Form1". Inside the window, there is a text label "Nhập vào 1 số n:" followed by a text input field. Below this, there are two columns of controls. The left column has a label "Count to Number (1)", a large red "0", and a button labeled "COUNT 1". The right column has a label "Count to Number (2)", a large red "0", and a button labeled "COUNT 2".

Form1

Nhập vào 1 số n:

Count to Number (1)

0

COUNT 1

Count to Number (2)

0

COUNT 2

# BÀI TẬP (GIAO DIỆN MINH HỌA)

Form1

Nhập vào 1 số n:

Count to Number (1)

87

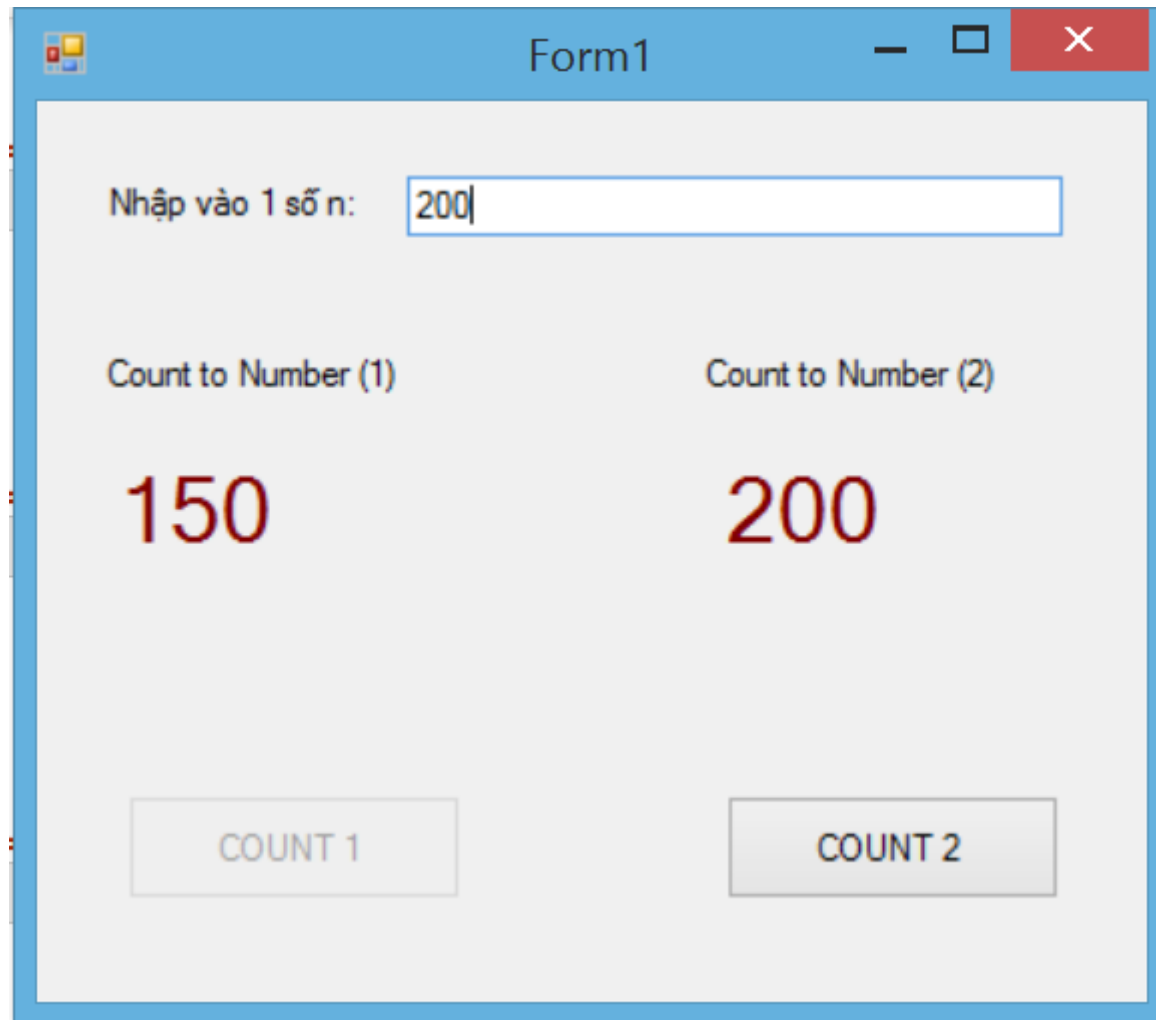
COUNT 1

Count to Number (2)

162

COUNT 2

# BÀI TẬP (GIAO DIỆN MINH HỌA)



The image shows a Windows application window titled "Form1". Inside the window, there is a text label "Nhập vào 1 số n:" followed by a text box containing the number "200". Below this, there are two columns of text. The left column is headed "Count to Number (1)" and displays the number "150" in a large, bold, red font. Below this is a button labeled "COUNT 1". The right column is headed "Count to Number (2)" and displays the number "200" in a large, bold, red font. Below this is a button labeled "COUNT 2".

Count to Number (1)	Count to Number (2)
150	200
COUNT 1	COUNT 2

# BÀI TẬP (GỢI Ý)

- Sử dụng

`CheckForIllegalCrossThreadCalls = false;`

Để fix lỗi Window Form báo lỗi chạy Thread