A One-Stop Solution To C# Threads With A Guide And Examples

Table of Contents

[What Is a C# Threads Class?](https://www.simplilearn.com/tutorials/c-sharp-tutorial/what-is-c-sharp-threads#what_is_a_c_threads_class)

[Thread Life-Cycle](https://www.simplilearn.com/tutorials/c-sharp-tutorial/what-is-c-sharp-threads#thread_lifecycle)

[What Is a Single-Threaded Model?](https://www.simplilearn.com/tutorials/c-sharp-tutorial/what-is-c-sharp-threads#what_is_a_singlethreaded_model)

[Implementation of Single-Threaded Model](https://www.simplilearn.com/tutorials/c-sharp-tutorial/what-is-c-sharp-threads#implementation_of_singlethreaded_model)

[What Is a Multi-Threaded Model?](https://www.simplilearn.com/tutorials/c-sharp-tutorial/what-is-c-sharp-threads#what_is_a_multithreaded_model)

Multitasking refers to the act of performing several tasks or activities at the same time. As an example of multitasking, the Windows operating system may run many applications concurrently, such as Google Chrome, Notepad, and MS Office. The operating system employs the concept of a process to run all of these programs simultaneously.

What Is a C# Threads Class?

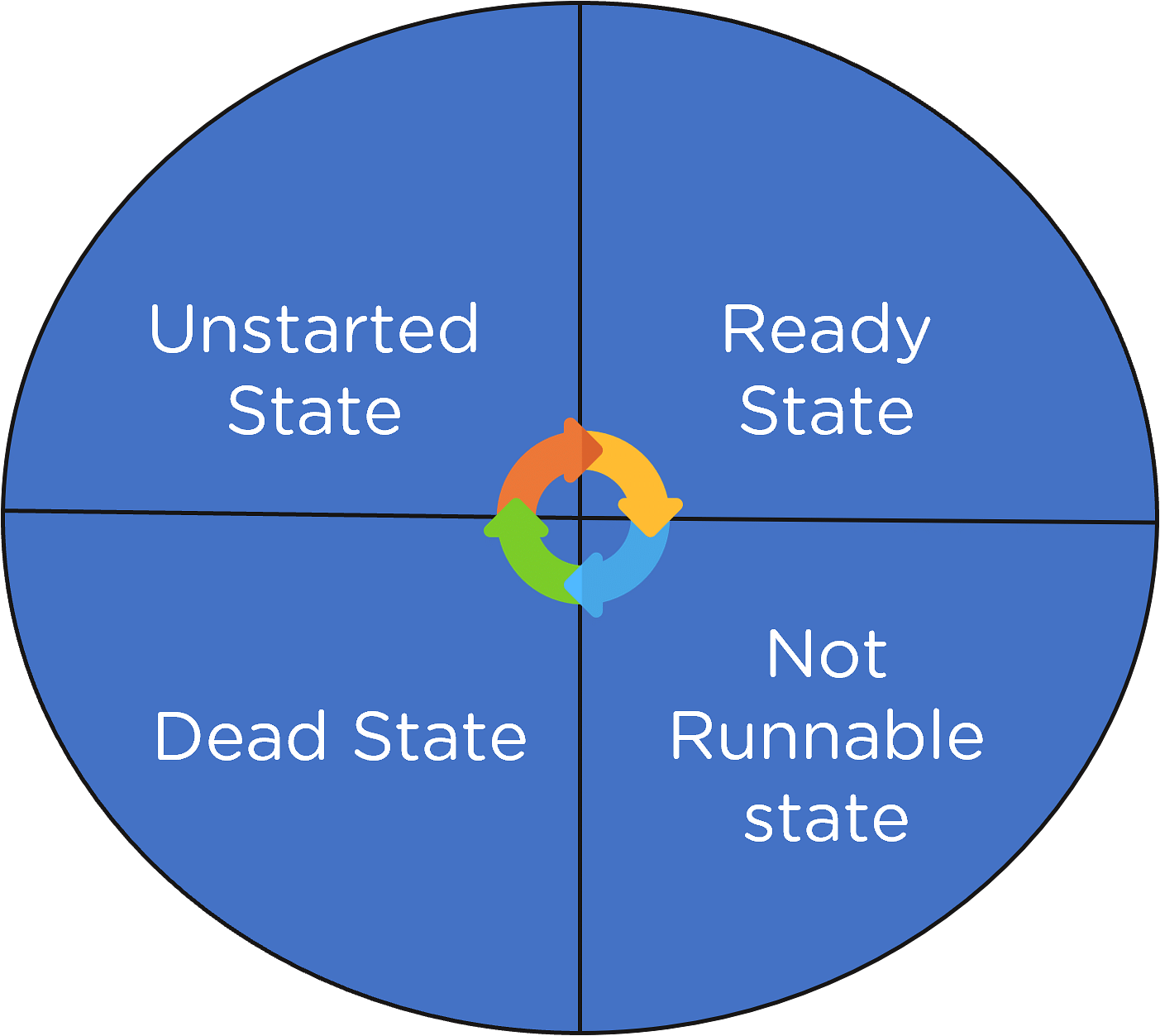


The term "thread" refers to a program's execution path. The control flow of each thread is unique. Threading decreases the number of wasted CPU cycles and enhances the overall performance of a program. New threads are created with the help of the thread class.

The thread class is used to specify a thread's priority. The current status of a thread is also displayed.

You can now go on to the next chapter to learn about the Thread Life-Cycle now that you have a basic knowledge of the definition of [C#](https://www.simplilearn.com/c-sharp-programming-for-beginners-article) Threads.

Thread Life-Cycle



Generally, a thread's life cycle starts with the instantiation of an object of the "System.Threading" Thread class ends whenever the thread is suspended or completes its processing.

There are four states of a Thread Life-Cycle, they are as follows:

* Unstarted State

It happens when a thread object is created, but the Start method is not called.

* Ready State

The Ready state indicates that the thread is ready for implementation but is waiting for a CPU loop.

* Not Runnable State

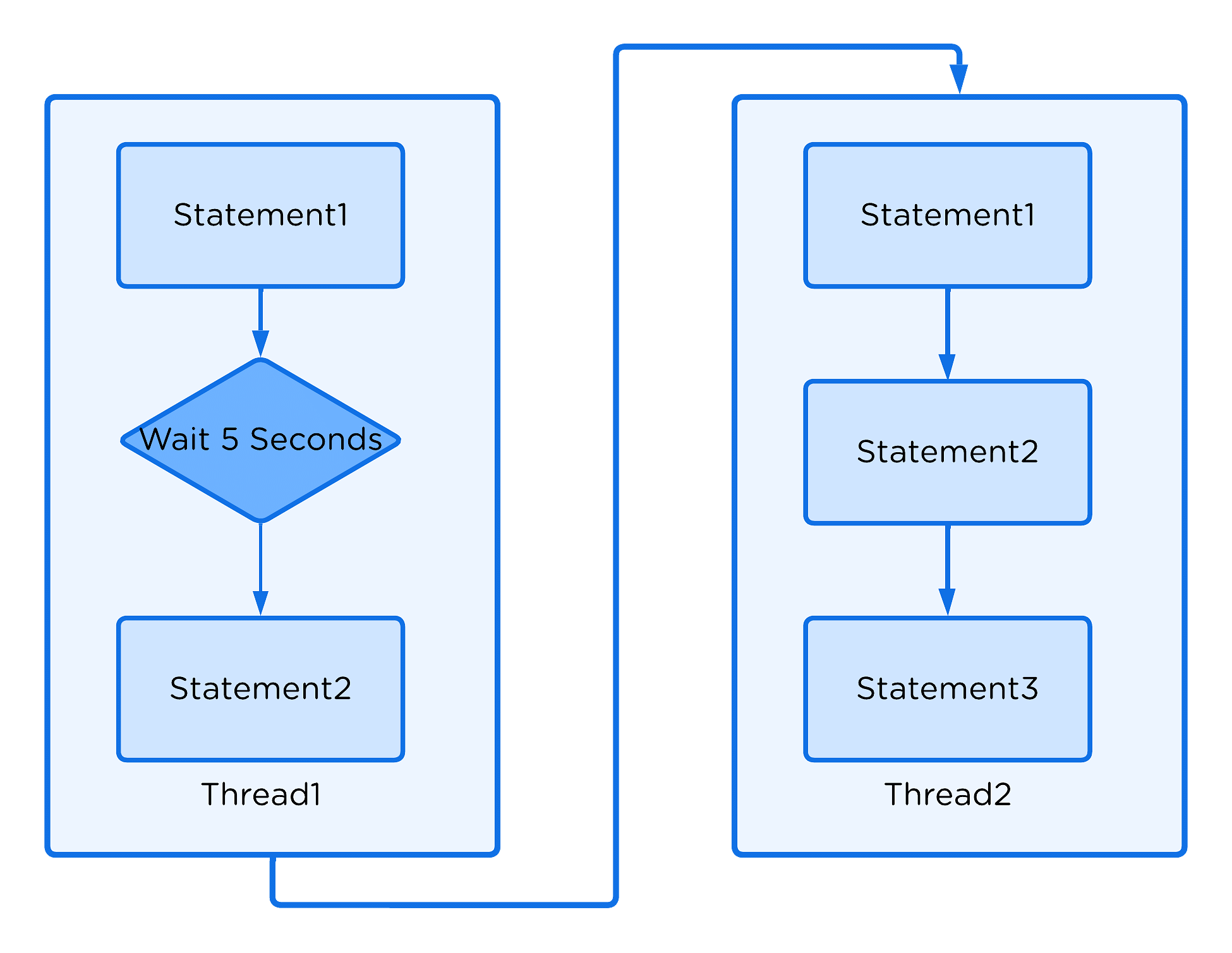
This is when you have a "not runnable state" and a thread is not executable because of the following:

* The sleep method has been invoked
* The wait method has been invoked
* I/O activities are causing a bottleneck
* Dead State

It is the state wherein the thread's implementation has finished or has been stopped.



What Is a Single-Threaded Model?



The Main Thread is the primary thread in any program that executes the logic of the program. As a result, by nature, all programs and applications are single-threaded. This single-threaded design has a drawback. The single thread synchronizes the execution of the program's processes, meaning that each one is completed sequentially. Because of the delay in processing the first step, the subsequent process is slower.

So far, you have learned about the Threads and the Single-threaded model. Now you will explore the practical implementation of the Single-threaded model.

Implementation of Single-Threaded Model

Now, open the visual studio [code](https://www.simplilearn.com/free-and-low-cost-online-resources-for-practicing-code-article) and write a code to demonstrate the single-threaded model.

Code:

using System;

using System.Threading;

namespace Threaddemo

{

    public class mainthread1 {

    // static method one

    public static void thrd1()

    {

        // It prints numbers from 0 to 10

        for (int t = 0; t <= 10; t++) {

            Console.WriteLine("Thread1 is : {0}", t);

            // When the value of t is equal to

            // 5 then this method sleeps for

            // 5 seconds and after 5 seconds

            // it resumes its working

            if (t == 5) {

                Thread.Sleep(5000);

            }

        }

    }

    // static method two

    public static void thrd2()

    {

        // It prints numbers from 0 to 10

        for (int p = 0; p <= 10; p++) {

            Console.WriteLine("Thread2 is : {0}", p);

        }

    }

}

// Driver Class

public class mainthread2 {

    // Main Method

    static public void Main()

    {

        // Calling static methods

        mainthread1.thrd1();

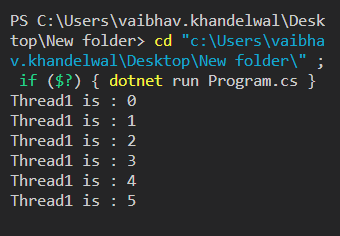
        mainthread1.thrd2();

    }

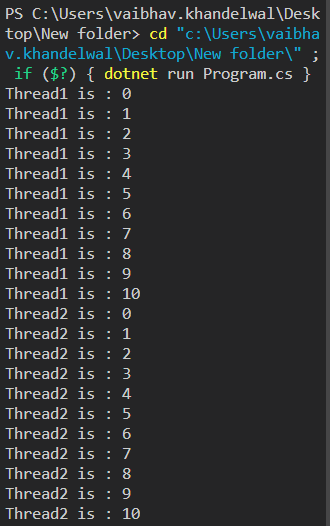
}

}

Now, go ahead and run it.



Now it went to sleep for 5 seconds after t=5.



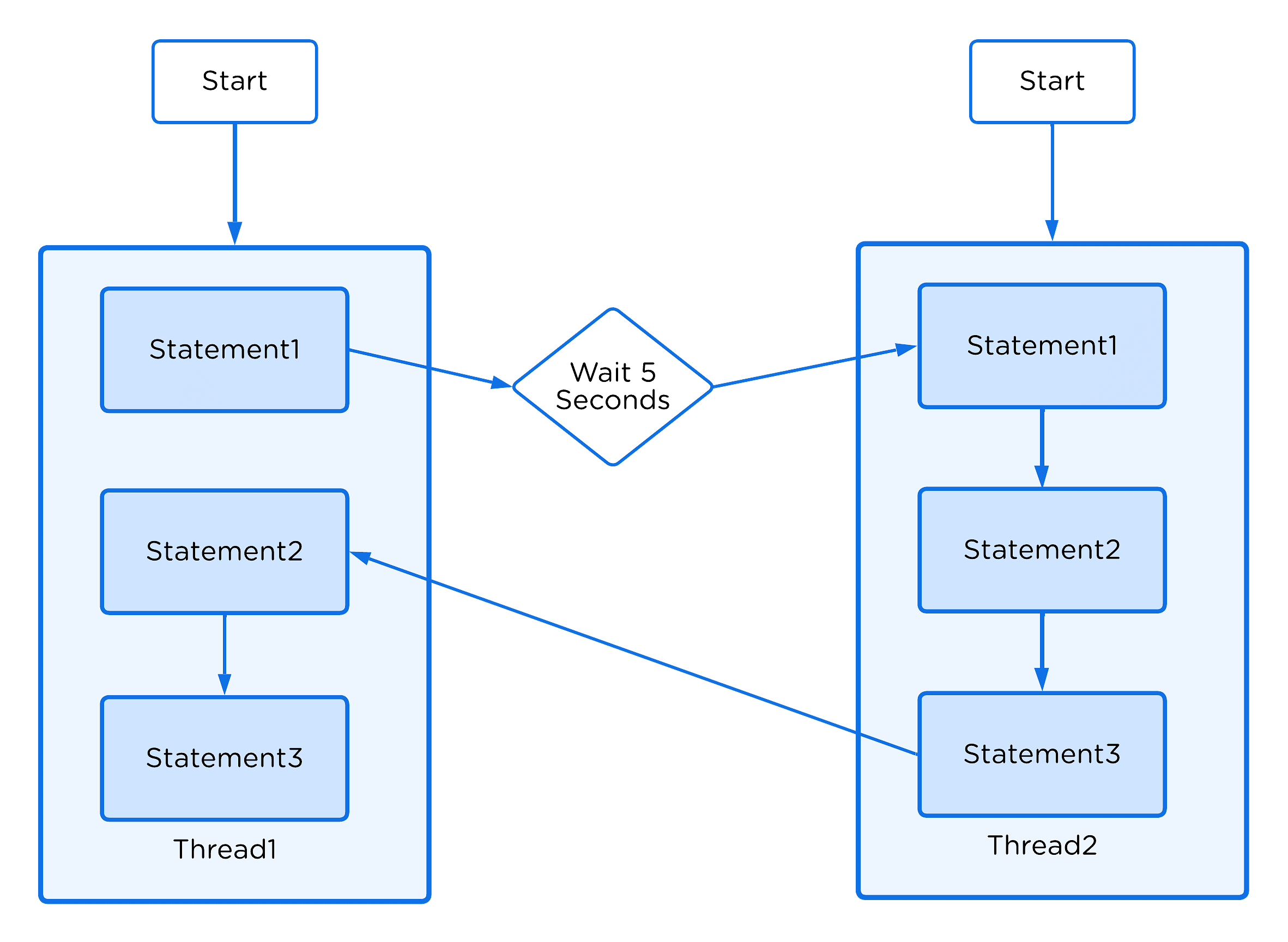
Now let's discuss the Multi-threaded model to overcome the drawbacks of the single-threaded model.

Unleash Your Career as a Full Stack Developer!

Full Stack Developer - MERN Stack[EXPLORE COURSE](https://www.simplilearn.com/full-stack-developer-course-mern-certification-training?source=GhPreviewCTABanner)



What Is a Multi-Threaded Model?



Multithreading is the utilization of several threads within a single task. Each thread has a distinct purpose. Therefore, the key benefit of [Multithreading](https://www.simplilearn.com/tutorials/java-tutorial/multithreading-in-java) is that it functions simultaneously, allowing several tasks to be executed simultaneously. Additionally, Multithreading enhances CPU resource use.

So far, you learned about the Single-threaded and Multi-threaded models. Now you will look at the practical implementation of the Multi-threaded model.

Implementation of Multi-Threaded Model

Now, it’s time to open your visual studio code and write a code to demonstrate the single-threaded model.

Code:

// C# program to illustrate the

// concept of Multithreading

using System;

using System.Threading;

public class mainthread {

    // static method one

    public static void thrd1()

    {

        // It prints numbers from 0 to 10

        for (int x = 0; x <= 10; x++) {

            Console.WriteLine("Thread1 is : {0}", x);

            // When the x's value is equal to 5 then

            // this method sleeps for 5 seconds

            if (x == 5) {

                Thread.Sleep(5000);

            }

        }

    }

    // static method two

    public static void thrd2()

    {

        // It prints numbers from 0 to 10

        for (int y = 0; y <= 10; y++) {

            Console.WriteLine("Thread2 is : {0}", y);

        }

    }

    // Main Method

    static public void Main()

    {

        // Creating and initializing threads

        Thread t1 = new Thread(thrd1);

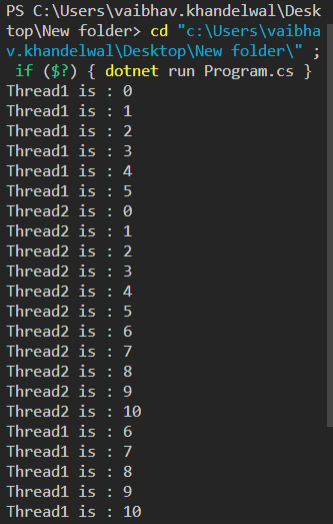
        Thread t2 = new Thread(thrd2);

        t1.Start();

        t2.Start();

    }

}



By now, you must have a good knowledge and command of the technical aspects of C# Threads and multithreading.