야매로 서버 개발자 되는 법 4강

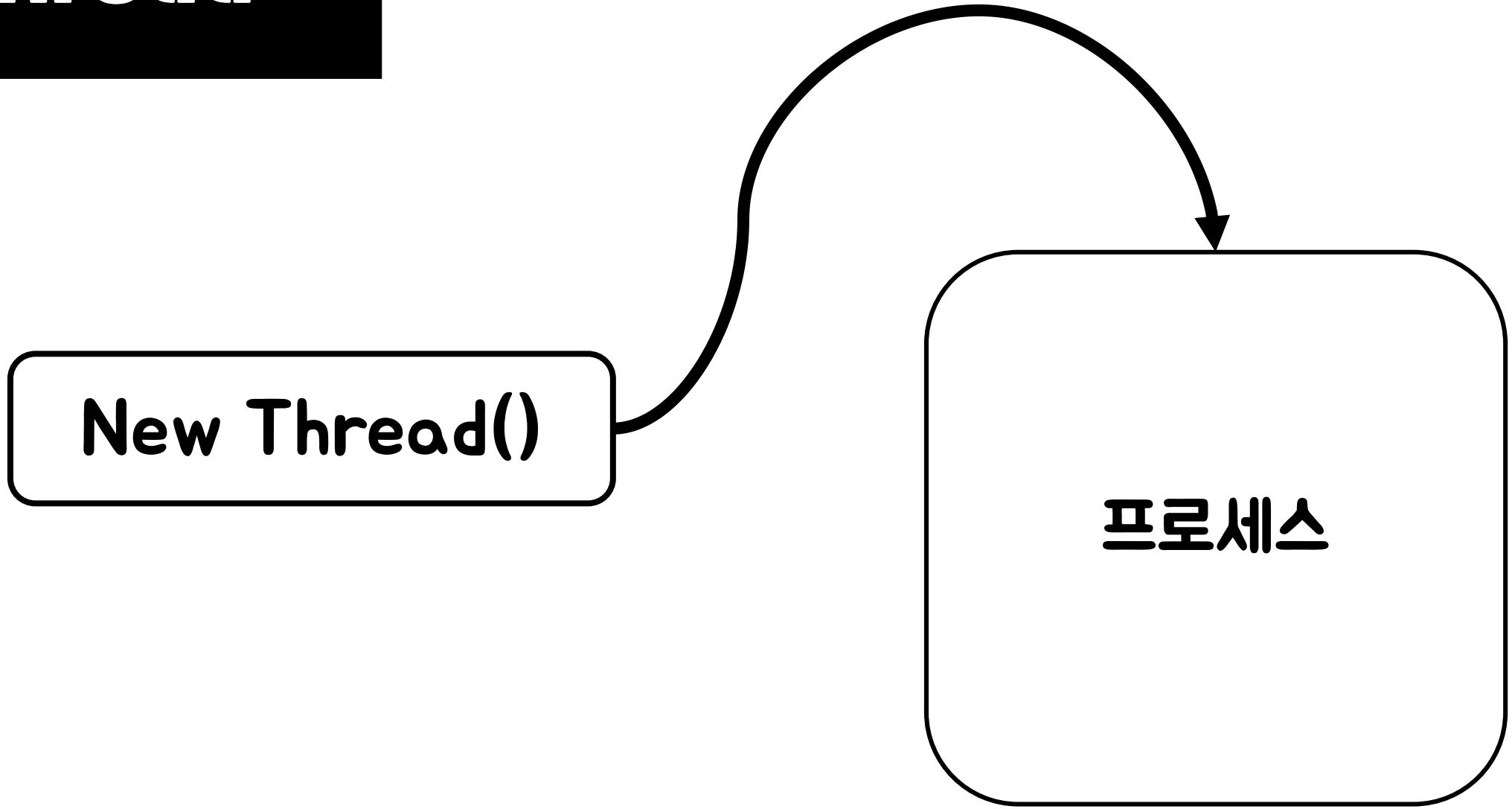
멀티스레딩 기초

스레드 사용 방법

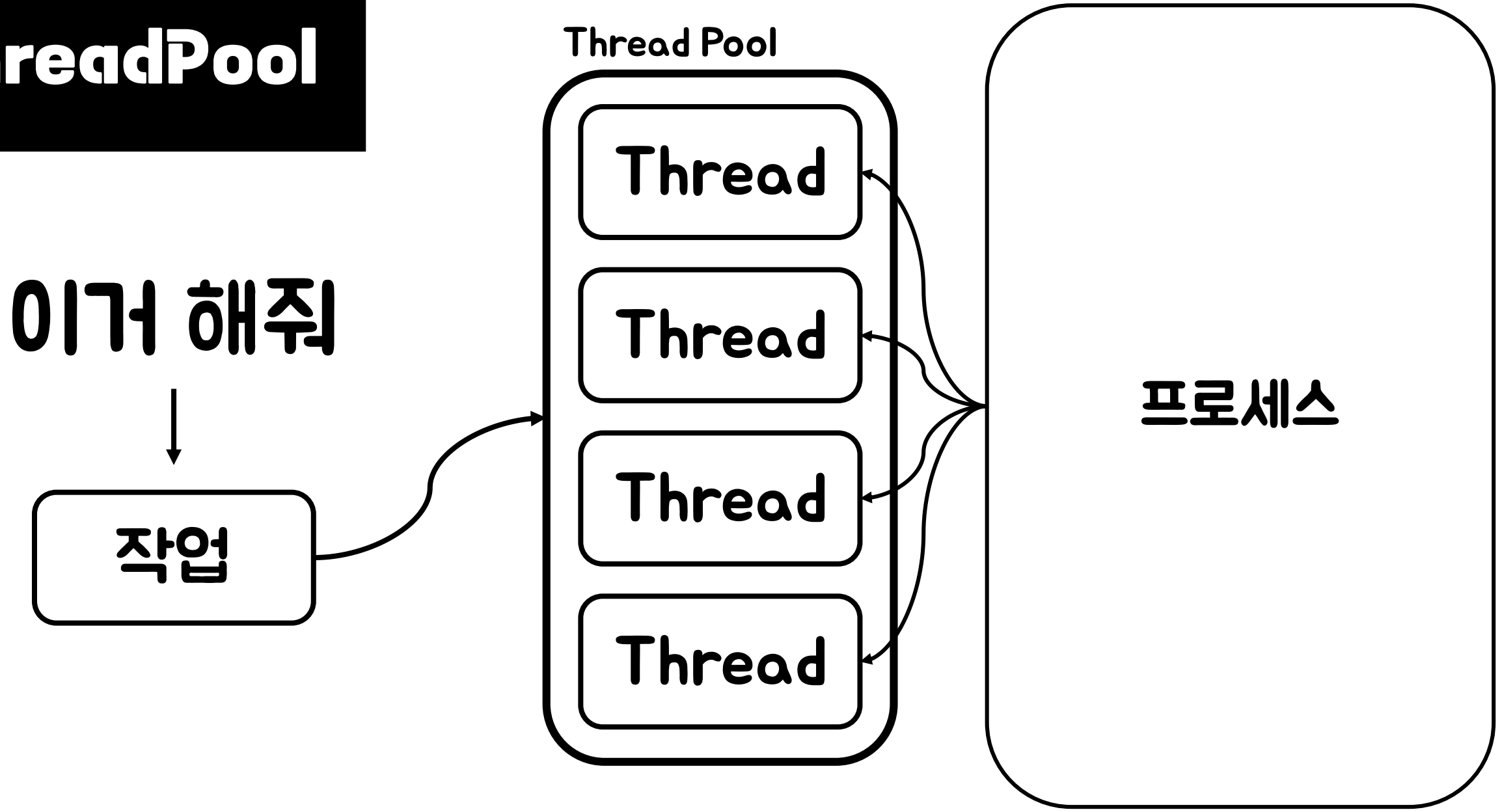
스레드 사용 방법

Thread, ThreadPool, Task, Async

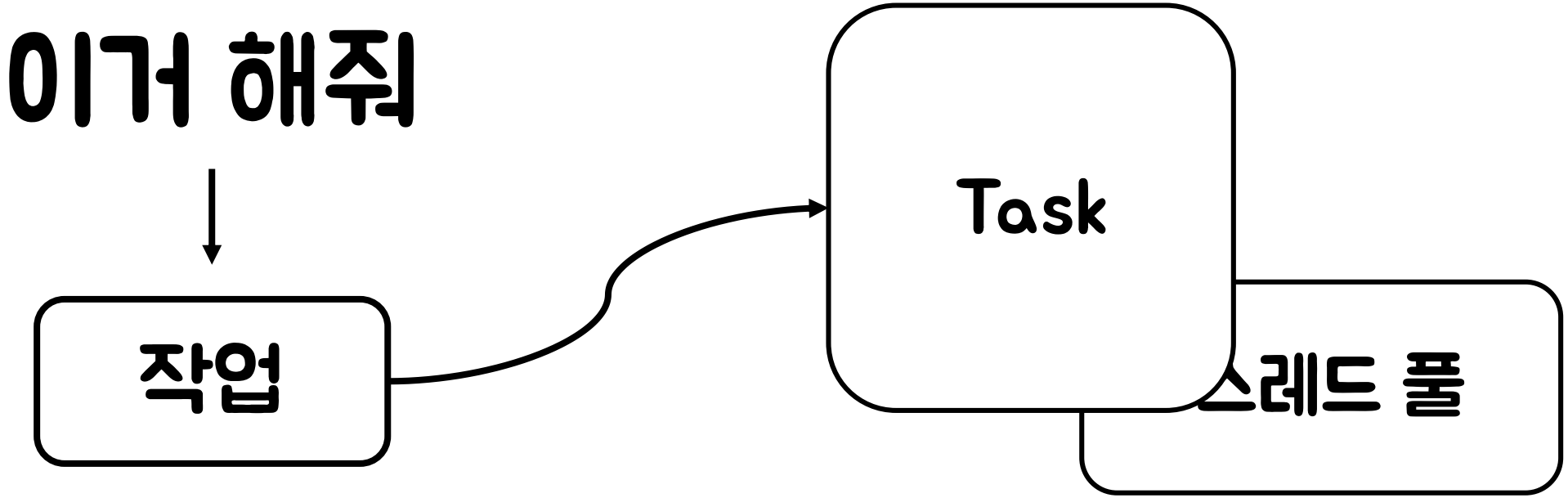
Thread



ThreadPool



Task



Async

갖고와 내가 할게

Task

스레드 풀

실행!!

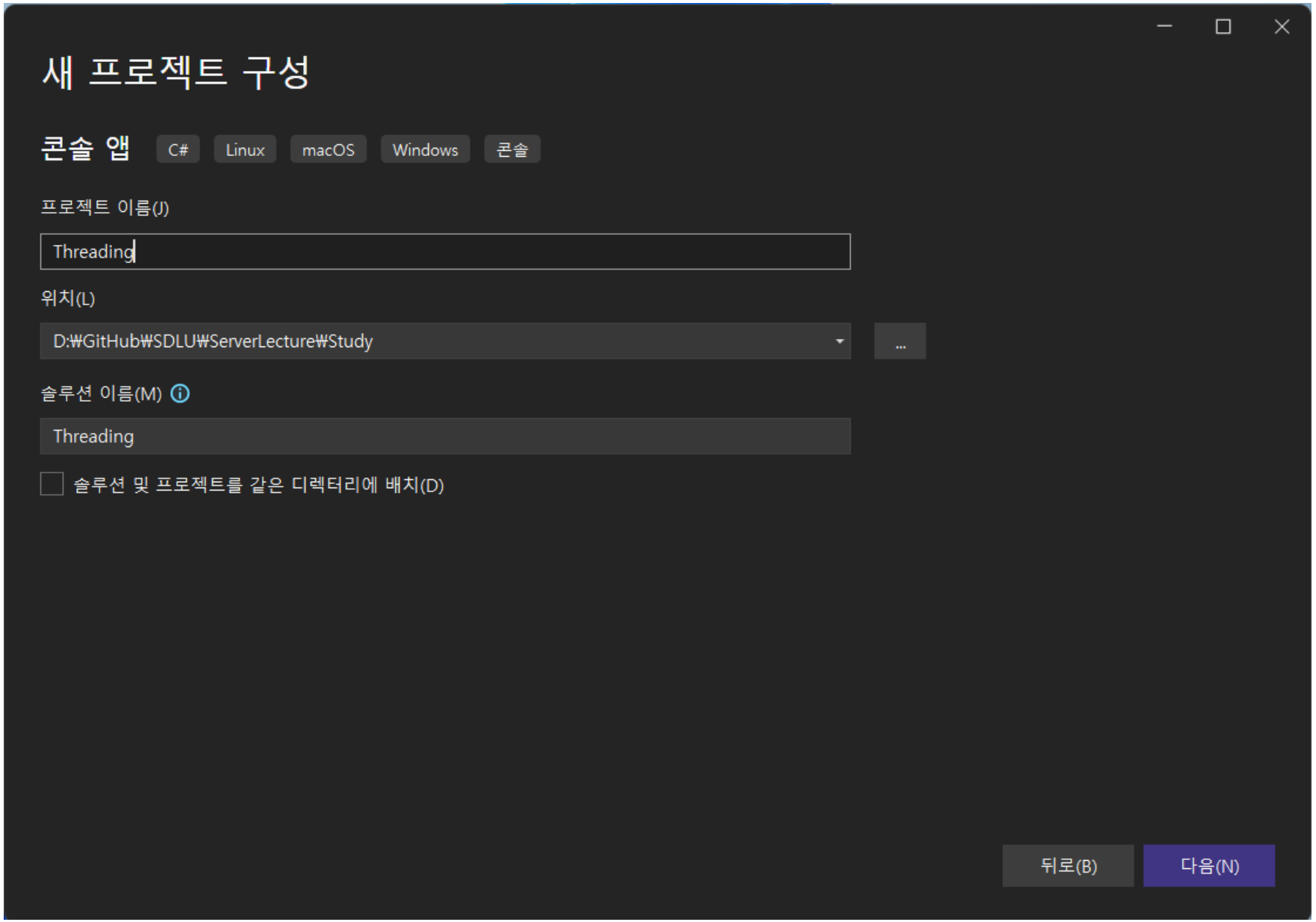
Async (비동기) 작업

환경 설정

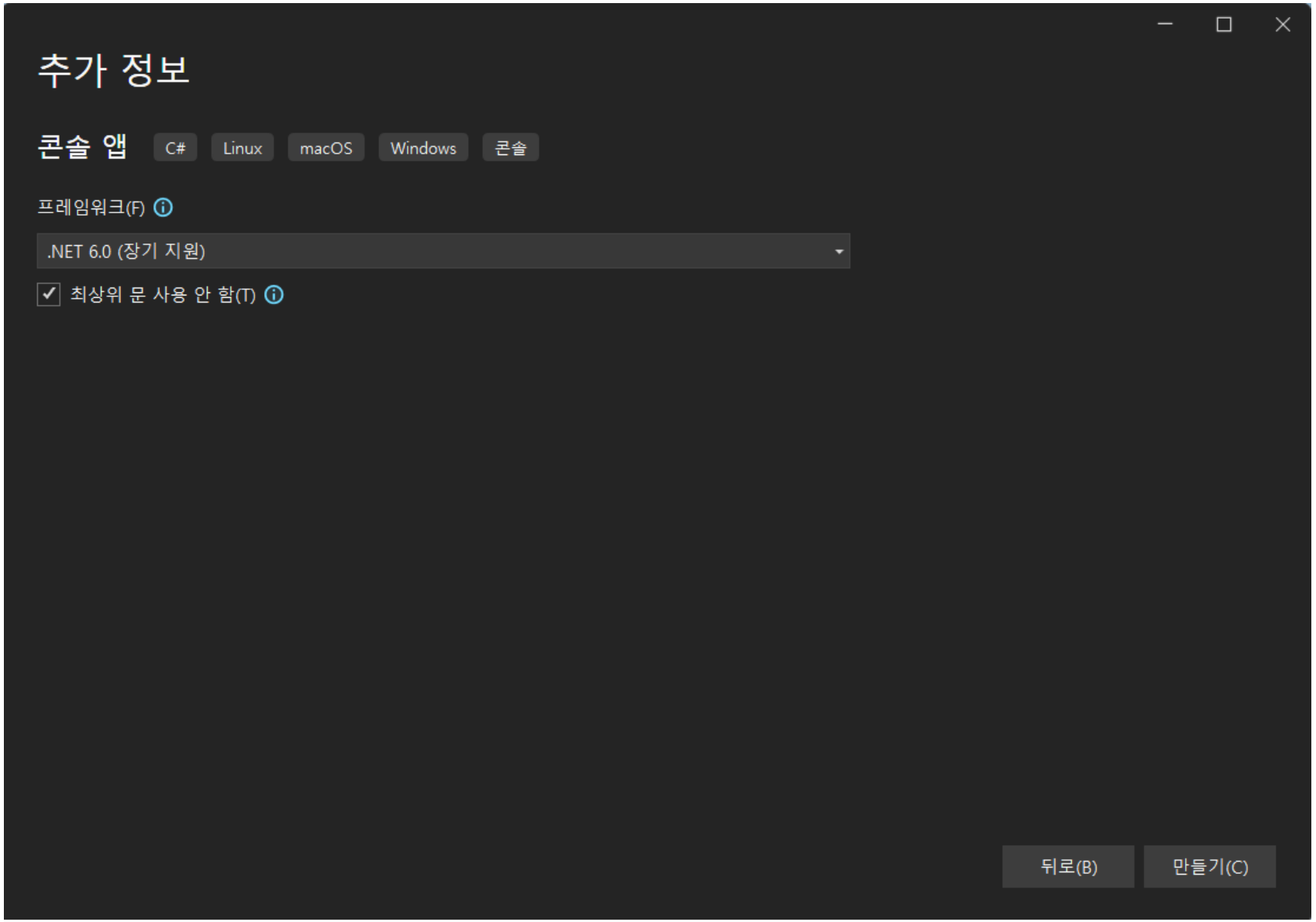
프로젝트 생성


새 프로젝트 만들기
템플릿 검색(Alt+S)(S)
모두 지우기(C)
최근 프로젝트 템플릿(R)
콘솔 앱 C#
콘솔 앱 C++
빈 프로젝트 C++
Windows 데스크톱 애플리케이션 C++
Windows Forms 앱 C#
정적 라이브러리 C++
C#
모든 플랫폼(P)
모든 프로젝트 형식(T)
콘솔 앱
Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 명령줄 응용 프로그램을 만들기 위한 프로젝트
C# Linux macOS Windows 콘솔
클래스 라이브러리
.NET 또는 .NET Standard를 대상으로 하는 클래스 라이브러리를 만들기 위한 프로젝트
C# Android Linux macOS Windows 라이브러리
MSTest 테스트 프로젝트
Windows, Linux 및 MacOS의 .NET에서 실행할 수 있는 MSTest 단위 테스트를 포함하는 프로젝트입니다.
C# Linux macOS Windows 테스트
Windows Forms 앱
.NET WinForms(Windows Forms) 앱을 만들기 위한 프로젝트 템플릿입니다.
C# Windows 데스크톱
Windows Forms 앱(.NET Framework)
뒤로(B)
다음(N)

프로젝트 생성


새 프로젝트 구성
콘솔 앱
C#
Linux
macOS
Windows
콘솔
프로젝트 이름(J)
Threading
위치(L)
D:₩GitHub₩SDLU₩ServerLecture₩Study
...
솔루션 이름(M)
Threading
솔루션 및 프로젝트를 같은 디렉터리에 배치(D)
뒤로(B)
다음(N)

프로젝트 생성


추가 정보
콘솔 앱
C#
Linux
macOS
Windows
콘솔
프레임워크(F)
.NET 6.0 (장기 지원)
최상위 문 사용 안 함(T)
뒤로(B)
만들기(C)

Thread

ThreadTest1

```
using System;
using System.Threading;

namespace Threading
{
    public class ThreadTest1
    {
        public void Start()
        {
            Thread thread1 = new Thread(Thread1);
            Thread thread2 = new Thread(Thread2);

            thread1.Name = "1번 스레드";
            thread2.Name = "2번 스레드";

            thread1.Start();
            thread2.Start();
        }

        private void Thread1()
        {
            Console.WriteLine($"Thread1, {Thread.CurrentThread.Name}");
        }

        private void Thread2()
        {
            Console.WriteLine($"Thread2, {Thread.CurrentThread.Name}");
        }
    }
}
```

ThreadTest1

Microsoft Visual Studio 디버그 콘솔

```
Thread1, 1번 스레드
Thread2, 2번 스레드

D:₩GitHub₩SDLU₩ServerLecture₩Thread
프로세스 34260개)이(가) 종료되었습니
디버깅이 중지될 때 콘솔을 자동으로 
깅이 중지되면 자동으로 콘솔 닫기]를
이 창을 닫으려면 아무 키나 누르세요
```

Microsoft Visual Studio 디버그 콘솔

```
Thread2, 2번 스레드
Thread1, 1번 스레드

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug₩net6.0₩MyThre
ading.exe(프로세스 32016개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버
깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.

이 창을 닫으려면 아무 키나 누르세요...
```

ThreadTest2

```
using System;
using System.Threading;

namespace MyThreading
{
    public class ThreadTest2
    {
        public void Start()
        {
            Thread thread1 = new Thread(Thread1);
            Thread thread2 = new Thread(Thread2);

            thread1.Start();
            thread2.Start();

            Console.WriteLine("This is main");
        }

        private void Thread1()
        {
            Console.WriteLine($"Hello, Thread1!");
        }

        private void Thread2()
        {
            Console.WriteLine($"Hello, Thread2!");
        }
    }
}
```

ThreadTest2

Microsoft Visual Studio 디버그 콘솔

```
This is main
Hello, Thread1!
Hello, Thread2!

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug
₩net6.0₩MyThreading.exe(프로세스 33920개)이(가) 종료되
었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [
옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫
기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

ThreadTest2

```
namespace MyThreading
{
    public class ThreadTest2
    {
        public void Start()
        {
            Thread thread1 = new Thread(Thread1);
            Thread thread2 = new Thread(Thread2);

            thread1.Start();
            thread2.Start();

            thread1.Join();
            thread2.Join();

            Console.WriteLine("This is main");
        }

        private void Thread1()
        {
            Console.WriteLine($"Hello, Thread1!");
        }

        private void Thread2()
        {
            Console.WriteLine($"Hello, Thread2!");
        }
    }

}
```

ThreadTest2

Microsoft Visual Studio 디버그 콘솔

```
Hello, Thread1!
Hello, Thread2!
This is main

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug₩net6.0₩MyThreading.exe(
프로세스 14828개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디
버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

ThreadTest3

```
1  using System;
2  using System.Threading;
3
4  namespace MyThreading
5  {
6      public class ThreadTest3
7      {
8          public void Start()
9          {
10             Thread thread1 = new Thread(Thread1);
11             Thread thread2 = new Thread(Thread2);
12
13             thread1.Start();
14             thread2.Start();
15
16             Console.WriteLine("This is main");
17         }
18
19         private void Thread1()
20         {
21             for(int i = 0; i < 100; i++)
22                 Console.WriteLine($"Hello, Thread1! : {i}");
23         }
24
25         private void Thread2()
26         {
27             for(int i = 0; i < 100; i++)
28                 Console.WriteLine($"Hello, Thread2! : {i}");
29         }
30     }
31 }
```

ThreadTest3

Microsoft Visual Studio 디버그 콘솔

```
This is main
Hello, Thread2! : 0
Hello, Thread1! : 0
Hello, Thread2! : 1
Hello, Thread2! : 2
Hello, Thread2! : 3
Hello, Thread2! : 4
Hello, Thread2! : 5
Hello, Thread2! : 6
Hello, Thread2! : 7
Hello, Thread2! : 8
Hello, Thread2! : 9
Hello, Thread2! : 10
Hello, Thread2! : 11
Hello, Thread2! : 12
Hello, Thread2! : 13
Hello, Thread2! : 14
Hello, Thread2! : 15
Hello, Thread2! : 16
Hello, Thread2! : 17
Hello, Thread2! : 18
Hello, Thread2! : 19
Hello, Thread2! : 20
Hello, Thread2! : 21
```

Microsoft Visual Studio 디버그 콘솔

```
Hello, Thread1! : 94
Hello, Thread1! : 95
Hello, Thread1! : 96
Hello, Thread1! : 97
Hello, Thread2! : 94
Hello, Thread1! : 98
Hello, Thread2! : 95
Hello, Thread1! : 99
Hello, Thread2! : 96
Hello, Thread2! : 97
Hello, Thread2! : 98
Hello, Thread2! : 99

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug₩net6.0₩MyThreading.exe(프
로세스 17020개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버
깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

ThreadTest3

```
using System;
using System.Threading;

namespace MyThreading
{
    public class ThreadTest3
    {
        public void Start()
        {
            Thread thread1 = new Thread(Thread1);
            Thread thread2 = new Thread(Thread2);

            thread1.IsBackground = true;
            thread2.IsBackground = true;

            thread1.Start();
            thread2.Start();

            Console.WriteLine("This is main");
        }

        private void Thread1()
        {
            for(int i = 0; i < 100; i++)
                Console.WriteLine($"Hello, Thread1! : {i}");
        }

        private void Thread2()
        {
            for(int i = 0; i < 100; i++)
                Console.WriteLine($"Hello, Thread2! : {i}");
        }
    }
}
```

ThreadTest3

Microsoft Visual Studio 디버그 콘솔

```
This is main
Hello, Thread1! : 0
Hello, Thread2! : 0
Hello, Thread2! : 1
Hello, Thread2! : 2
Hello, Thread2! : 3
Hello, Thread1! : 1
Hello, Thread1! : 2
Hello, Thread1! : 3
Hello, Thread1! : 4
Hello, Thread1! : 5
Hello, Thread1! : 6
Hello, Thread1! : 7
Hello, Thread1! : 8
Hello, Thread1! : 9
Hello, Thread1! : 10
Hello, Thread1! : 11
Hello, Thread1! : 12

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug₩net6.0₩MyThreading.exe(
프로세스 15116개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버
깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

ThreadPool

ThreadPoolTest

```
1  using System;
2  using System.Threading;
3
4  namespace MyThreading
5  {
6      public class ThreadPoolTest
7      {
8          public void Start()
9          {
10             ThreadPool.QueueUserWorkItem(Work1, 10);
11             ThreadPool.QueueUserWorkItem(Work2, "This is String");
12
13             Console.WriteLine("This is main");
14
15             Console.ReadKey();
16         }
17
18         private void Work1(object obj)
19         {
20             Console.WriteLine($"Work1 : {obj}");
21         }
22
23         private void Work2(object obj)
24         {
25             Console.WriteLine($"Work2 : {obj}");
26         }
27     }
28 }
```

ThreadPoolTest

Microsoft Visual Studio 디버그 콘솔

```
This is main
Work1 : 10
Work2 : This is String

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thr
hreading.exe(프로세스 25184개)이(가) 종료되
디버깅이 중지될 때 콘솔을 자동으로 닫으려면
디버깅] > [디버깅이 중지되면 자동으로 콘솔
합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

선택 Microsoft Visual Studio 디버그 콘솔

```
This is main
Work2 : This is String
Work1 : 10

D:₩GitHub₩SDLU₩ServerLecture₩Threading₩Thread₩bin₩Debug₩net6.0₩My
Threading.exe(프로세스 28732개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [
디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설
정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

Task

TaskTest1

```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class TaskTest1
    {
        public void Start()
        {
            Task task1 = new Task(Task1);
            Task task2 = new Task(Task2);

            task1.Start();
            task2.Start();

            Task.WaitAll(task1, task2);

            Console.WriteLine("This is main");
        }

        private void Task1()
        {
            for (int i = 0; i < 100; i++)
                Console.WriteLine($"Task1 : {i}");
        }

        private void Task2()
        {
            for (int i = 0; i < 100; i++)
                Console.WriteLine($"Task2 : {i}");
        }
    }
}
```

TaskTest2

```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class TaskTest2
    {
        public void Start()
        {
            Task task1 = Task.Run(Task1);
            Task task2 = Task.Run(Task2);

            Task.WaitAll(task1, task2);

            Console.WriteLine("This is main");
        }

        private void Task1()
        {
            for (int i = 0; i < 100; i++)
                Console.WriteLine($"Task1 : {i}");
        }

        private void Task2()
        {
            for (int i = 0; i < 100; i++)
                Console.WriteLine($"Task2 : {i}");
        }
    }
}
```

Async

AsyncTest1

```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class AsyncTest1
    {
        public void Start()
        {
            HelloAsync(() => Console.WriteLine("Task done"));
            Console.WriteLine("Hello, Main!");
        }

        private async void HelloAsync(Action callback)
        {
            for(int i = 0; i < 10; i ++)
            {
                Console.WriteLine($"Hello, Async! : {i}");

                await Task.Delay(500);
            }

            callback?.Invoke();
        }
    }
}
```

AsyncTest2

```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class AsyncTest2
    {
        public async void Start()
        {
            await HelloAsync(() => Console.WriteLine("Task done"));

            Console.WriteLine("Hello, Main!");
        }

        private async Task HelloAsync(Action callback)
        {
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine($"Hello, Async! : {i}");

                await Task.Delay(500);
            }

            callback?.Invoke();
        }
    }
}
```

AsyncTest3

```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class AsyncTest3
    {
        public async void Start()
        {
            int printedCount = await HelloAsync(() => Console.WriteLine("Task done"));

            Console.WriteLine($"printed count : {printedCount}");
        }

        private async Task<int> HelloAsync(Action callback)
        {
            int i = 0;
            for (; i < 10; i++)
            {
                Console.WriteLine($"Hello, Async! : {i}");

                await Task.Delay(500);
            }

            callback?.Invoke();

            return i;
        }
    }
}
```

AsyncTest4

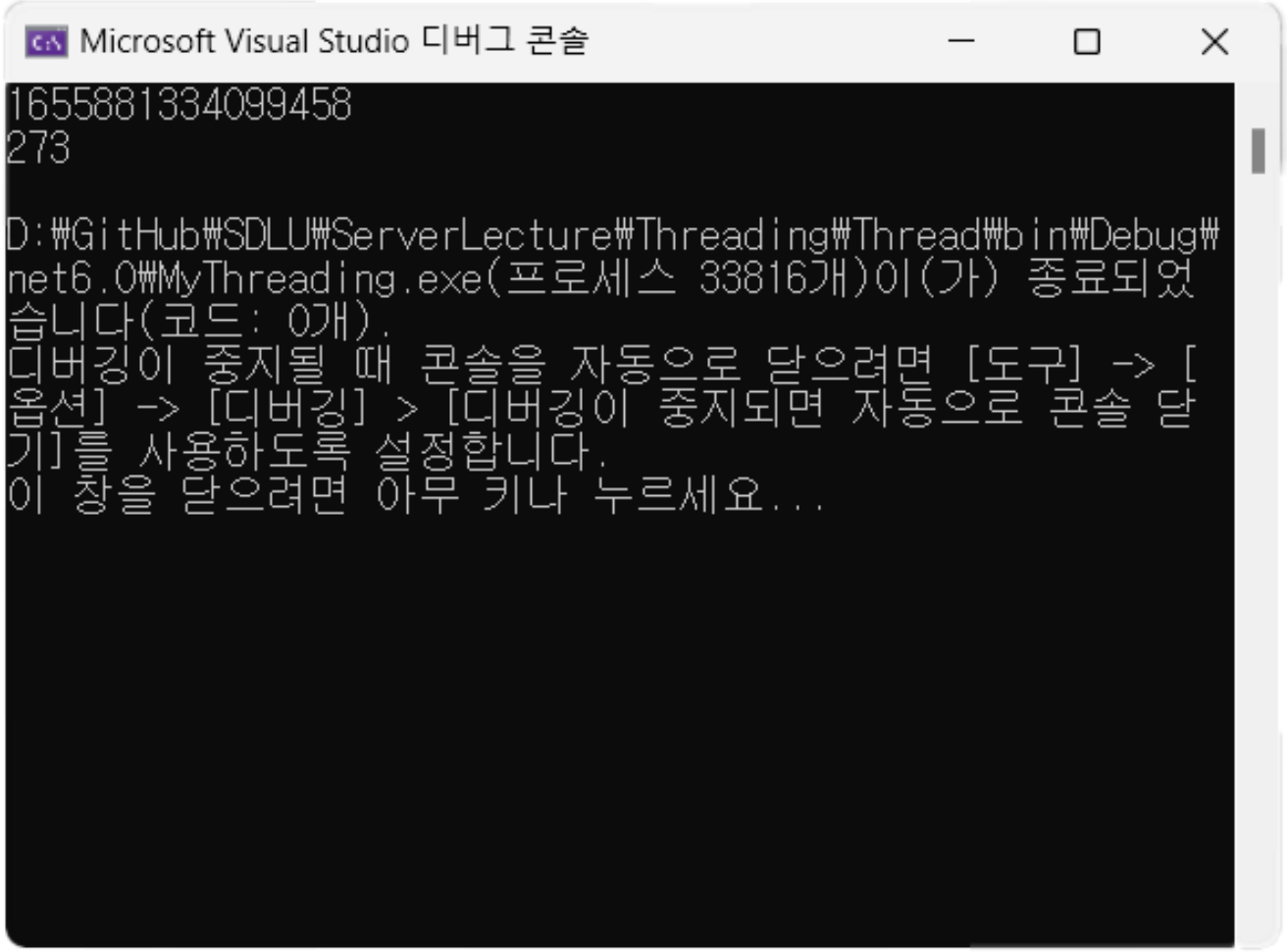
```
using System;
using System.Threading.Tasks;

namespace MyThreading
{
    public class AsyncTest3
    {
        public async void Start()
        {
            int printedCount = await HelloAsync(() => Console.WriteLine("Task done"));

            Console.WriteLine($"printed count : {printedCount}");
        }

        private async Task<int> HelloAsync(Action callback)
        {
            int i = 0;
            for (; i < 10; i++)
            {
                Console.WriteLine($"Hello, Async! : {i}");

                await Task.Delay(500);
            }

            callback?.Invoke();

            return i;
        }
    }
}
```

문제 발생시키기

Lock

```
1  using System;
2  using System.Diagnostics;
3  using System.Threading.Tasks;
4
5  namespace MyThreading
6  {
7      public class LockTest
8      {
9          private long origin = 0;
10
11         public void Start()
12         {
13             Stopwatch timer = new Stopwatch();
14             timer.Start();
15
16             Task t1 = Task.Run(() => Add(0 * 10000 + 1, 2500 * 10000));
17             Task t2 = Task.Run(() => Add(2500 * 10000 + 1, 5000 * 10000));
18             Task t3 = Task.Run(() => Add(5000 * 10000 + 1, 7500 * 10000));
19             Task t4 = Task.Run(() => Add(7500 * 10000 + 1, 10000 * 10000));
20
21             Task.WaitAll(t1, t2, t3, t4);
22             Console.WriteLine(origin);
23
24             timer.Stop();
25             Console.WriteLine(timer.ElapsedMilliseconds);
26         }
27
28         private void Add(long from, long to)
29         {
30             for (long i = from; i < to + 1; i++)
31                 origin += i;
32         }
33     }
34 }
```

Lock



기대값 : 5000000050000000

결과값 : ???

덧셈

```
public void Add(int* origin, int num)
{
    int temp = *origin;
    temp += num;
    origin* = temp;
}
```

덧셈

```
public void Add(int* origin, int num)
{
    int temp = *origin;
    temp += num;
    origin* = temp;
}
```

<- 덮어쓰기 전에 누군가 또 덧셈을 했다면?

덧셈

Thread1	Thread2
1 temp = 10 + 15	2 temp = 10 + 10
3 origin* = temp	4 origin* = temp

기대값 : 10 + 15 + 10 = 35

결과값 : 10 + 15(무시됨) + 10 = 20

Lock

한 번에 하나의 스레드만 접근하게 하자!

Lock

Lock

한 번에 하나의 스레드만 접근하게 하자!

locking

Lock

MemoryBarrier
Interlocked
Mutex
Semaphore
Monitor
lock

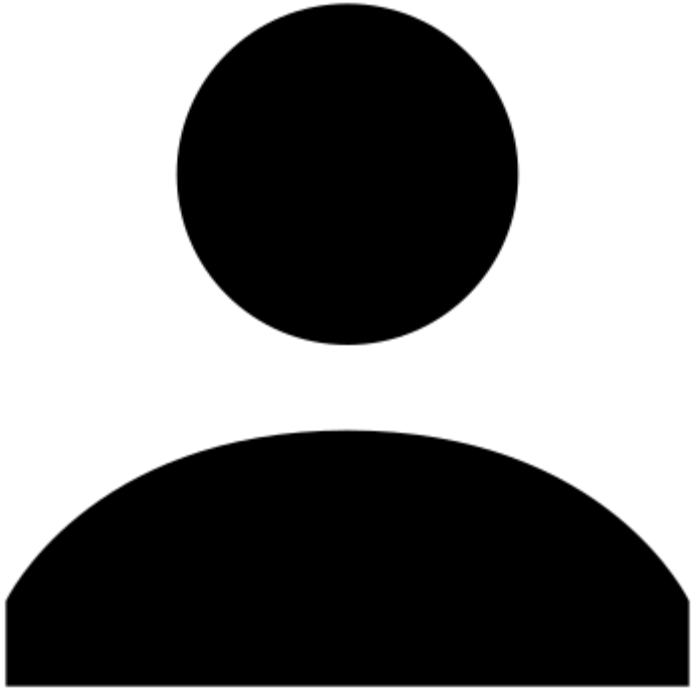
너무 많아! 스레드 관리하기 어렵자!

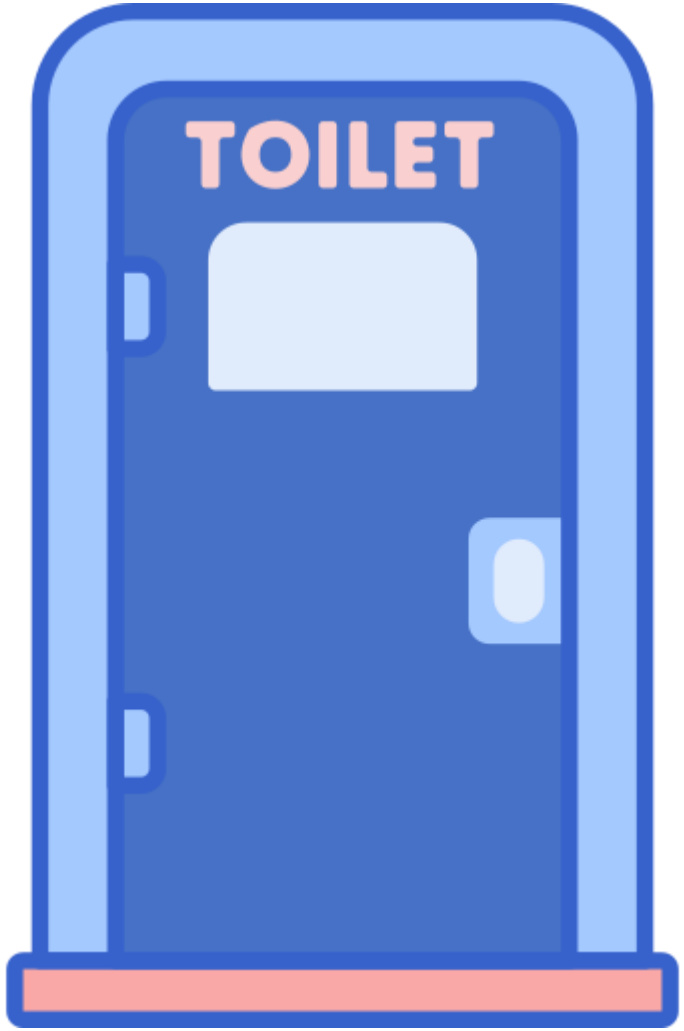
Lock

이것만 기억하자 lock

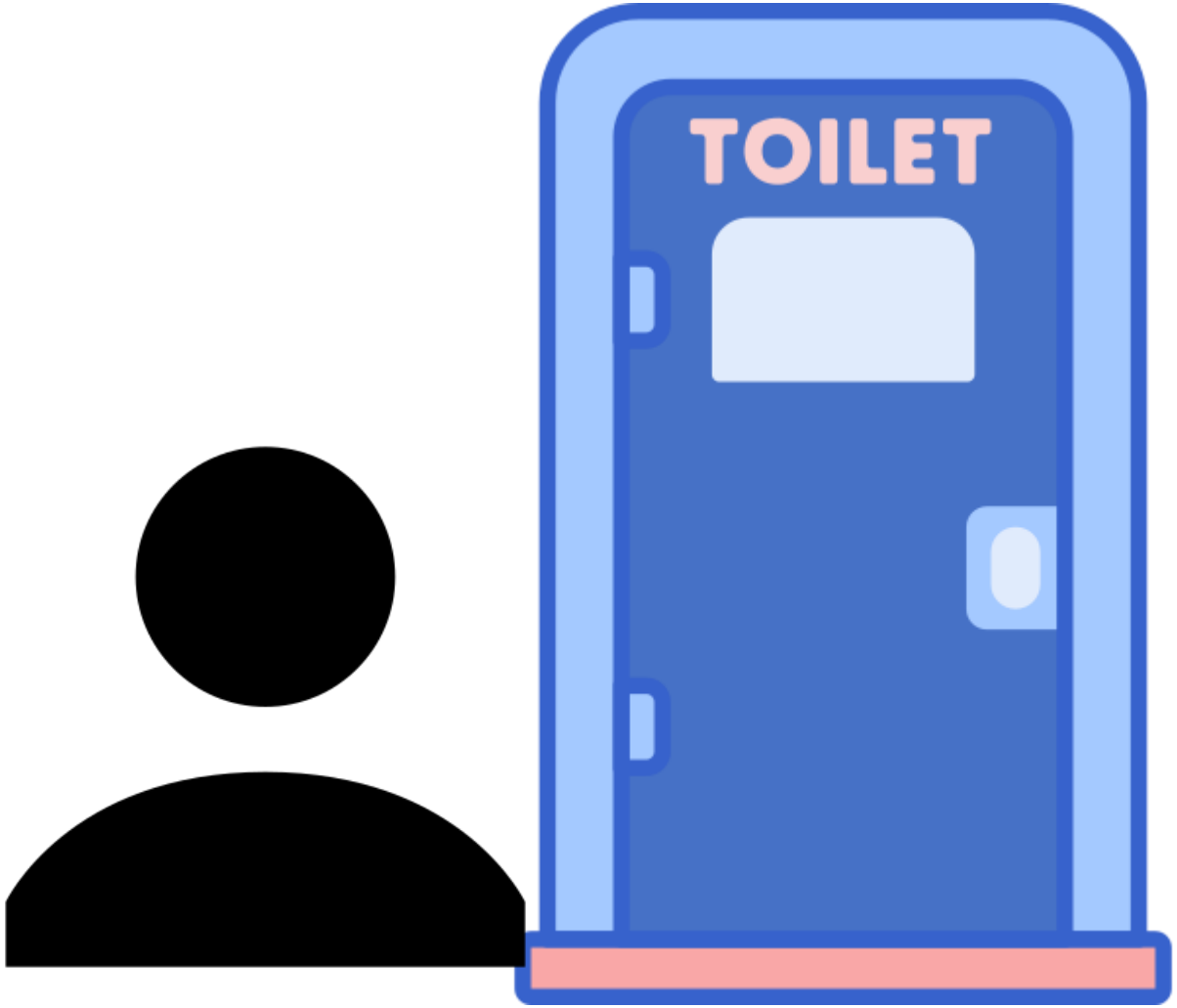
Lock 원리

아 똥매려

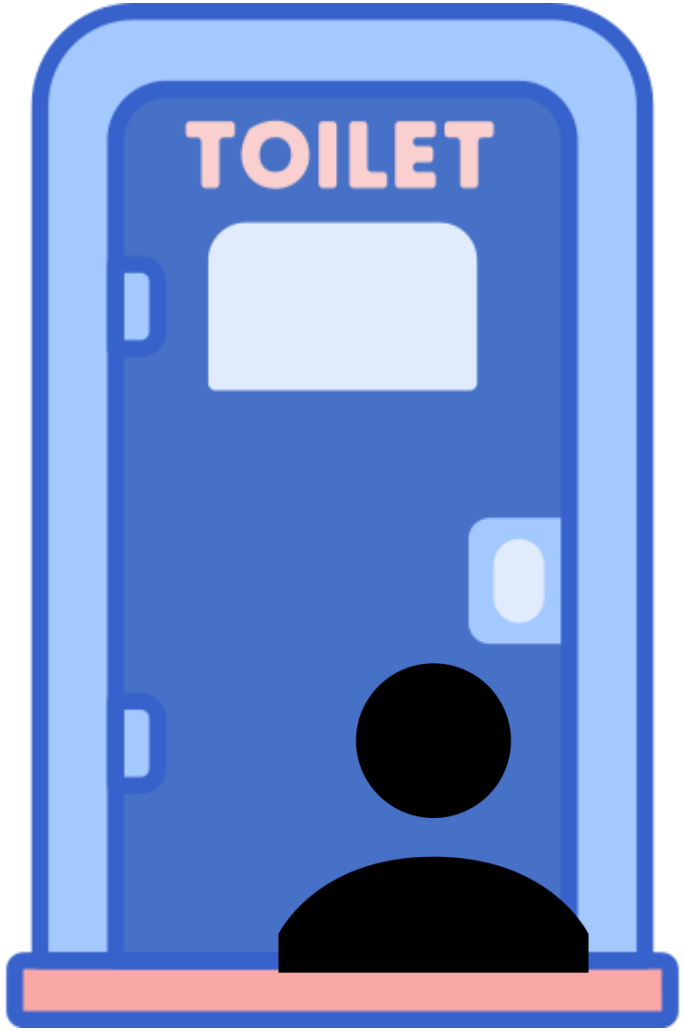




Lock 원리



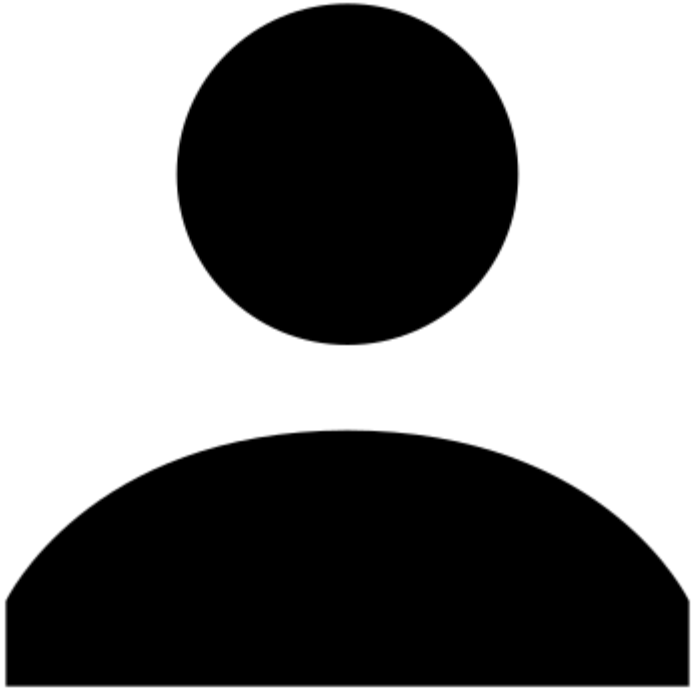
Lock 원리



Lock 원리

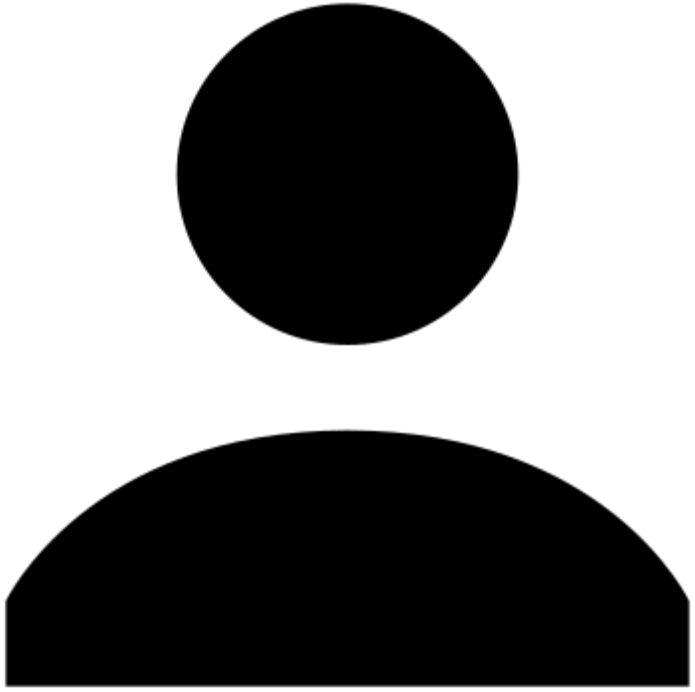
Lock 원리

아씨 나도 똥매린데

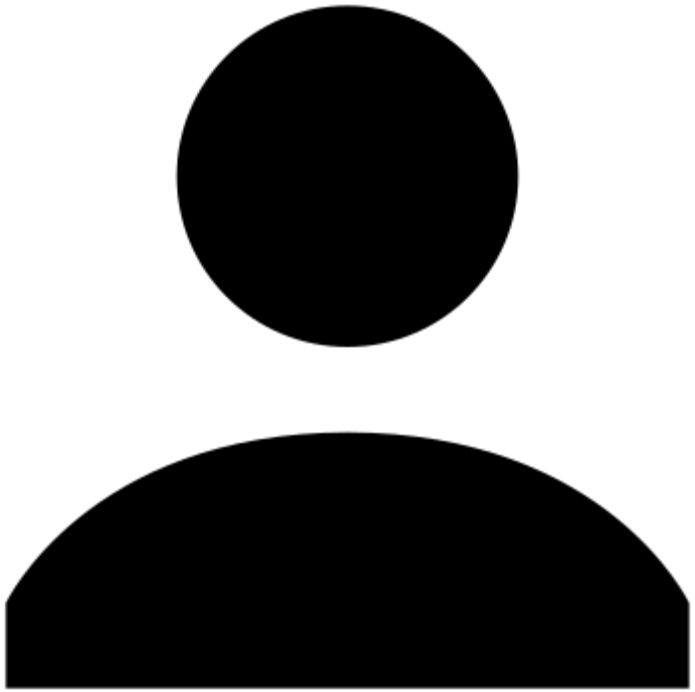


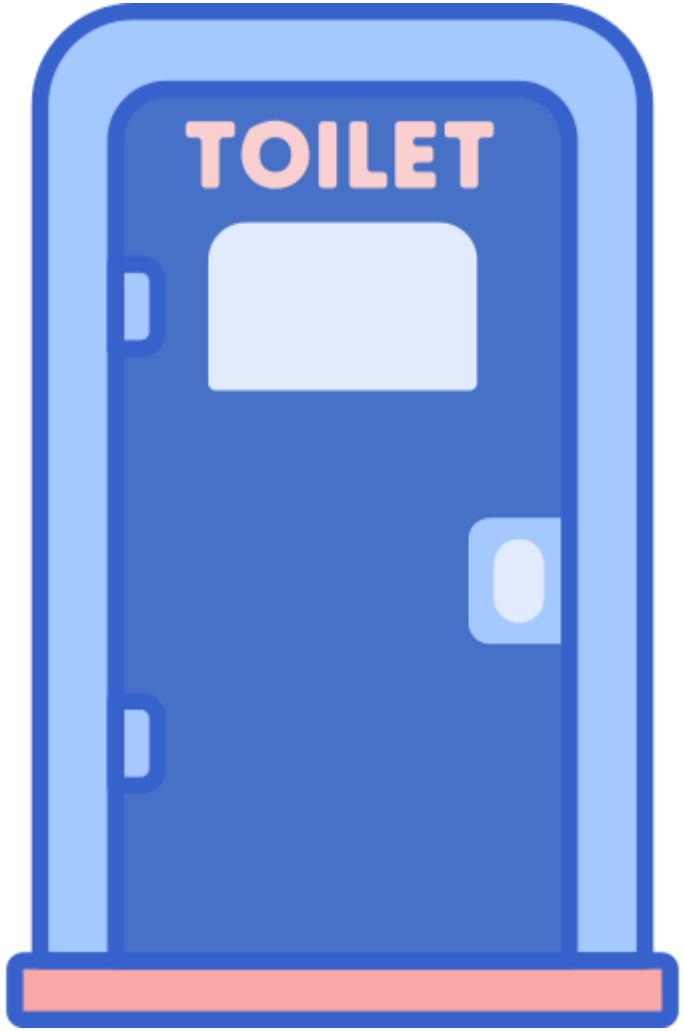
Lock 원리

아씨 나도 똥매린데

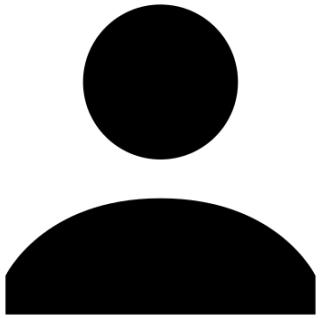


Lock 원리



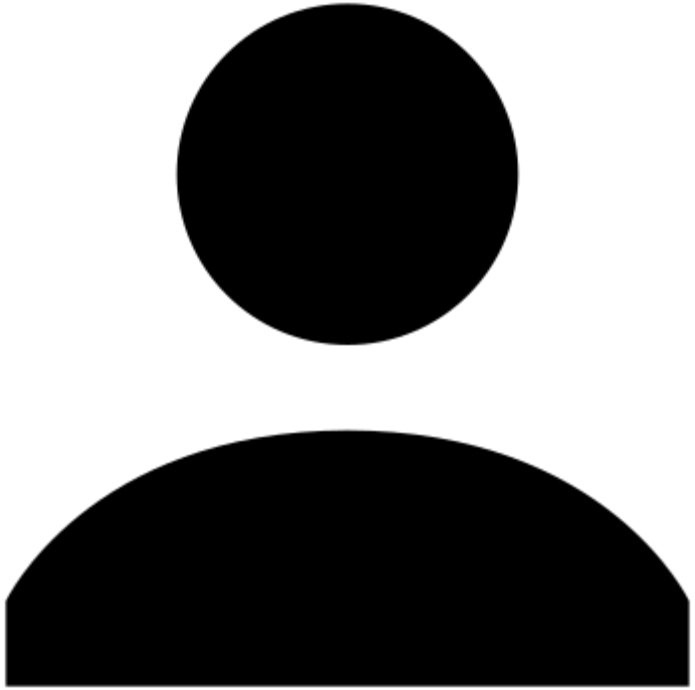


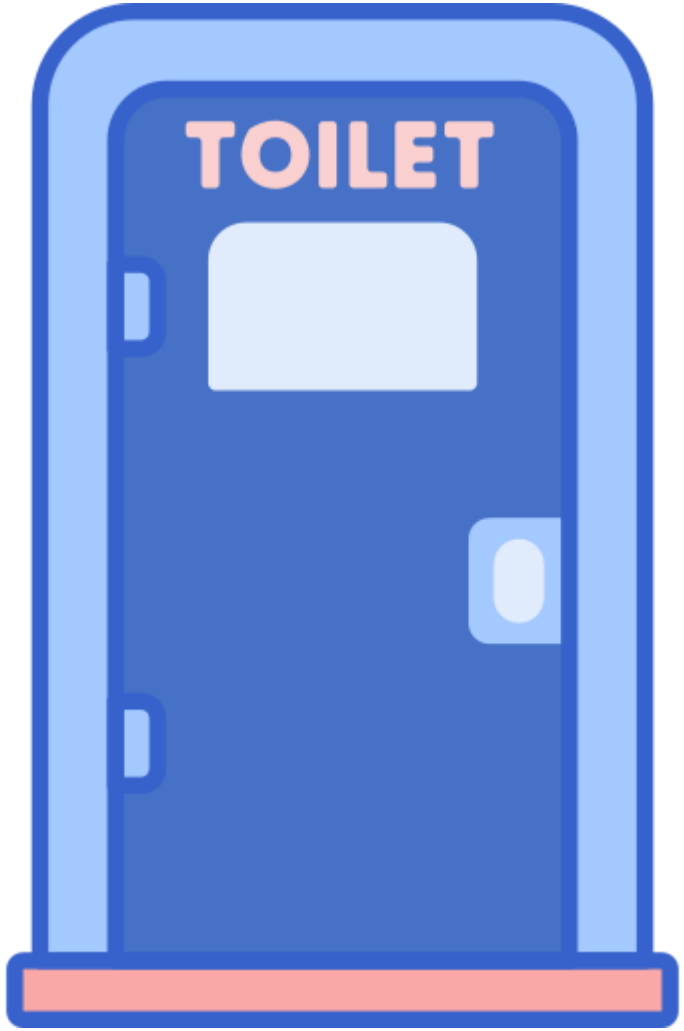
나 다 씀



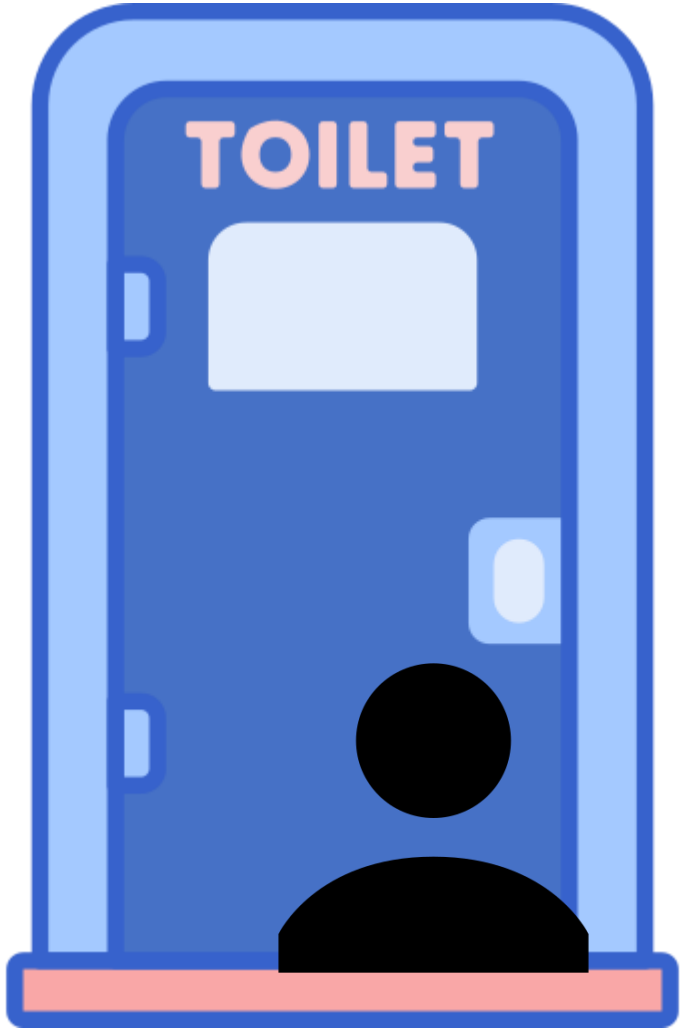
Lock 원리

ㅇㅋ 내 차례





Lock 원리



Lock 원리

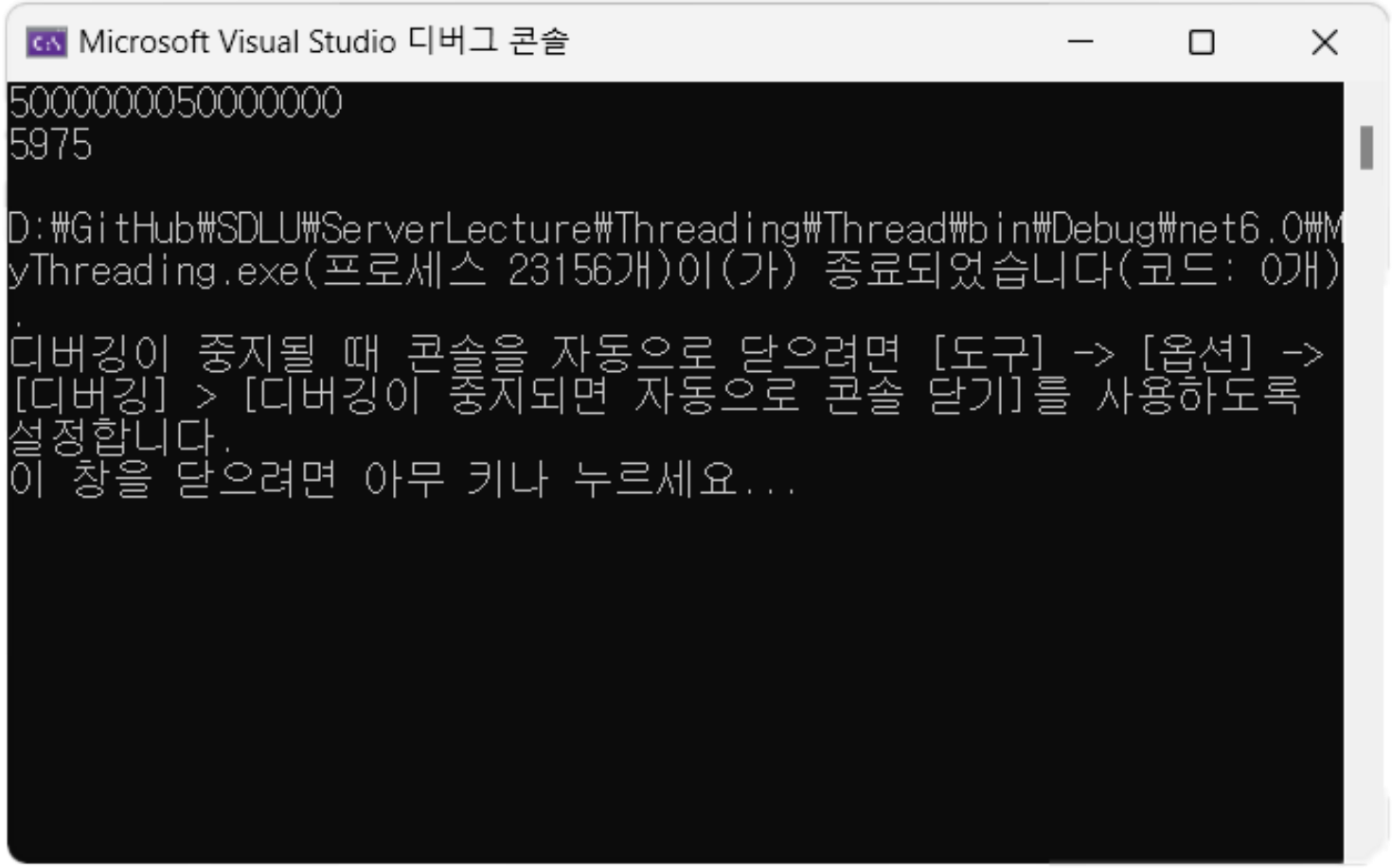
Lock 원리

<- lock

Lock 사용

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;

namespace MyThreading
{
    public class LockTest
    {
        private long origin = 0;
        private object locker = new object();

        public void Start()
        {
            Stopwatch timer = new Stopwatch();
            timer.Start();

            Task t1 = Task.Run(() => Add(0 * 10000 + 1, 2500 * 10000));
            Task t2 = Task.Run(() => Add(2500 * 10000 + 1, 5000 * 10000));
            Task t3 = Task.Run(() => Add(5000 * 10000 + 1, 7500 * 10000));
            Task t4 = Task.Run(() => Add(7500 * 10000 + 1, 10000 * 10000));

            Task.WaitAll(t1, t2, t3, t4);

            Console.WriteLine(origin);

            timer.Stop();
            Console.WriteLine(timer.ElapsedMilliseconds);
        }

        private void Add(long from, long to)
        {
            for (long i = from; i < to + 1; i++)
                lock(locker)
                    origin += i;
        }
    }
}
```

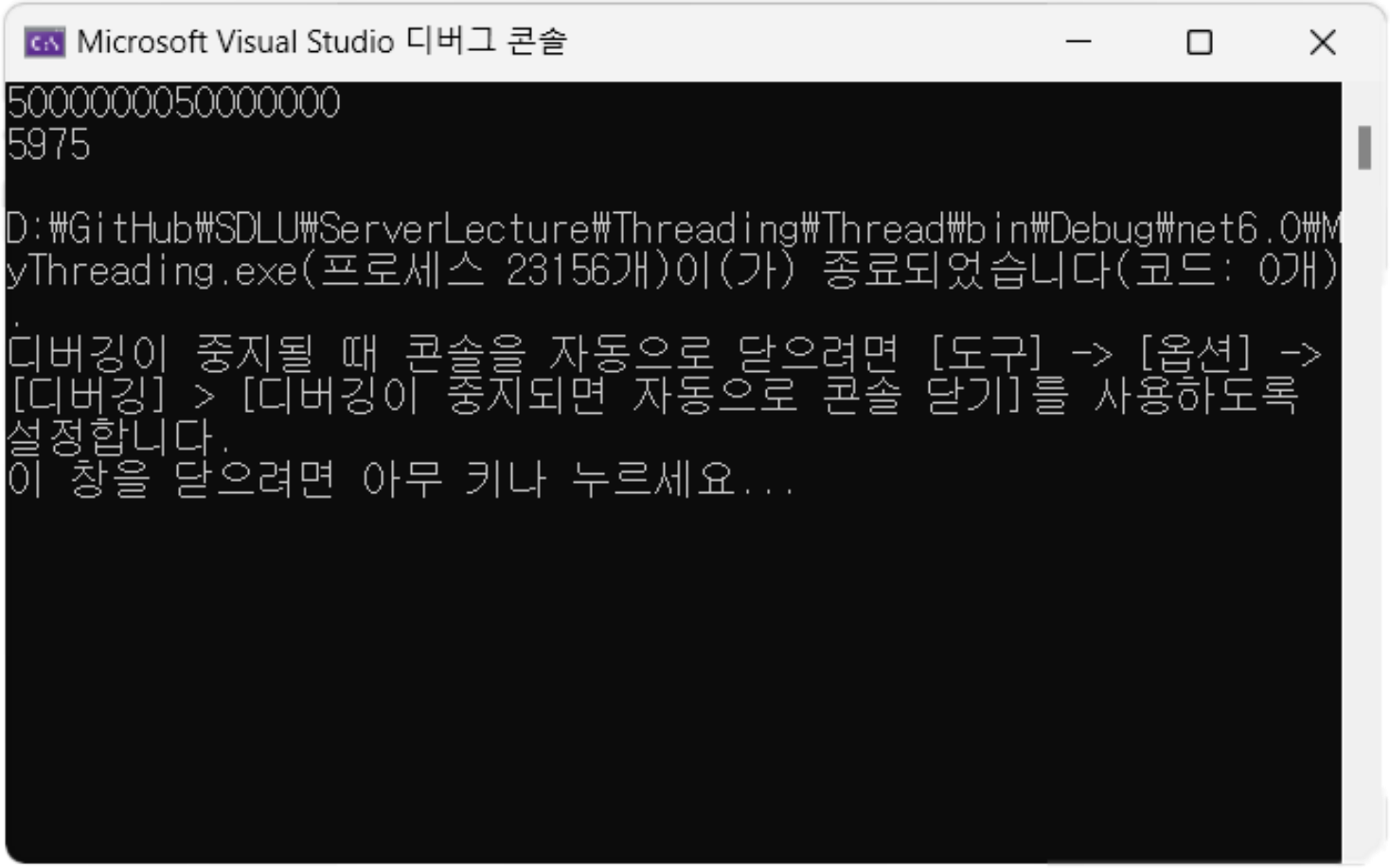
Lock 사용



기대값 : 5000000050000000

결과값 : 5000000050000000

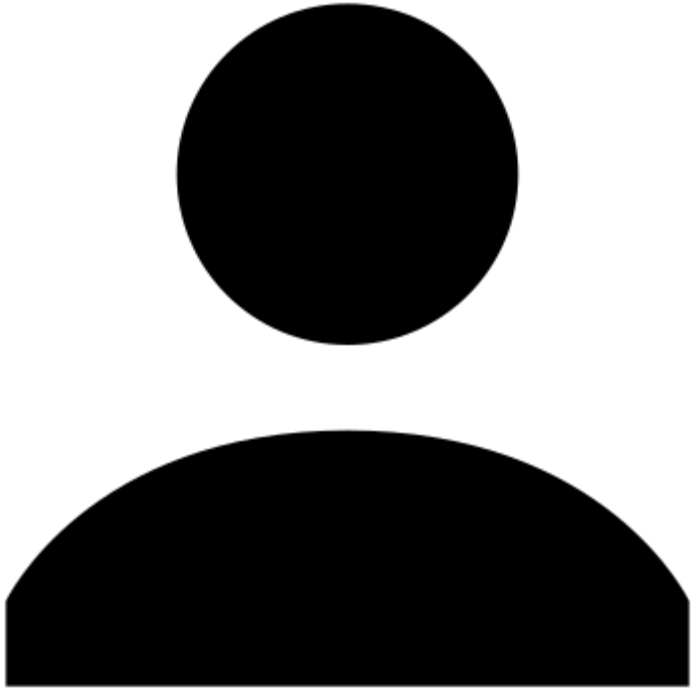
Lock 사용



시간이 왜 저래?

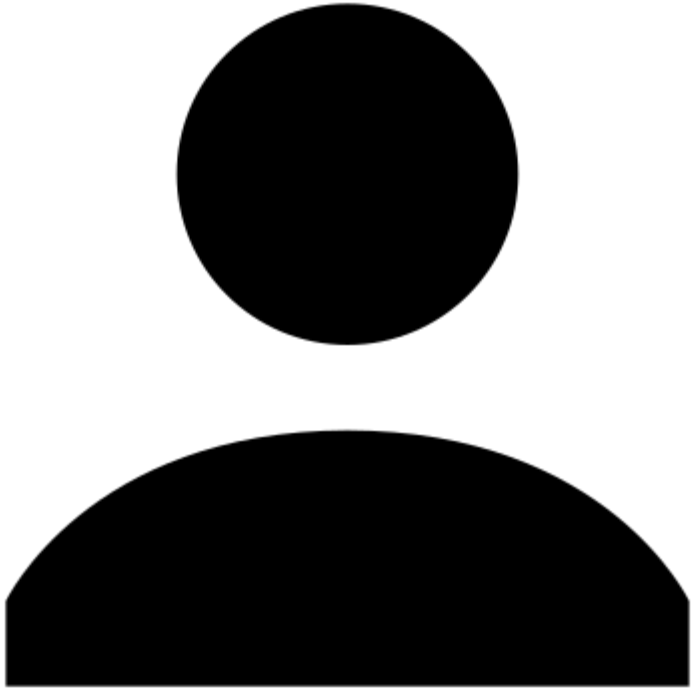
Lock 사용

아씨 나도 똥매린데



Lock 사용

나 스레드 2



Lock 사용

나 스레드 1

나 스레드 2

스레드 1이 자리를 비울 때까지 무한 대기

Lock 사용

나 스레드 1

나 스레드 2

lock은 신중하게 쓰도록!