GC .NET

Александр Павлов

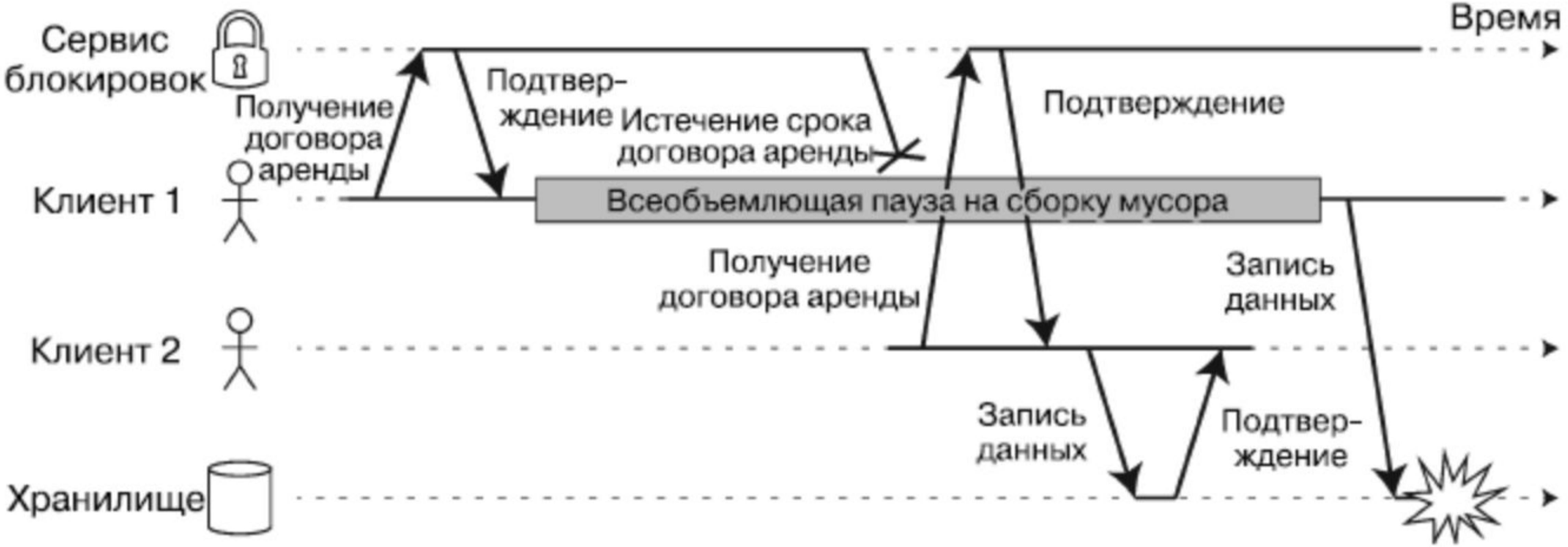
Плюсы

Автоматическое управление памятью

Минусы

Таки должен знать, как это работает

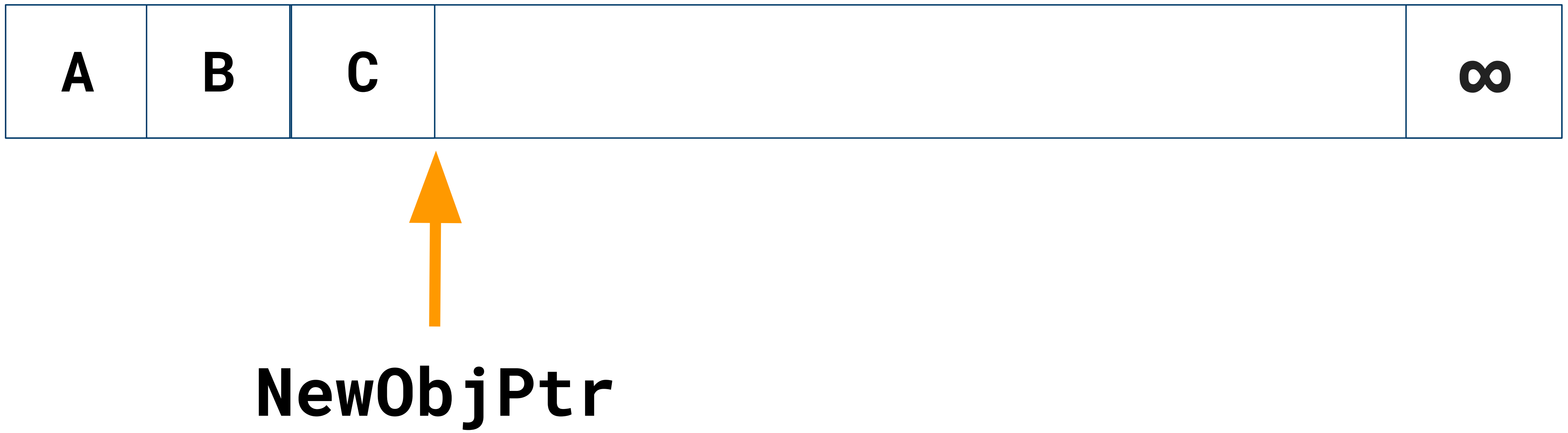
Почему?



Сервис блокировок
Время
Получение договора аренды
Подтвер-ждение
Истечение срока договора аренды
Подтверждение
Клиент 1
Всеобъемлющая пауза на сборку мусора
Получение договора аренды
Запись данных
Клиент 2
Запись данных
Подтвер-ждение
Хранилище

© Мартин Клеппман “Высоконагруженные приложения. Программирование, масштабирование, поддержка”

Как оно работает



G-0						
.	.	.	.	.	.	∞
↑new						

G-0						
.	.	.	.	.	.	∞
↑new						

G-0						
A	B	C	.	.	.	∞
			↑new			

G-0

.	.	.	.	.	.	∞
↑new						

G-0

A	B	C	.	.	.	∞
			↑new			

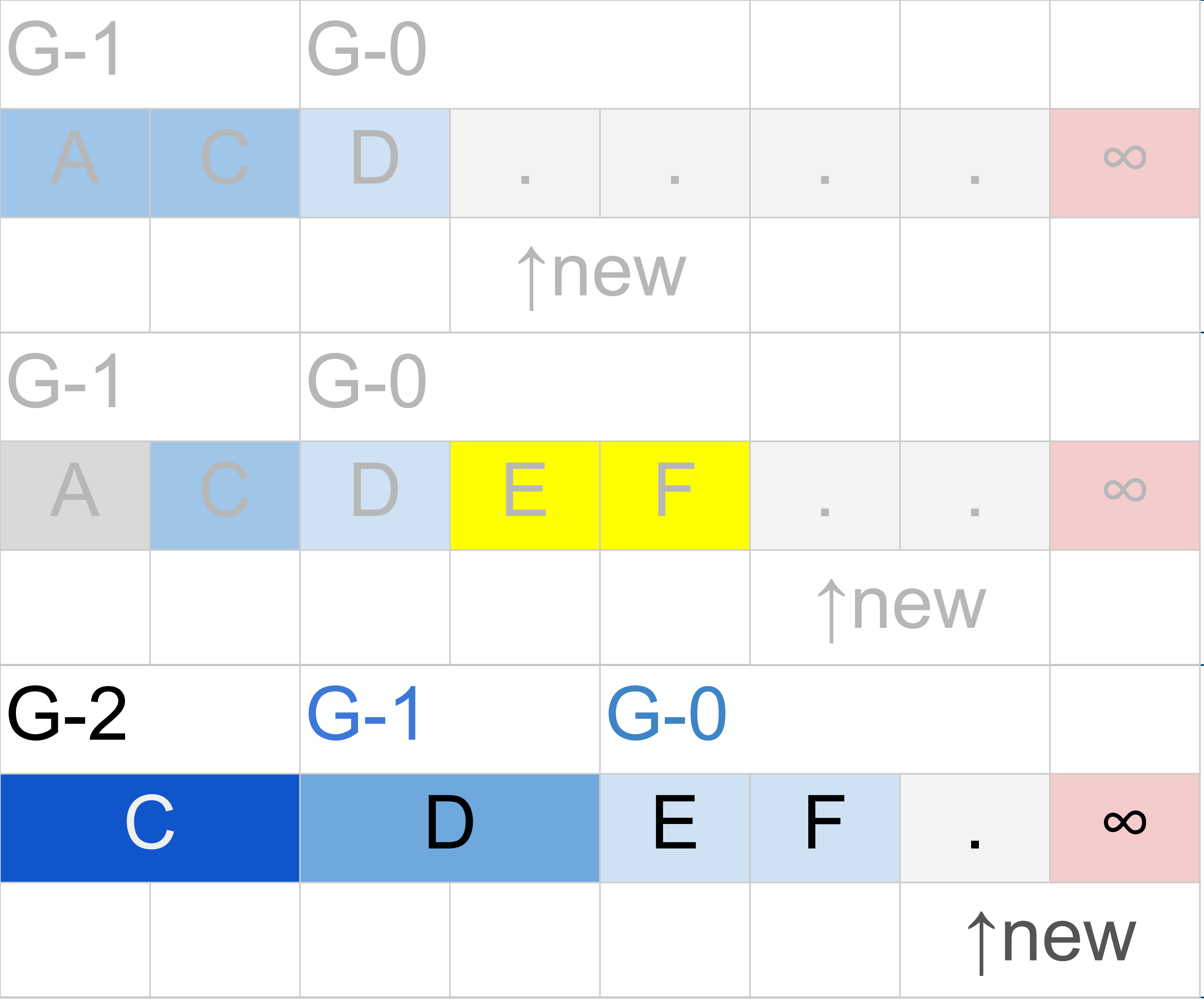
G-0

A	B	C	D	.	.	∞
				↑new		

G-1		G-0					
A	C	D	.	.	.	.	∞
			↑new				

G-1		G-0					
A	C	D	.	.	.	.	∞
			↑new				

G-1		G-0					
A	C	D	E	F	.	.	∞
					↑new		


G-1
G-0
A
C
D
.
.
.
.
∞
↑new
G-1
G-0
A
C
D
E
F
.
.
∞
↑new
G-2
G-1
G-0
C
D
E
F
.
∞
↑new

Виды GC

Workstation

- одна куча на всё приложение
- приоритет сборки - уровня пользовательского потока

Server

- куча на каждое ядро
- сборка в потоке с высоким приоритетом
- отдельный тред на ядро для сборщика

Виды GC

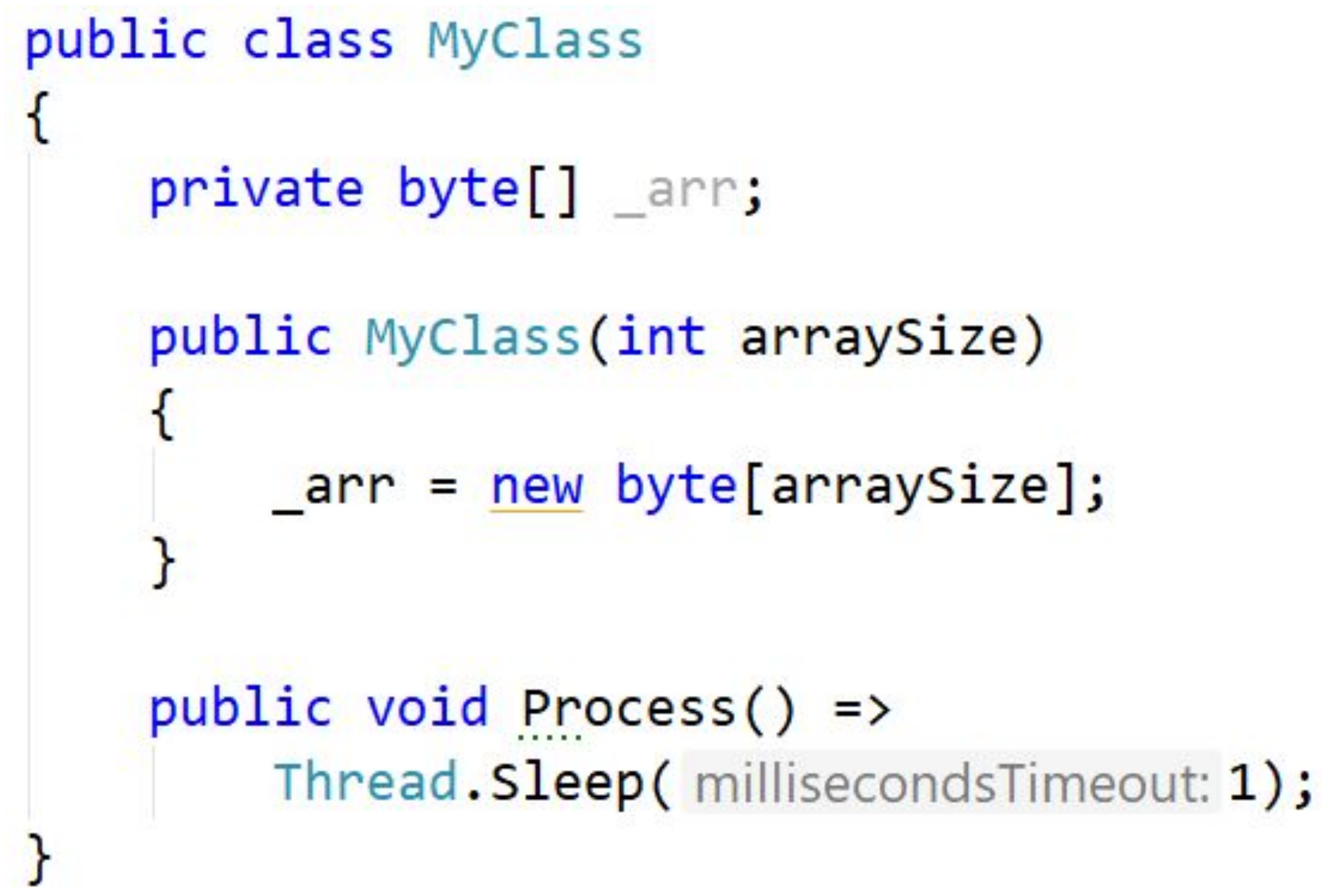
Workstation

- одна куча на всё приложение
- приоритет сборки - уровня пользовательского потока

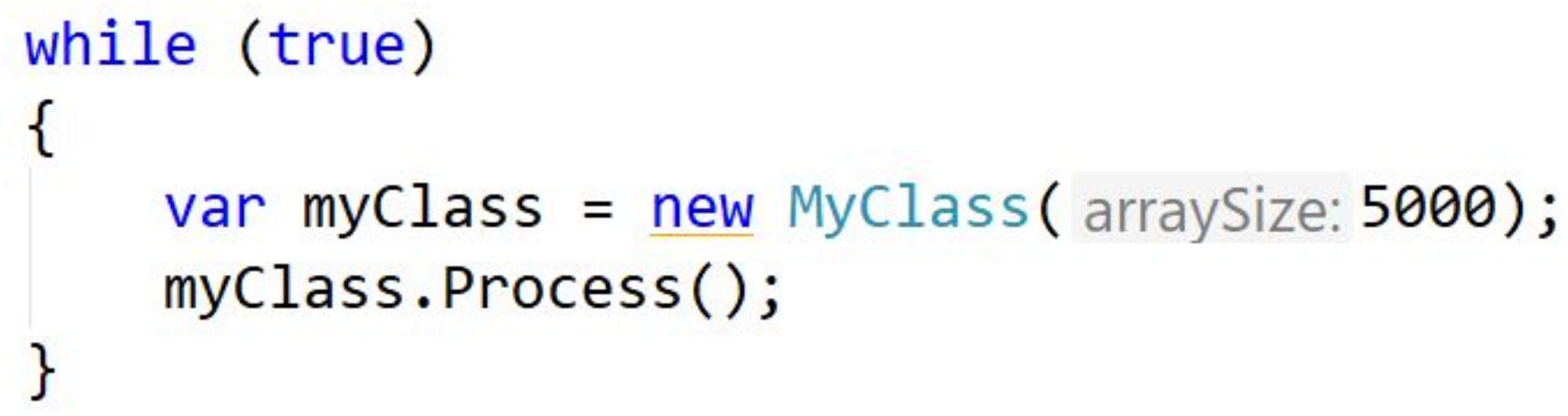
Server

- куча на каждое ядро
- сборка в потоке с высоким приоритетом
- отдельный тред на ядро для сборщика

```
<configuration>
   <runtime>
       <gcServer enabled="true"/>
   </runtime>
</configuration>
```

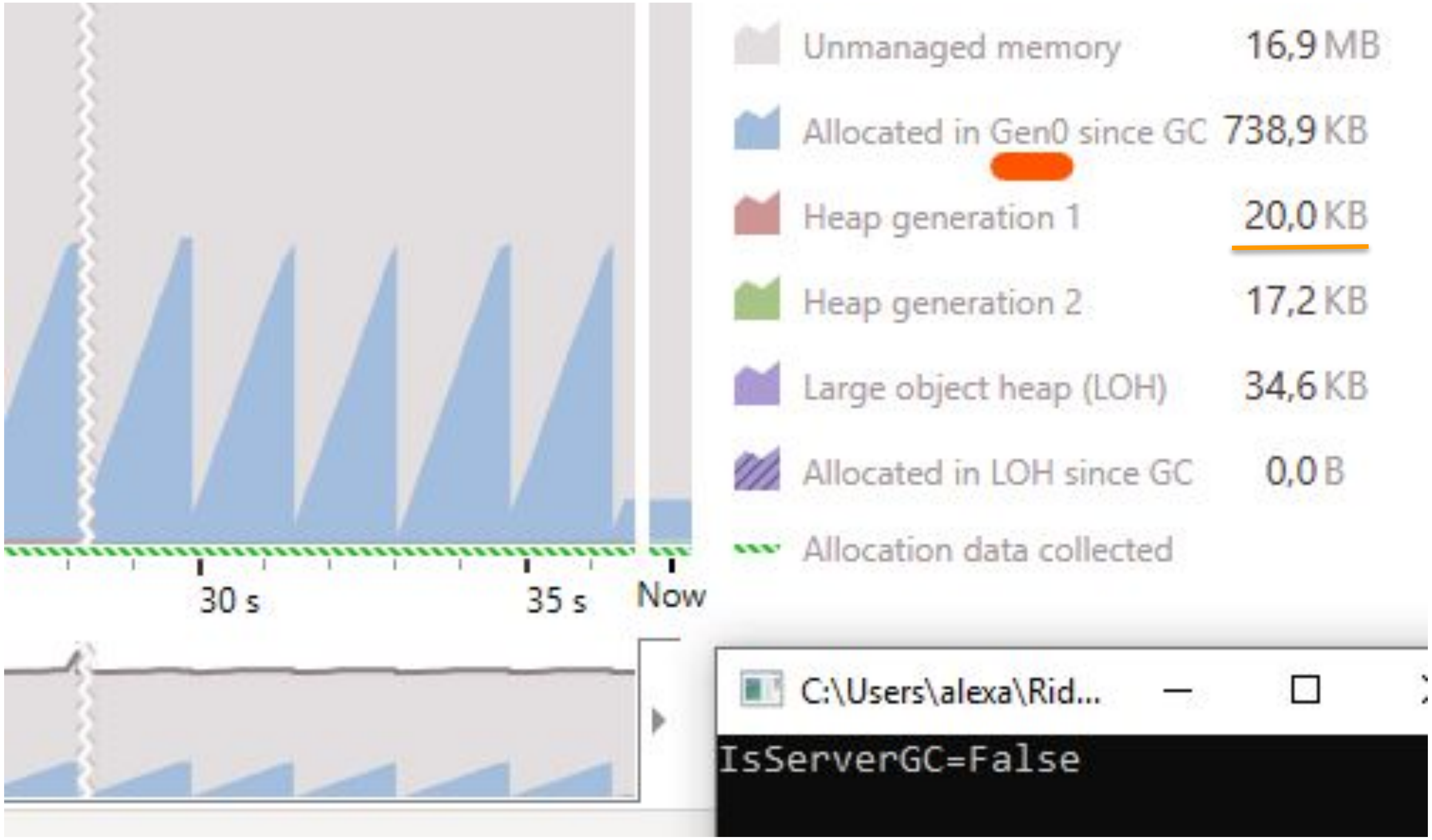


```
public class MyClass
{
    private byte[] _arr;

    public MyClass(int arraySize)
    {
        _arr = new byte[arraySize];
    }

    public void Process() =>
        Thread.Sleep( millisecondsTimeout: 1);
}
```



```
while (true)
{
    var myClass = new MyClass( arraySize: 5000);
    myClass.Process();
}
```

GC Workstation

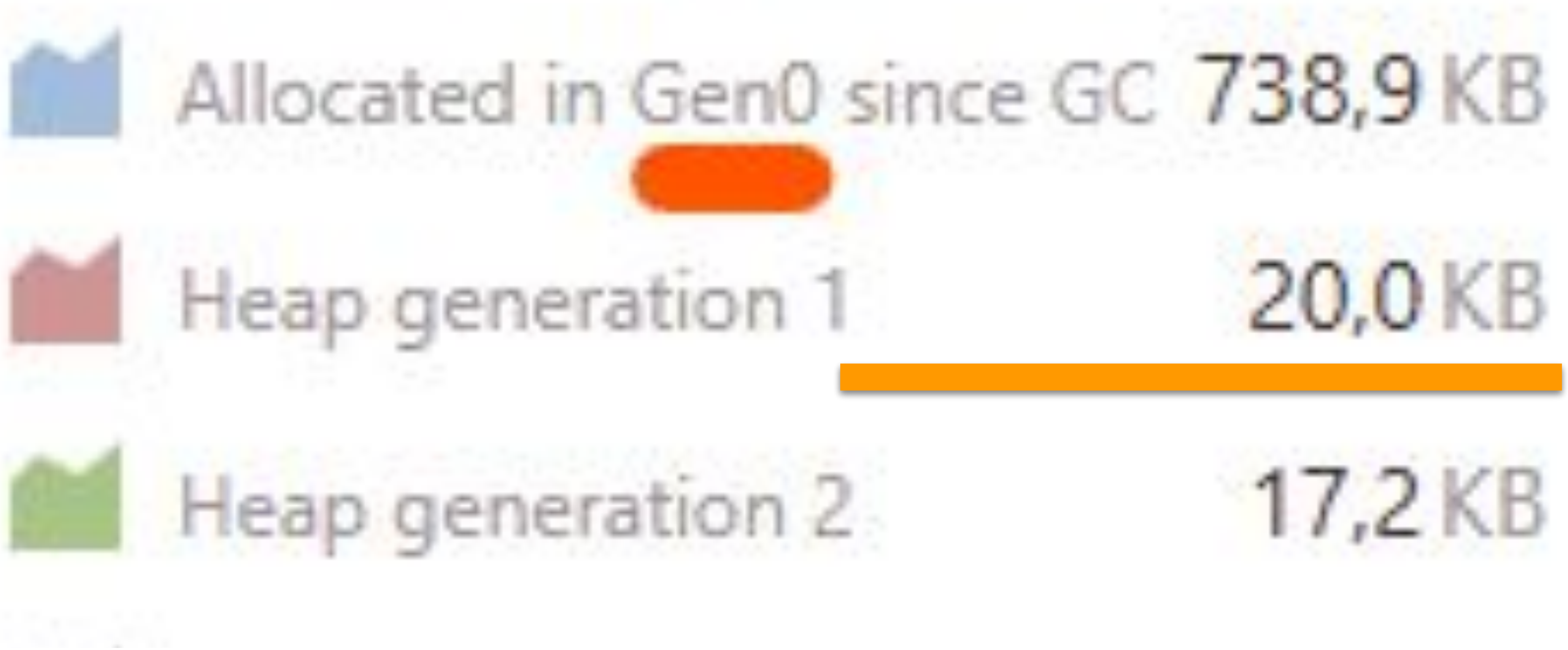


▨	Unmanaged memory	16,9 MB
▨	Allocated in Gen0 since GC	738,9 KB
▨	Heap generation 1	20,0 KB
▨	Heap generation 2	17,2 KB
▨	Large object heap (LOH)	34,6 KB
▨	Allocated in LOH since GC	0,0 B
▨	Allocation data collected	

C:\Users\alexa\Rid...

```
IsServerGC=False
```

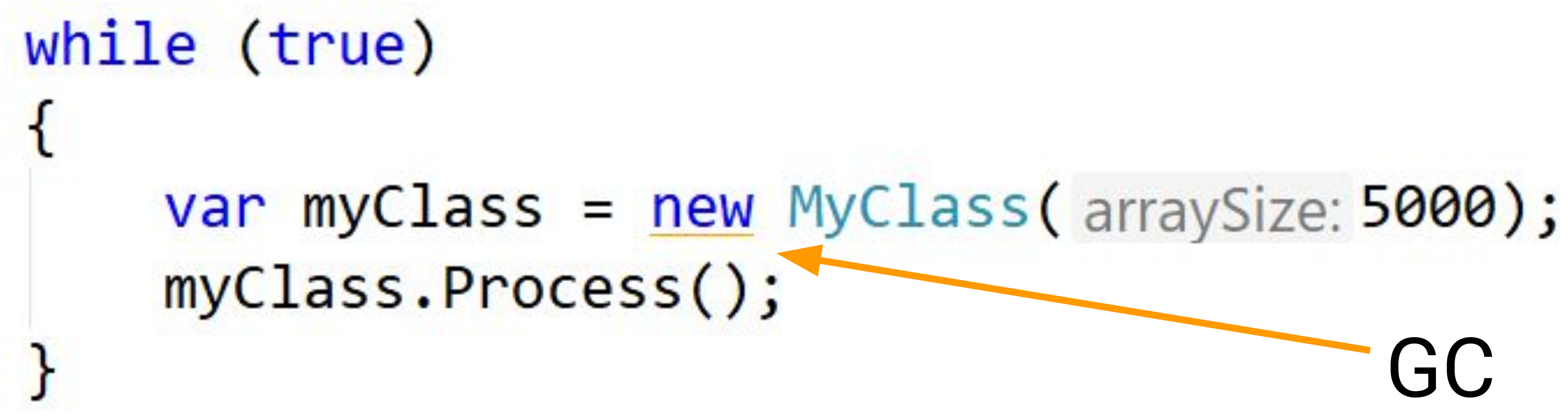
GC Workstation



Allocated in Gen0 since GC 738,9 KB

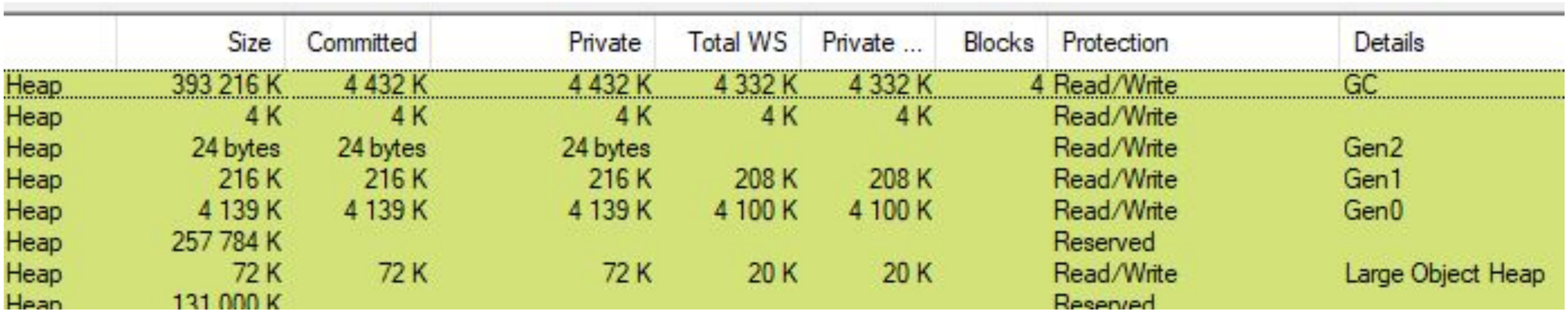
Heap generation 1 20,0 KB

Heap generation 2 17,2 KB



```
while (true)
{
    var myClass = new MyClass( arraySize: 5000);
    myClass.Process();
}
```

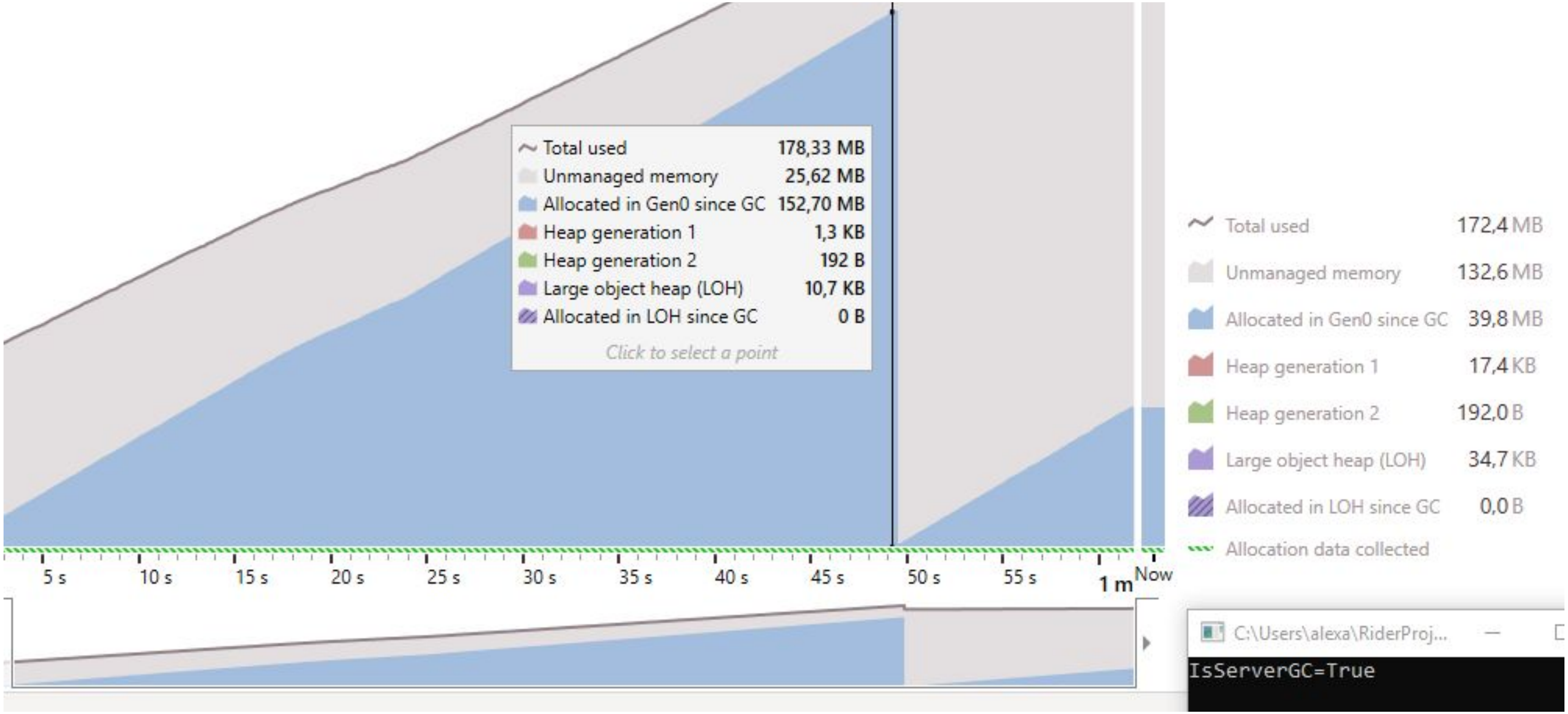
GC Workstation



	Size	Committed	Private	Total WS	Private ...	Blocks	Protection	Details
Heap	393 216 K	4 432 K	4 432 K	4 332 K	4 332 K	4	Read/Write	GC
Heap	4 K	4 K	4 K	4 K	4 K		Read/Write	
Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen2
Heap	216 K	216 K	216 K	208 K	208 K		Read/Write	Gen1
Heap	4 139 K	4 139 K	4 139 K	4 100 K	4 100 K		Read/Write	Gen0
Heap	257 784 K						Reserved	
Heap	72 K	72 K	72 K	20 K	20 K		Read/Write	Large Object Heap
Heap	131 000 K						Reserved	

https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap

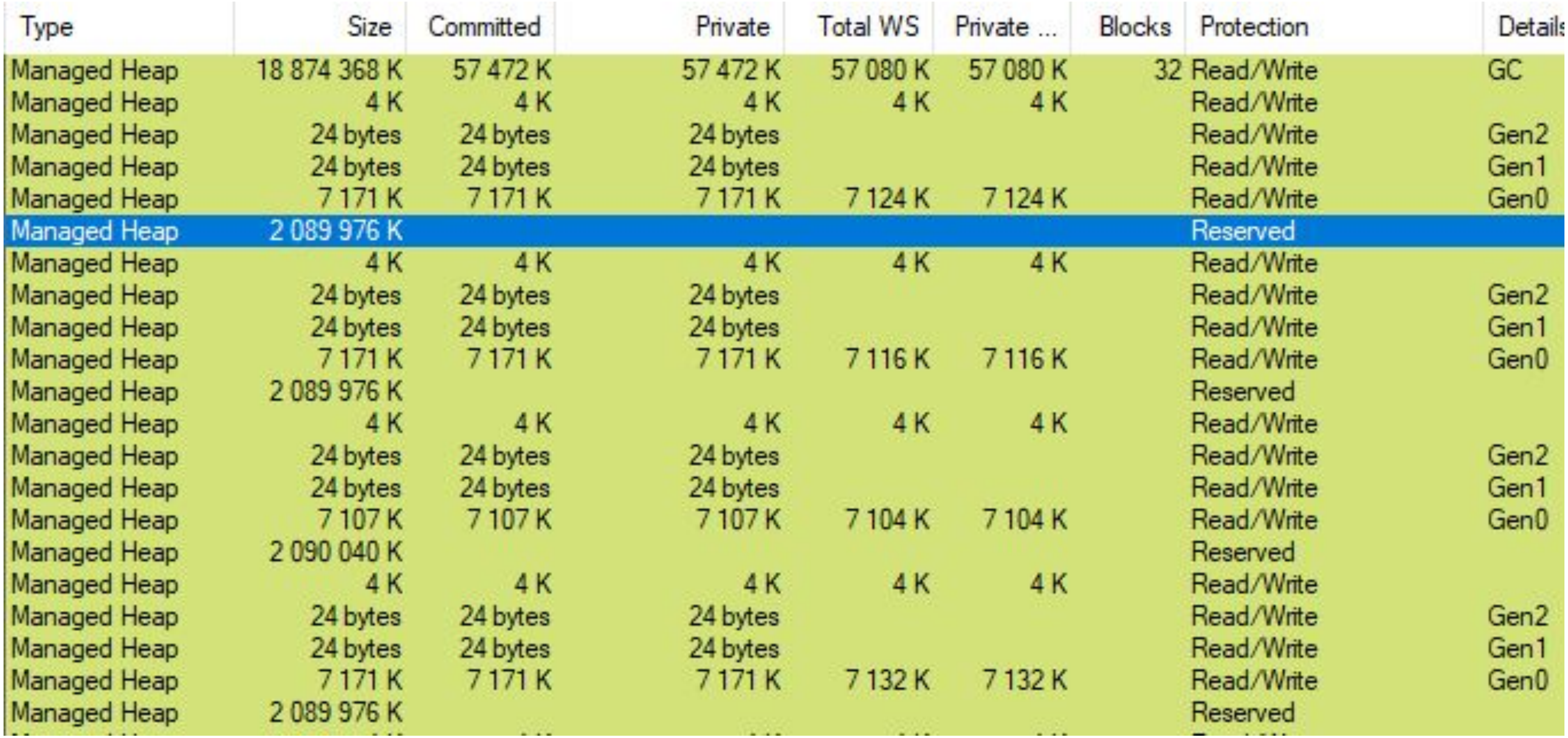
GC Server



Total used 178,33 MB
Unmanaged memory 25,62 MB
Allocated in Gen0 since GC 152,70 MB
Heap generation 1 1,3 KB
Heap generation 2 192 B
Large object heap (LOH) 10,7 KB
Allocated in LOH since GC 0 B
Click to select a point
5 s
10 s
15 s
20 s
25 s
30 s
35 s
40 s
45 s
50 s
55 s
1 m
Now
Total used 172,4 MB
Unmanaged memory 132,6 MB
Allocated in Gen0 since GC 39,8 MB
Heap generation 1 17,4 KB
Heap generation 2 192,0 B
Large object heap (LOH) 34,7 KB
Allocated in LOH since GC 0,0 B
Allocation data collected
C:\Users\alexa\RiderProj...

```
IsServerGC=True
```

GC Server

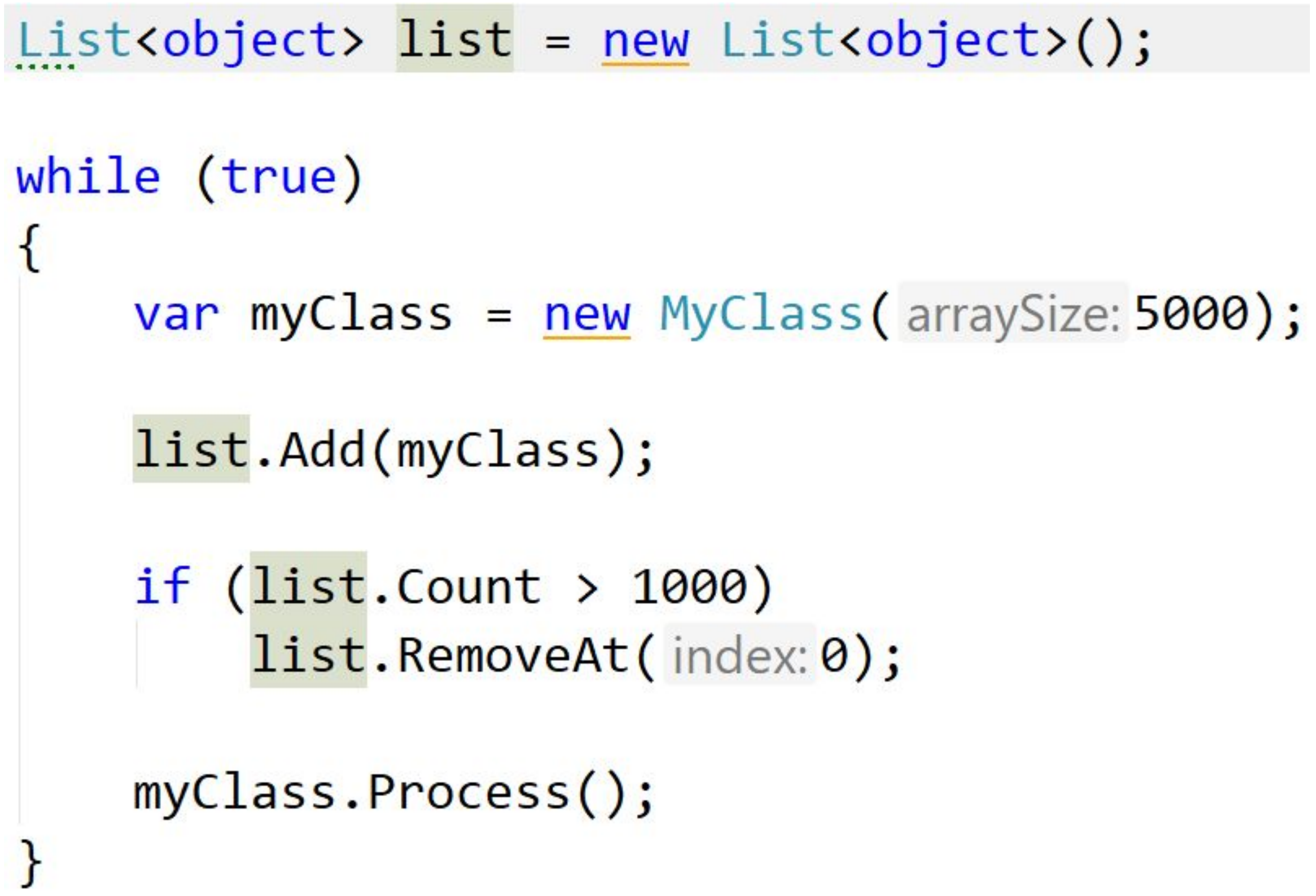


Type	Size	Committed	Private	Total WS	Private ...	Blocks	Protection	Details
Managed Heap	18 874 368 K	57 472 K	57 472 K	57 080 K	57 080 K	32	Read/Write	GC
Managed Heap	4 K	4 K	4 K	4 K	4 K		Read/Write	
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen2
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen1
Managed Heap	7 171 K	7 171 K	7 171 K	7 124 K	7 124 K		Read/Write	Gen0
Managed Heap	2 089 976 K						Reserved	
Managed Heap	4 K	4 K	4 K	4 K	4 K		Read/Write	
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen2
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen1
Managed Heap	7 171 K	7 171 K	7 171 K	7 116 K	7 116 K		Read/Write	Gen0
Managed Heap	2 089 976 K						Reserved	
Managed Heap	4 K	4 K	4 K	4 K	4 K		Read/Write	
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen2
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen1
Managed Heap	7 107 K	7 107 K	7 107 K	7 104 K	7 104 K		Read/Write	Gen0
Managed Heap	2 090 040 K						Reserved	
Managed Heap	4 K	4 K	4 K	4 K	4 K		Read/Write	
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen2
Managed Heap	24 bytes	24 bytes	24 bytes				Read/Write	Gen1
Managed Heap	7 171 K	7 171 K	7 171 K	7 132 K	7 132 K		Read/Write	Gen0
Managed Heap	2 089 976 K						Reserved	

Корни - не мусор

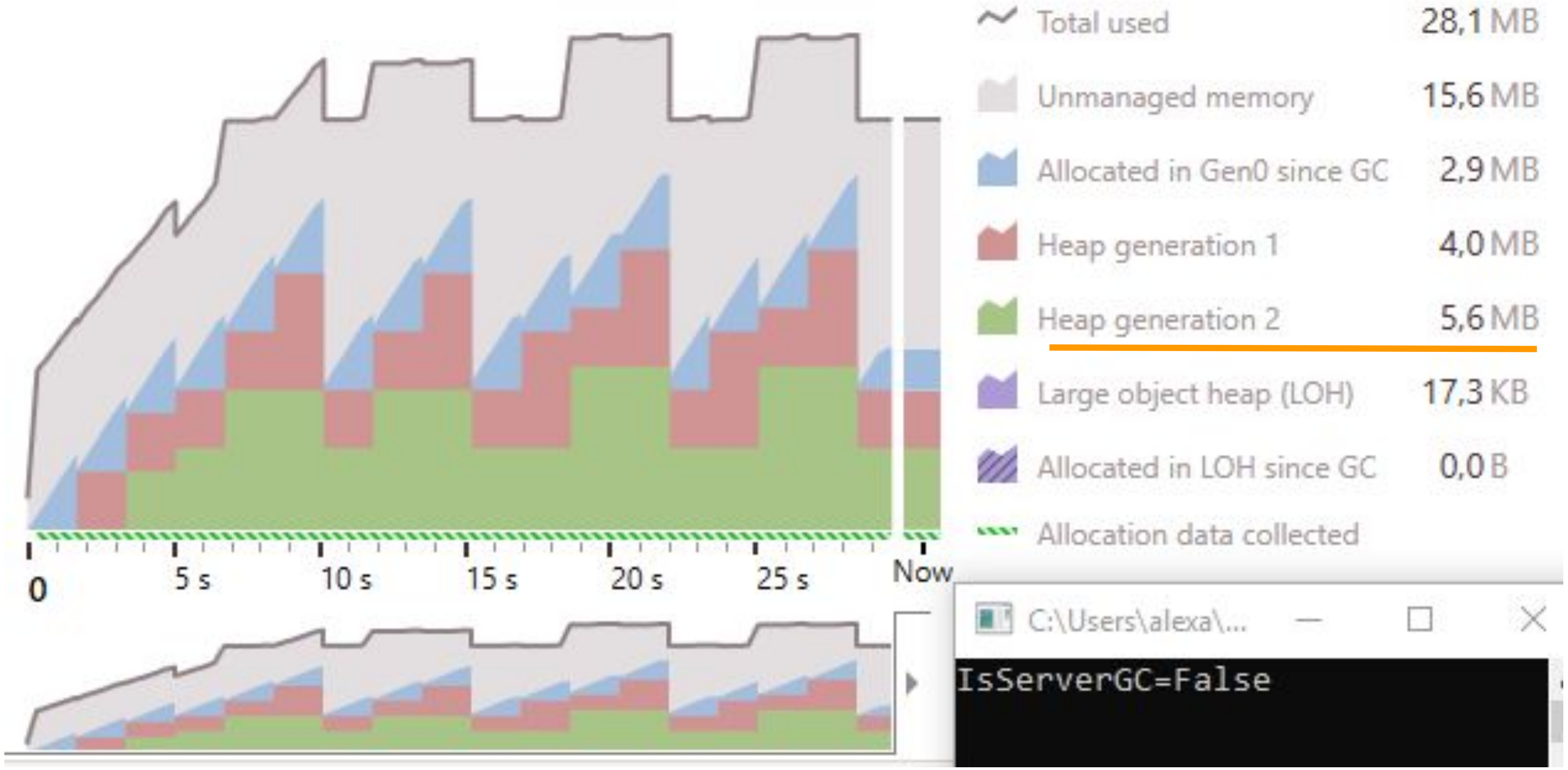
- статические поля
- переменные
- события
- очередь финализации

Generation 2



```
List<object> list = new List<object>();

while (true)
{
    var myClass = new MyClass( arraySize: 5000);

    list.Add(myClass);

    if (list.Count > 1000)
        list.RemoveAt( index: 0);

    myClass.Process();
}
```

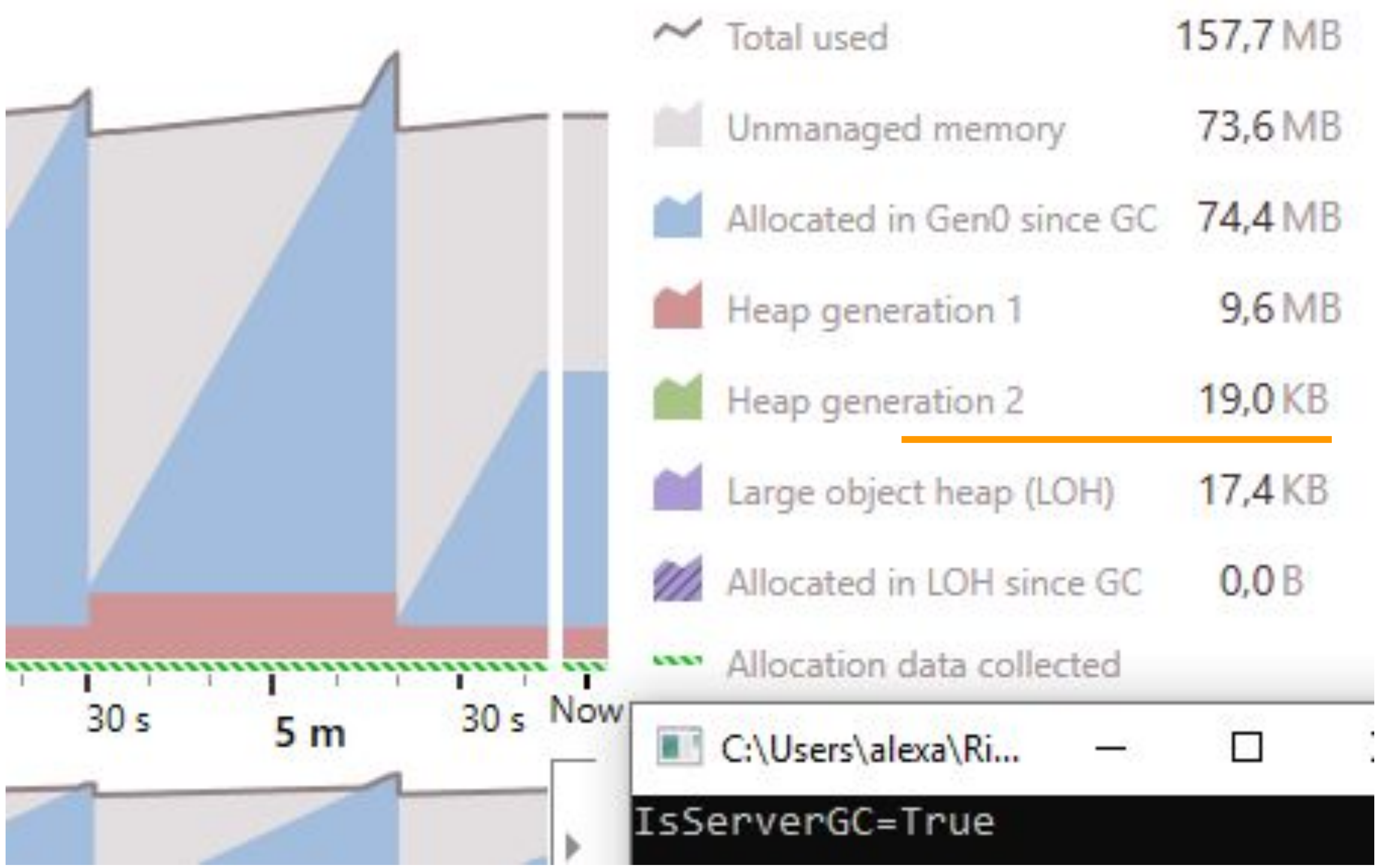
GC Workstation (G2)



〰	Total used	28,1 MB
⬜	Unmanaged memory	15,6 MB
🟦	Allocated in Gen0 since GC	2,9 MB
🟥	Heap generation 1	4,0 MB
🟩	Heap generation 2	5,6 MB
🟪	Large object heap (LOH)	17,3 KB
▨	Allocated in LOH since GC	0,0 B
〰	Allocation data collected	

```
IsServerGC=False
```

GC Server (G2?)

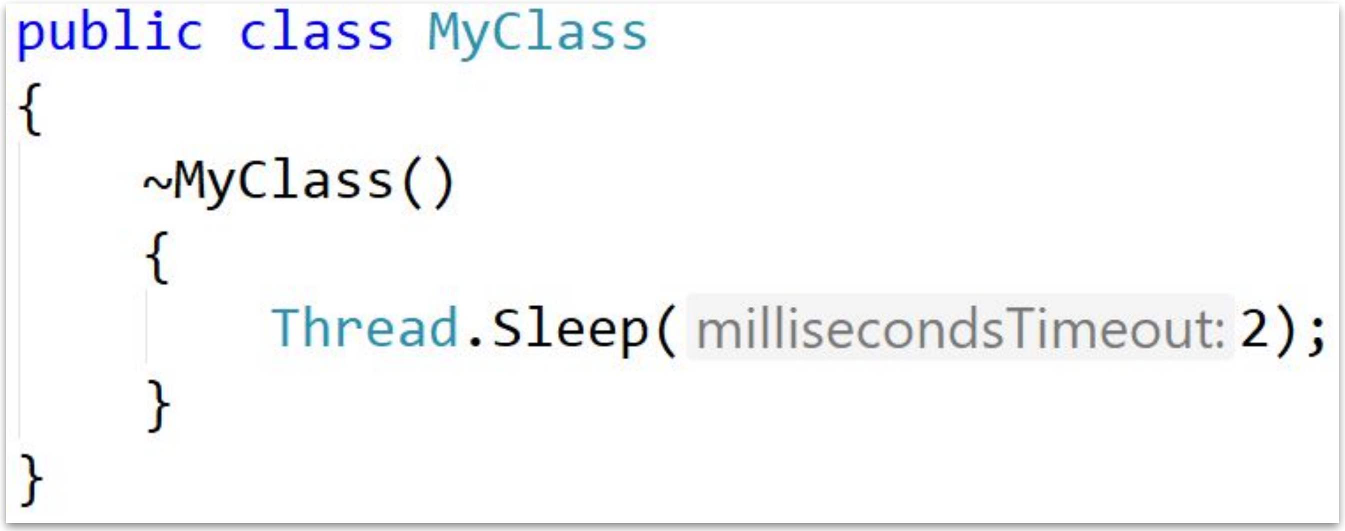


〰	Total used	157,7 MB
⬜	Unmanaged memory	73,6 MB
🟦	Allocated in Gen0 since GC	74,4 MB
🟥	Heap generation 1	9,6 MB
🟩	Heap generation 2	19,0 KB
🟪	Large object heap (LOH)	17,4 KB
▨	Allocated in LOH since GC	0,0 B
▨	Allocation data collected	

C:\Users\alexa\Ri...

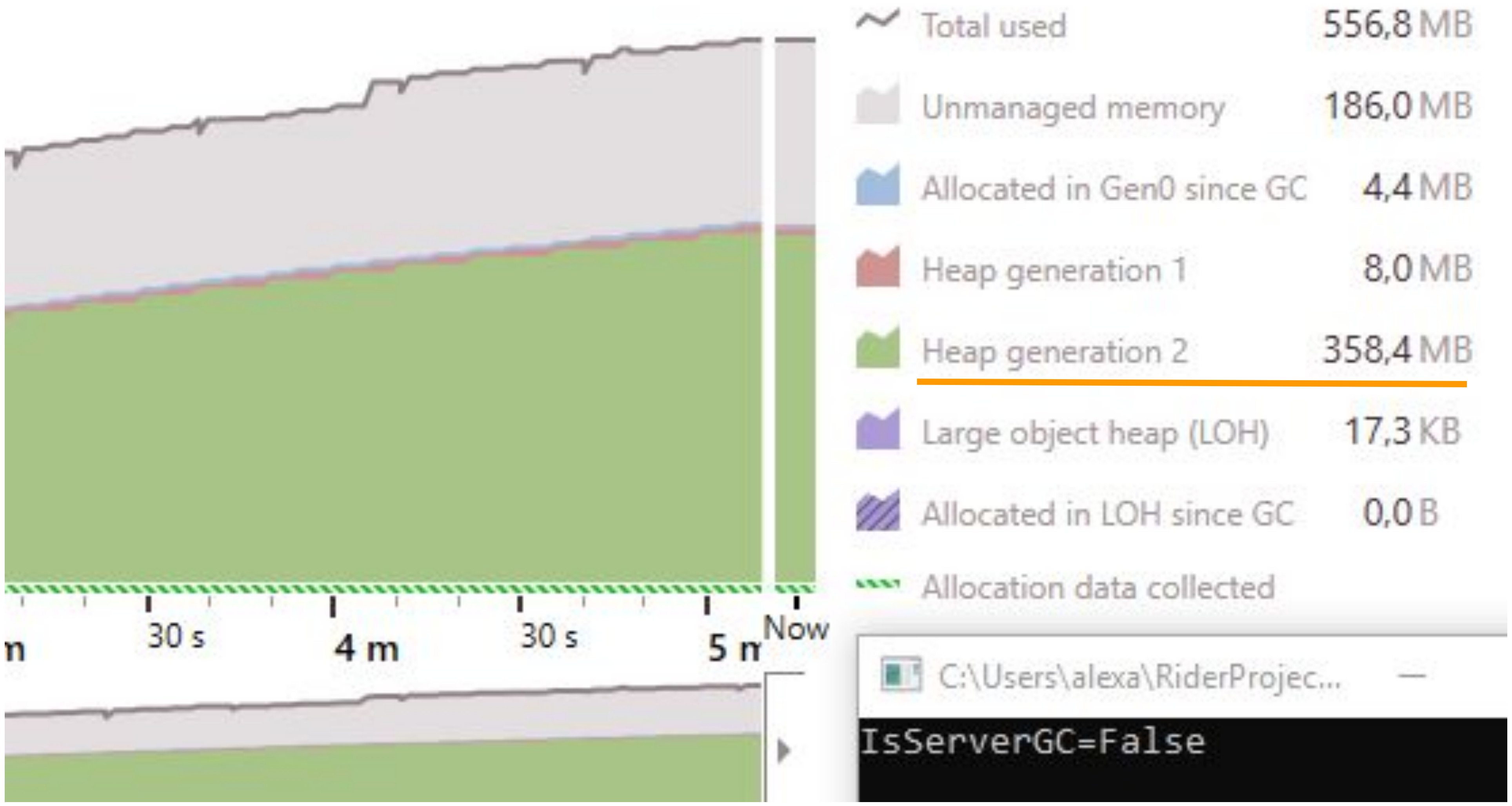
```
IsServerGC=True
```

Finalize



```
public class MyClass
{
    ~MyClass()
    {
        Thread.Sleep( millisecondsTimeout: 2);
    }
}
```

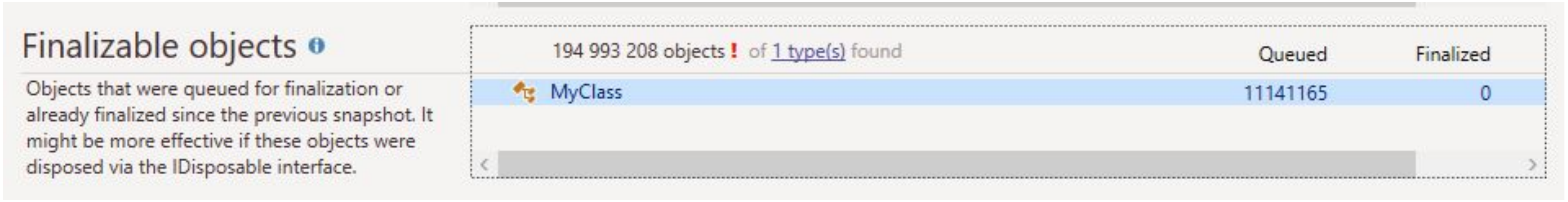
Finalize



〰	Total used	556,8 MB
⬜	Unmanaged memory	186,0 MB
🟦	Allocated in Gen0 since GC	4,4 MB
🟥	Heap generation 1	8,0 MB
🟩	Heap generation 2	358,4 MB
🟪	Large object heap (LOH)	17,3 KB
▨	Allocated in LOH since GC	0,0 B
┅	Allocation data collected	

```
IsServerGC=False
```

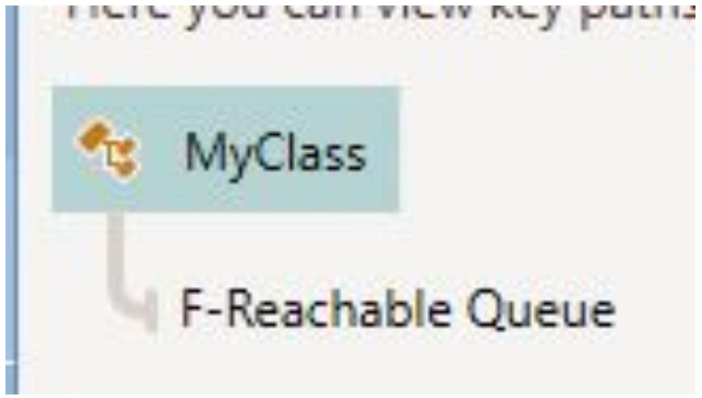
Finalize


Finalizable objects

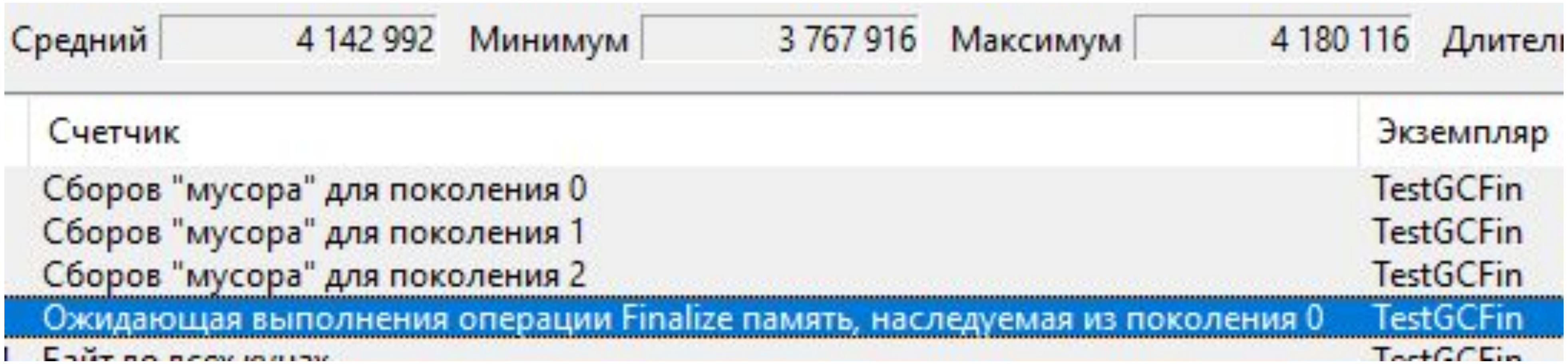
Objects that were queued for finalization or already finalized since the previous snapshot. It might be more effective if these objects were disposed via the IDisposable interface.

194 993 208 objects ! of 1 type(s) found

	Queued	Finalized
MyClass	11141165	0



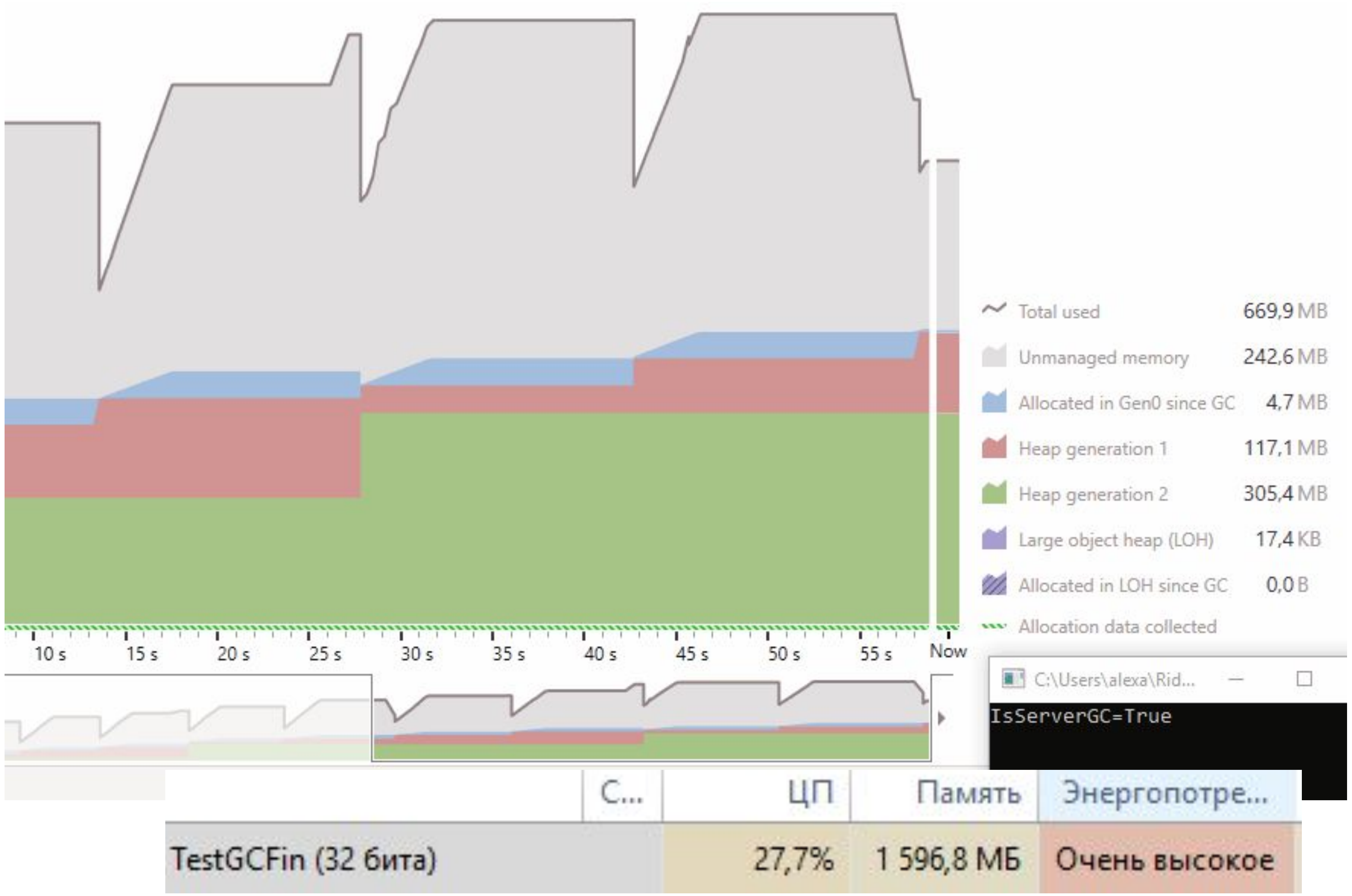
Finalize



Средний 4 142 992 Минимум 3 767 916 Максимум 4 180 116 Длител

Счетчик	Экземпляр
Сборов "мусора" для поколения 0	TestGCFin
Сборов "мусора" для поколения 1	TestGCFin
Сборов "мусора" для поколения 2	TestGCFin
Ожидающая выполнения операции Finalize память, наследуемая из поколения 0	TestGCFin
Байт во всех кучах	TestGCFin

Finalize (GC Server)



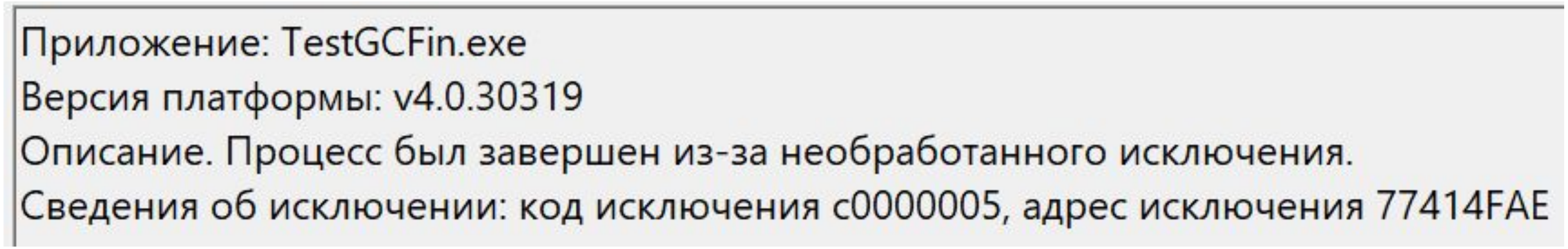
Total used	669,9 MB
Unmanaged memory	242,6 MB
Allocated in Gen0 since GC	4,7 MB
Heap generation 1	117,1 MB
Heap generation 2	305,4 MB
Large object heap (LOH)	17,4 KB
Allocated in LOH since GC	0,0 B
Allocation data collected	

C:\Users\alexa\Rid...

```
IsServerGC=True
```

	С...	ЦП	Память	Энергопотре...
TestGCFin (32 бита)		27,7%	1 596,8 МБ	Очень высокое

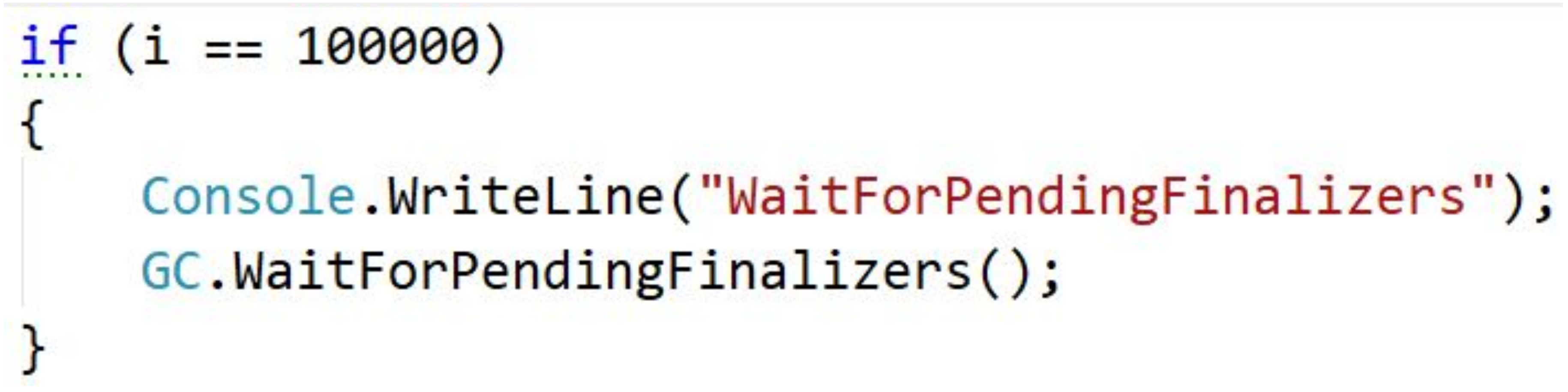
Finalize (GC Server)



```
Приложение: TestGCFin.exe
Версия платформы: v4.0.30319
Описание. Процесс был завершен из-за необработанного исключения.
Сведения об исключении: код исключения c0000005, адрес исключения 77414FAE
```

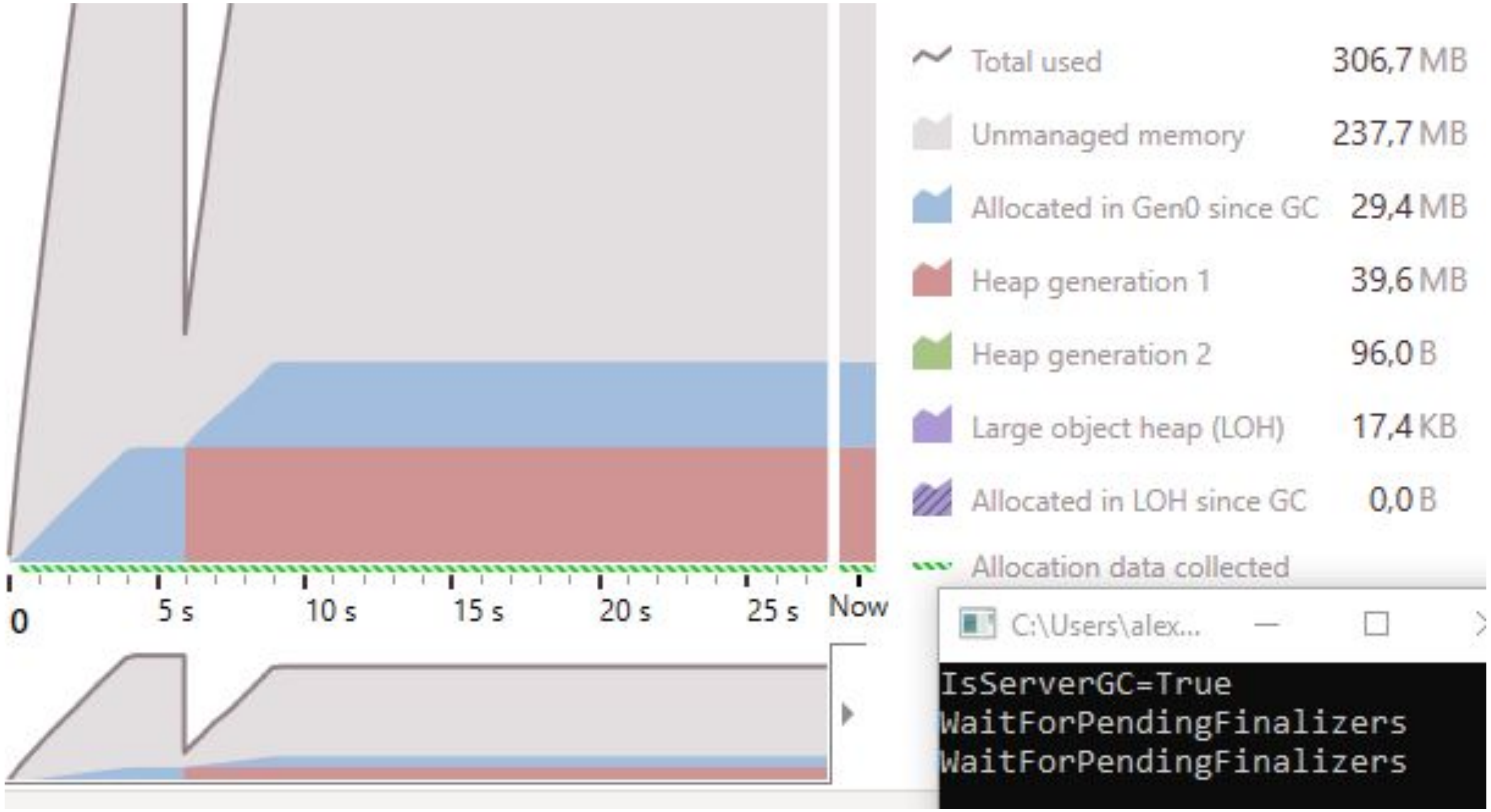
Finalize (GC Server)

```
Suspends the current thread until the
thread that is processing the queue
of finalizers has emptied that queue
```



```
if (i == 100000)
{
    Console.WriteLine("WaitForPendingFinalizers");
    GC.WaitForPendingFinalizers();
}
```

Finalize (GC Server)

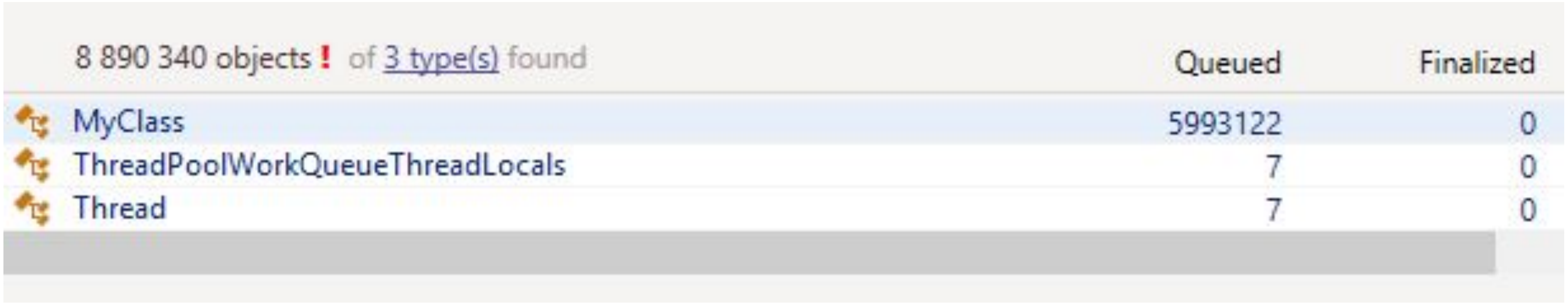


Total used	306,7 MB
Unmanaged memory	237,7 MB
Allocated in Gen0 since GC	29,4 MB
Heap generation 1	39,6 MB
Heap generation 2	96,0 B
Large object heap (LOH)	17,4 KB
Allocated in LOH since GC	0,0 B
Allocation data collected	

C:\Users\alex...

```
IsServerGC=True
WaitForPendingFinalizers
WaitForPendingFinalizers
```

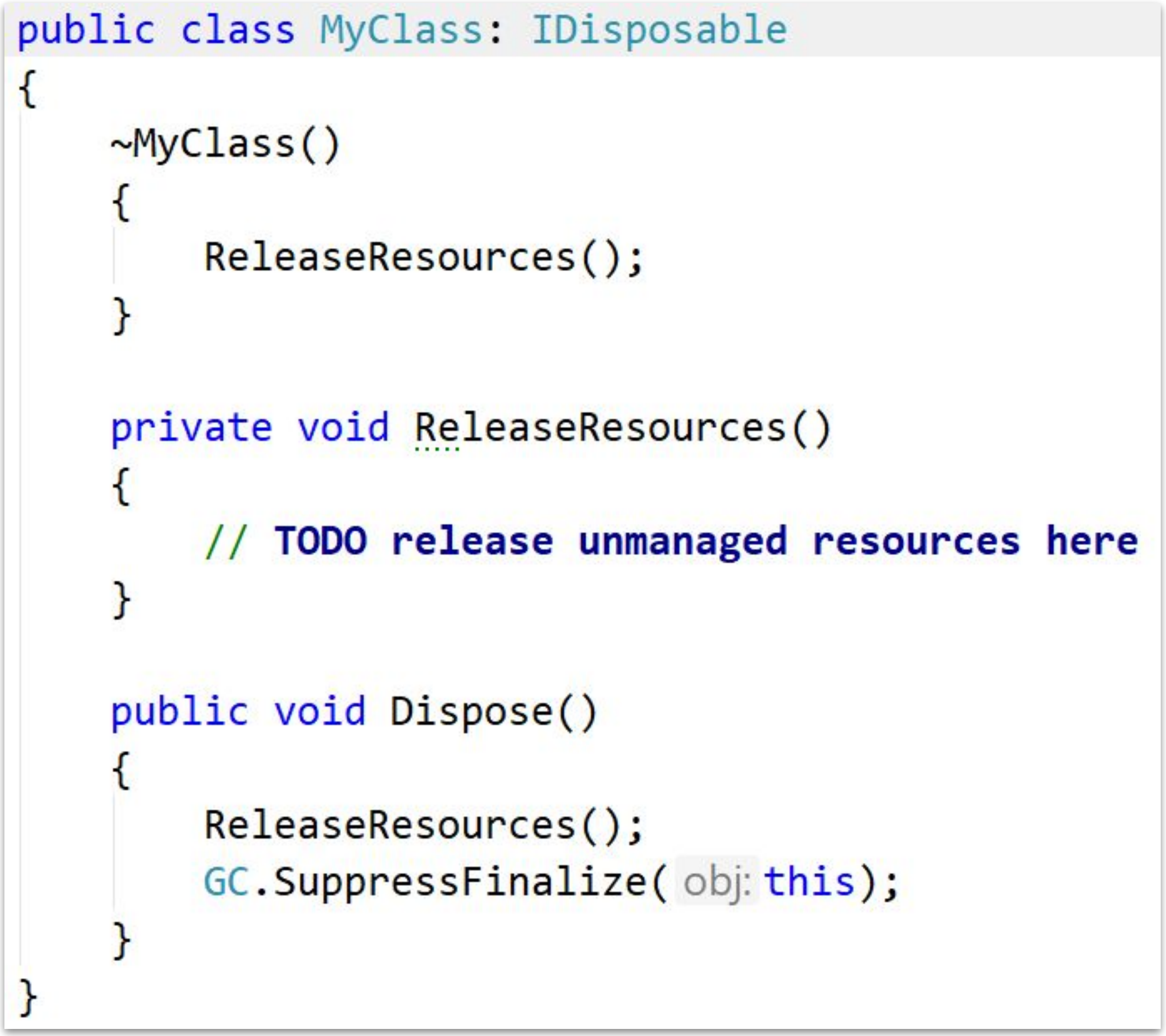
Finalize (GC Server)



8 890 340 objects ! of 3 type(s) found

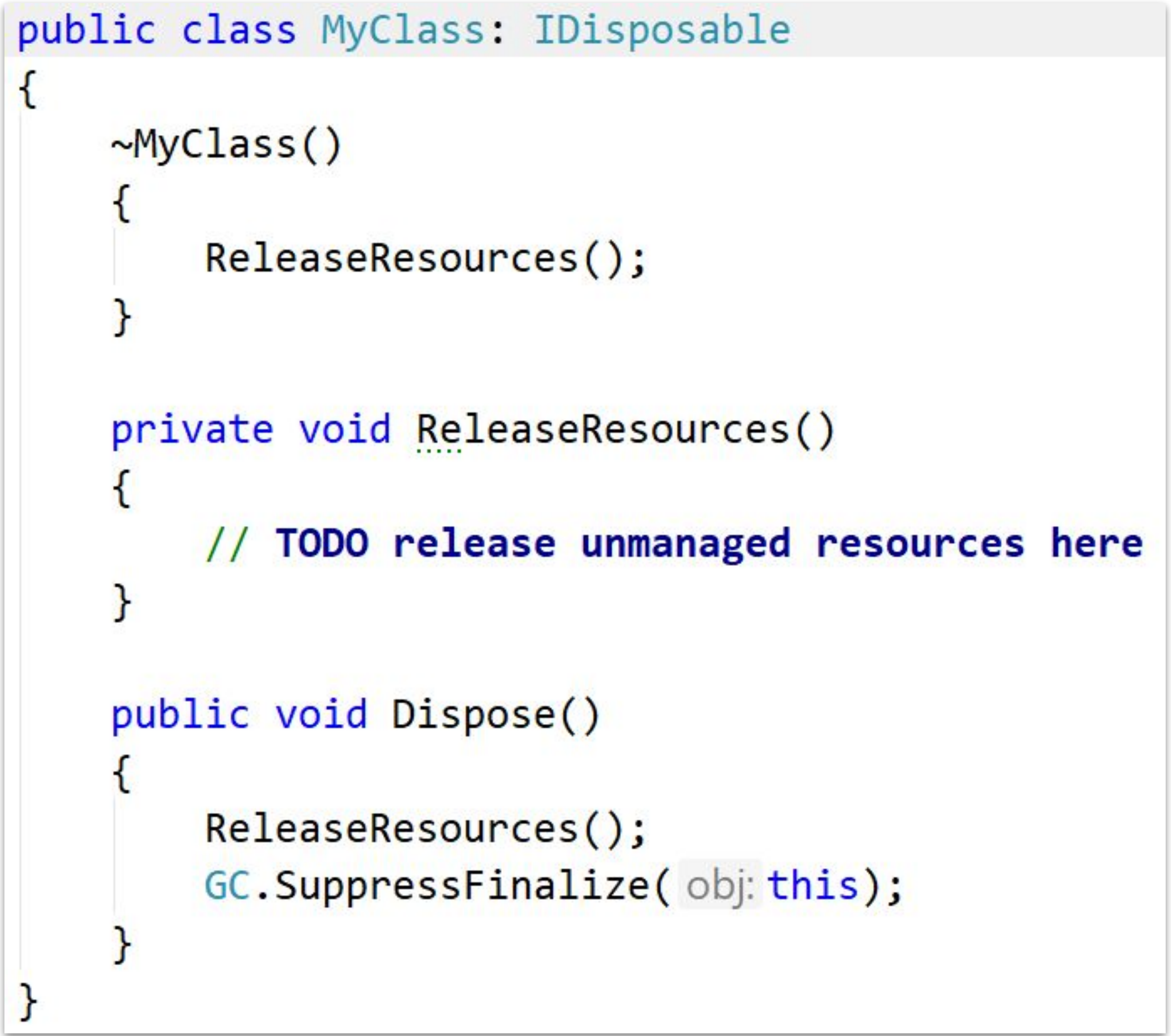
	Queued	Finalized
MyClass	5993122	0
ThreadPoolWorkQueueThreadLocals	7	0
Thread	7	0

Finalize (GC Server)

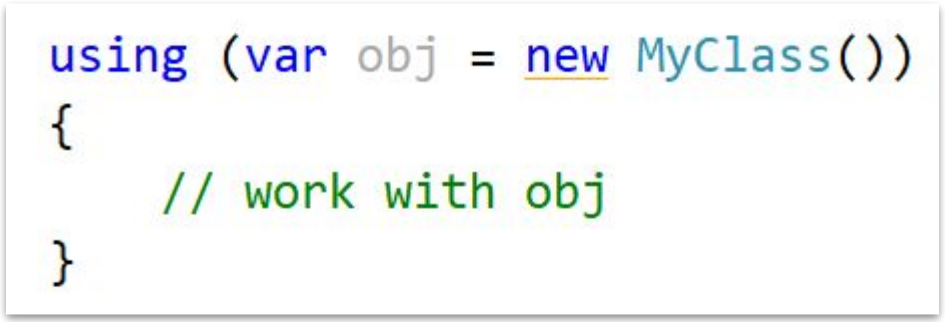


```
public class MyClass: IDisposable
{
    ~MyClass()
    {
        ReleaseResources();
    }

    private void ReleaseResources()
    {
        // TODO release unmanaged resources here
    }

    public void Dispose()
    {
        ReleaseResources();
        GC.SuppressFinalize( obj: this);
    }
}
```

Finalize (GC Server)

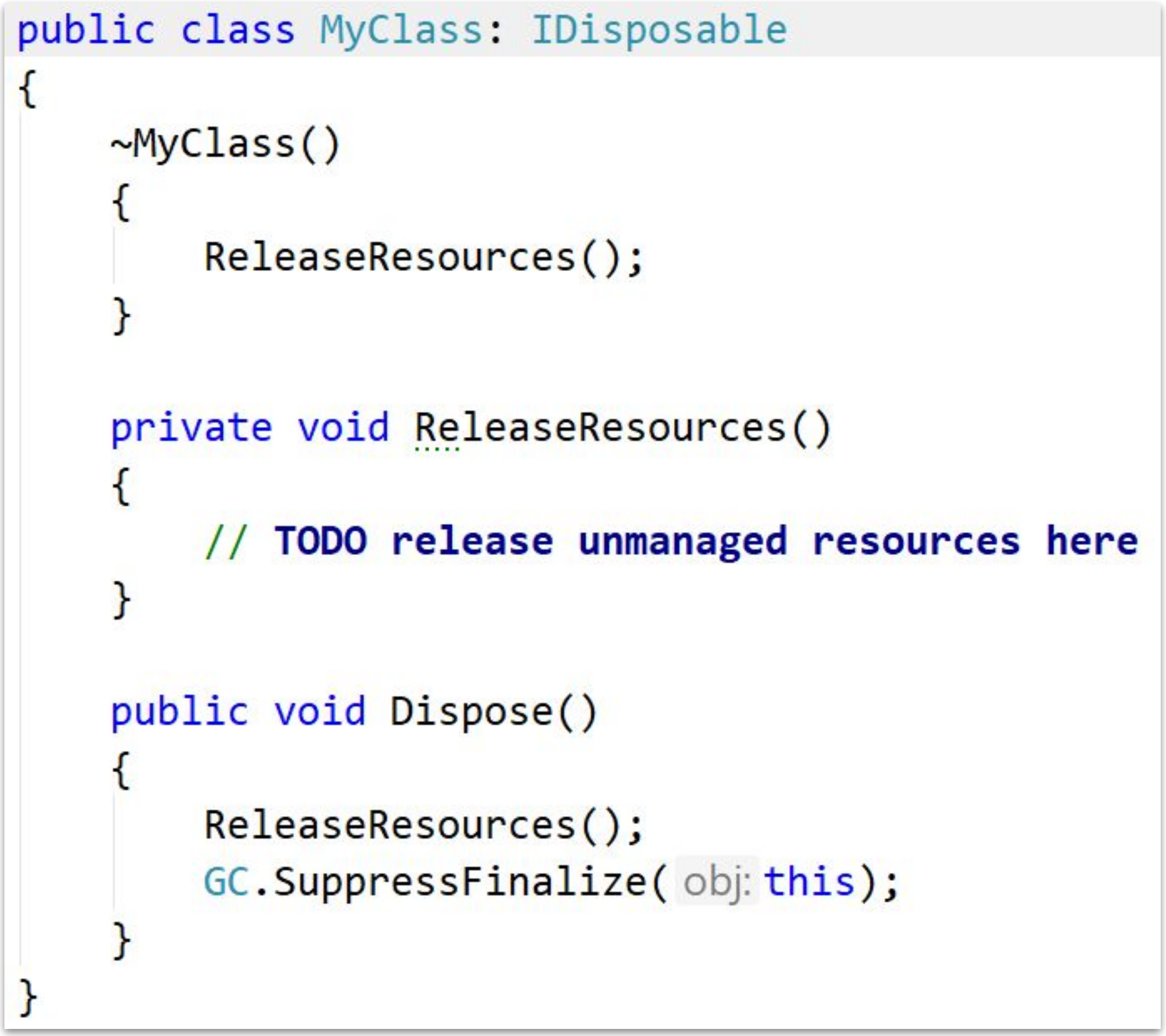


```
public class MyClass: IDisposable
{
    ~MyClass()
    {
        ReleaseResources();
    }

    private void ReleaseResources()
    {
        // TODO release unmanaged resources here
    }

    public void Dispose()
    {
        ReleaseResources();
        GC.SuppressFinalize( obj: this);
    }
}
```

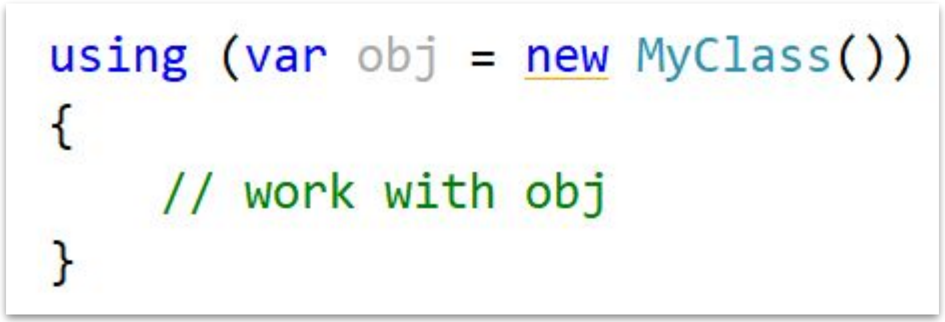


```
using (var obj = new MyClass())
{
    // work with obj
}
```

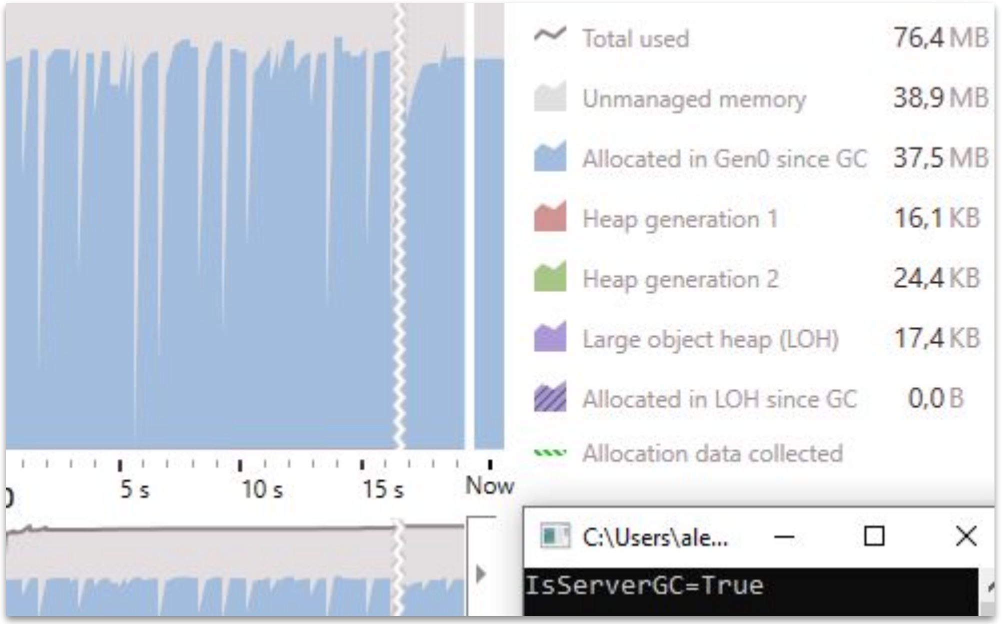
Finalize (GC Server)



```
public class MyClass: IDisposable
{
    ~MyClass()
    {
        ReleaseResources();
    }

    private void ReleaseResources()
    {
        // TODO release unmanaged resources here
    }

    public void Dispose()
    {
        ReleaseResources();
        GC.SuppressFinalize( obj: this);
    }
}
```



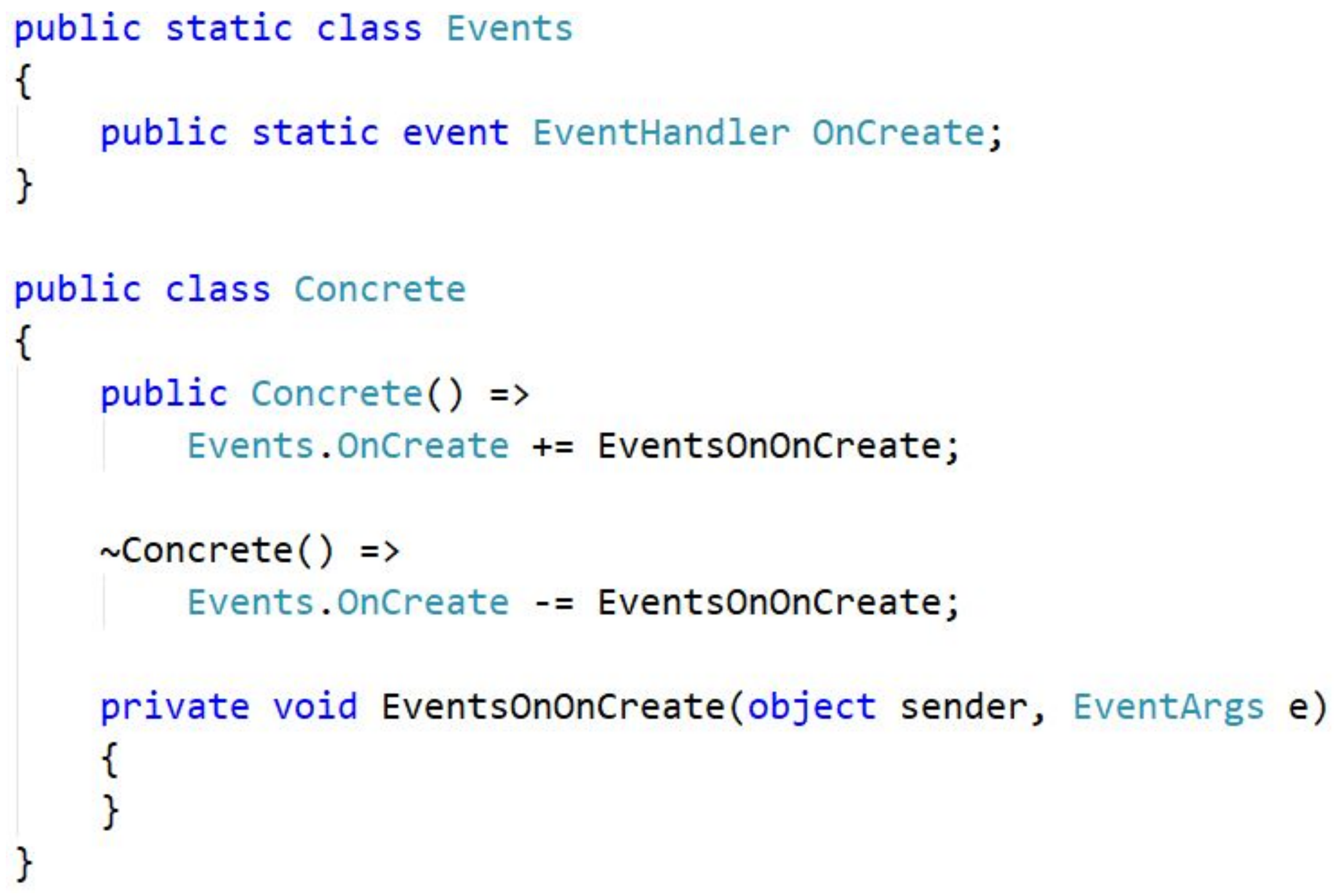
```
using (var obj = new MyClass())
{
    // work with obj
}
```



〰	Total used	76,4 MB
⬜	Unmanaged memory	38,9 MB
🟦	Allocated in Gen0 since GC	37,5 MB
🟥	Heap generation 1	16,1 KB
🟩	Heap generation 2	24,4 KB
🟪	Large object heap (LOH)	17,4 KB
▨	Allocated in LOH since GC	0,0 B
〰	Allocation data collected	

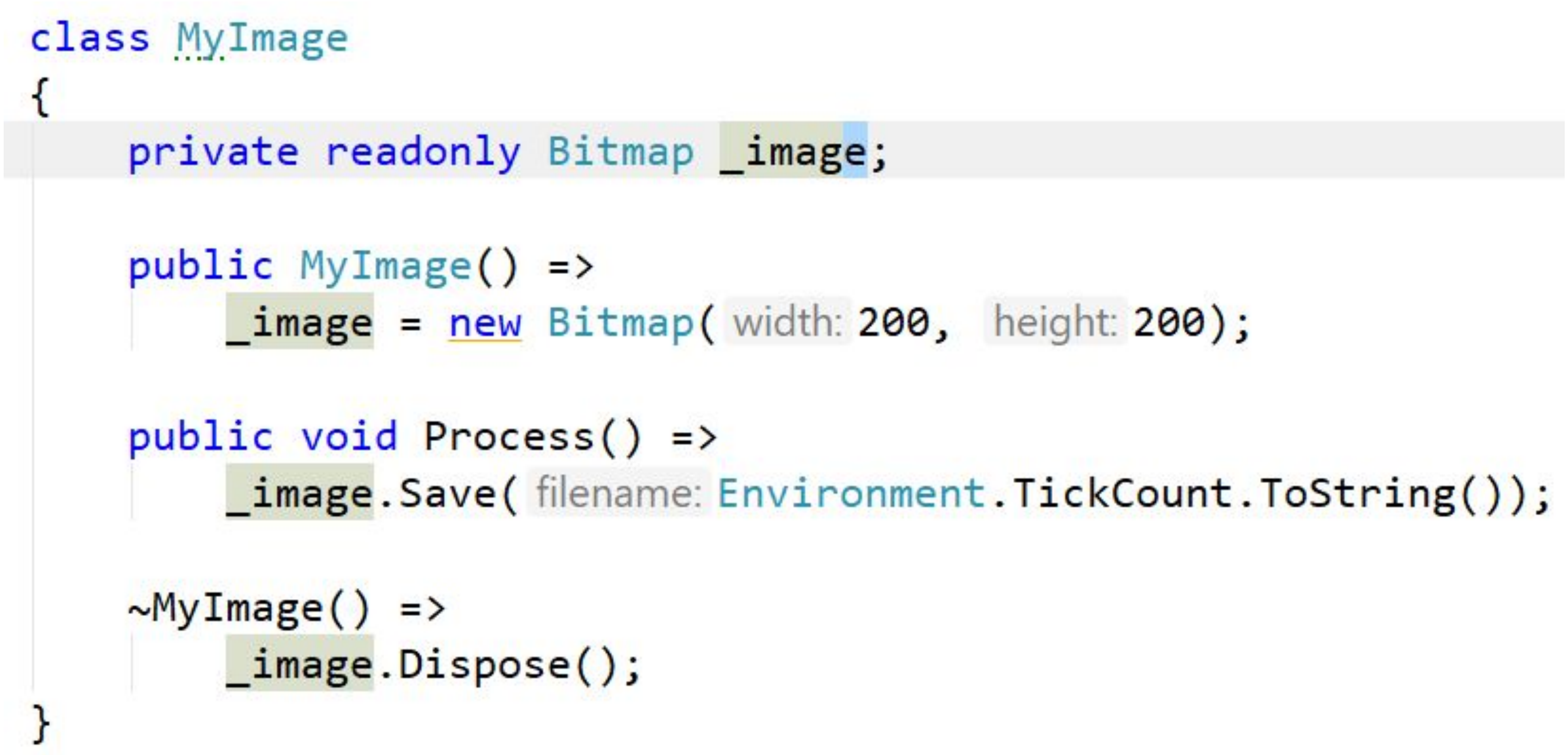
```
IsServerGC=True
```

Never Finalized



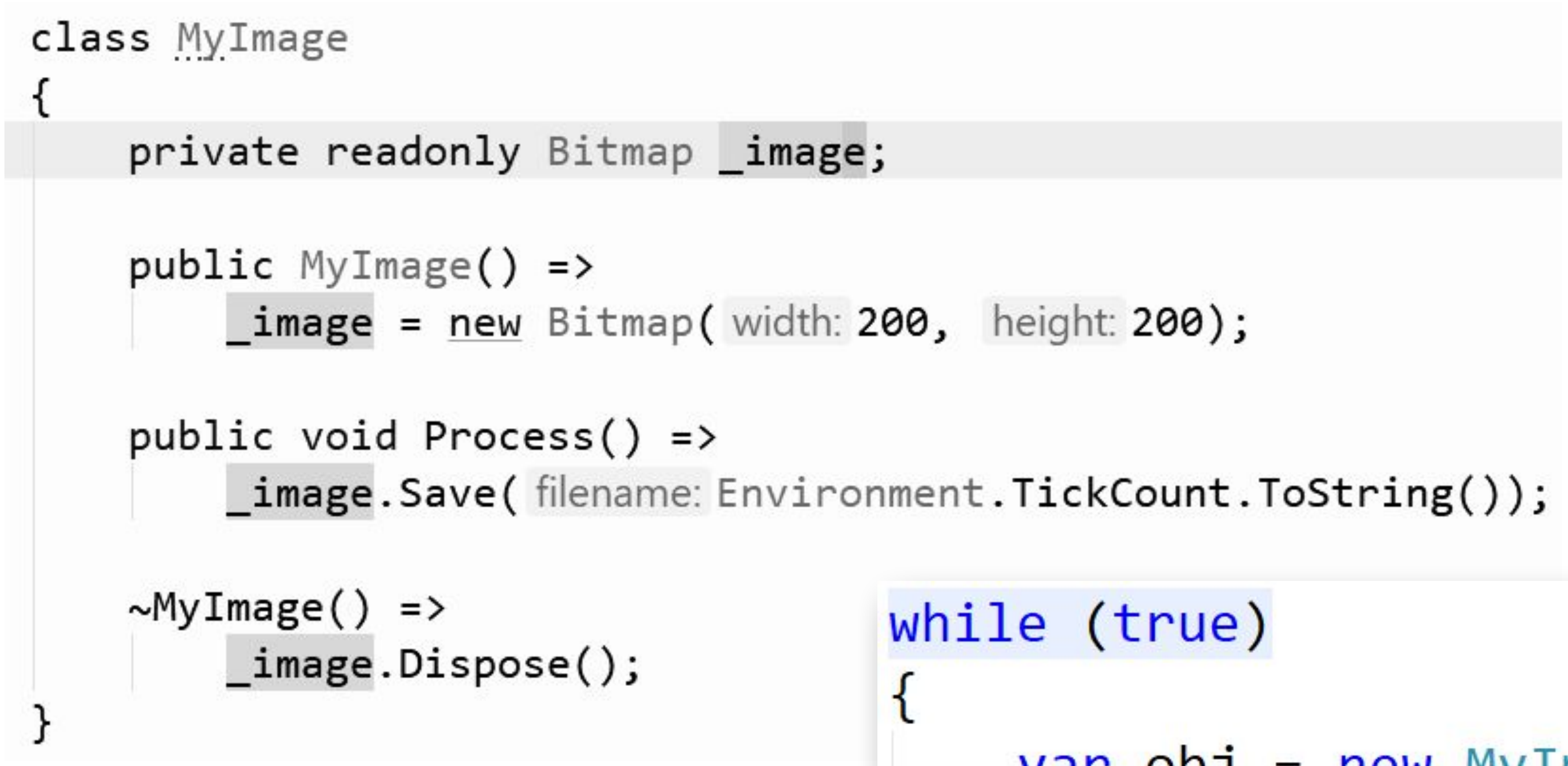
```
public static class Events
{
    public static event EventHandler OnCreate;
}

public class Concrete
{
    public Concrete() =>
        Events.OnCreate += EventsOnOnCreate;

    ~Concrete() =>
        Events.OnCreate -= EventsOnOnCreate;

    private void EventsOnOnCreate(object sender, EventArgs e)
    {
    }
}
```

Unhandled Finalized

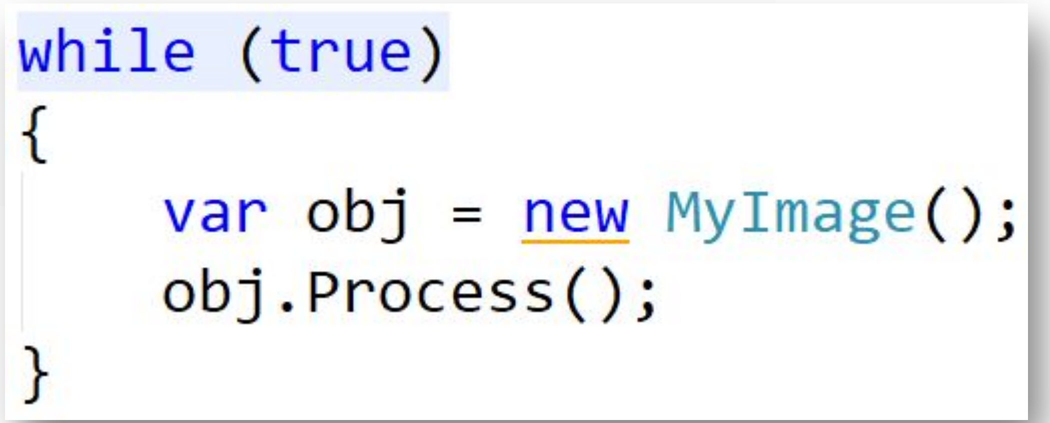


```
class MyImage
{
    private readonly Bitmap _image;

    public MyImage() =>
        _image = new Bitmap( width: 200,  height: 200);

    public void Process() =>
        _image.Save( filename: Environment.TickCount.ToString());

    ~MyImage() =>
        _image.Dispose();
}
```

Unhandled Finalized

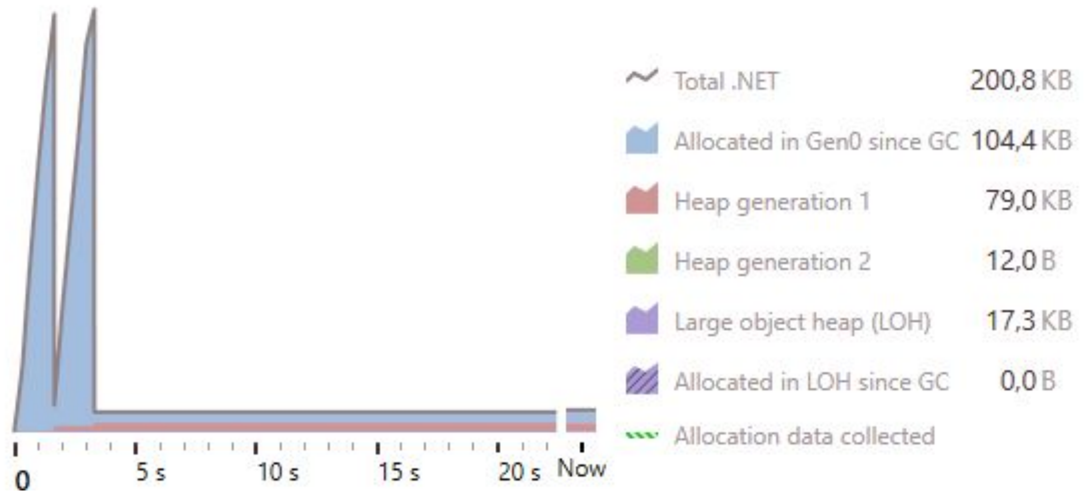


```
class MyImage
{
    private readonly Bitmap _image;

    public MyImage() =>
        _image = new Bitmap( width: 200,  height: 200);

    public void Process() =>
        _image.Save( filename: Environment.TickCount.ToString());

    ~MyImage() =>
        _image.Dispose();
}
```

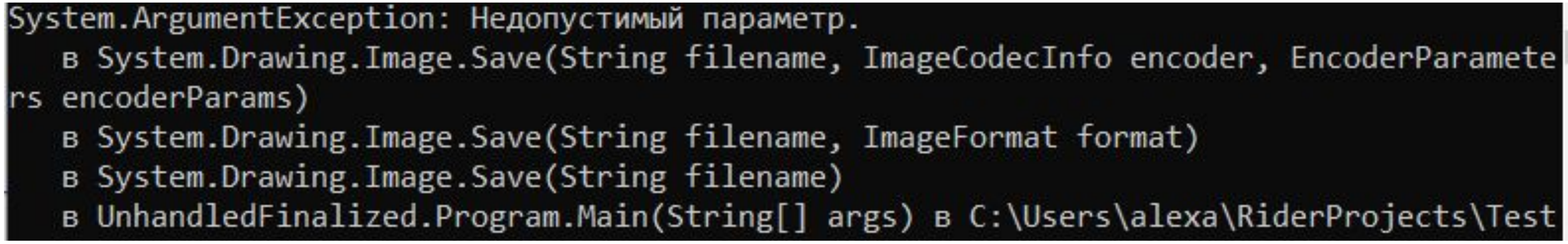


```
while (true)
{
    var obj = new MyImage();
    obj.Process();
}
```

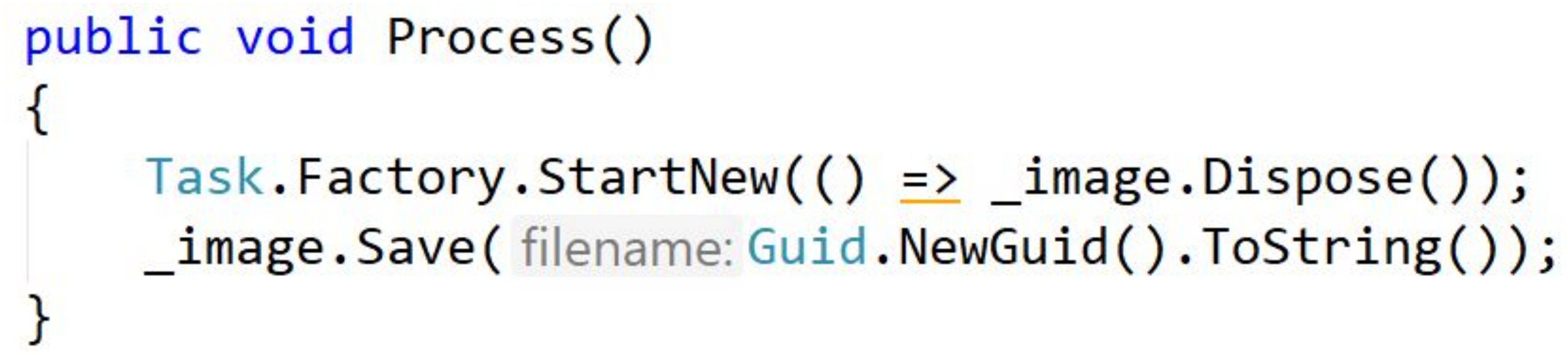
Unhandled Finalized



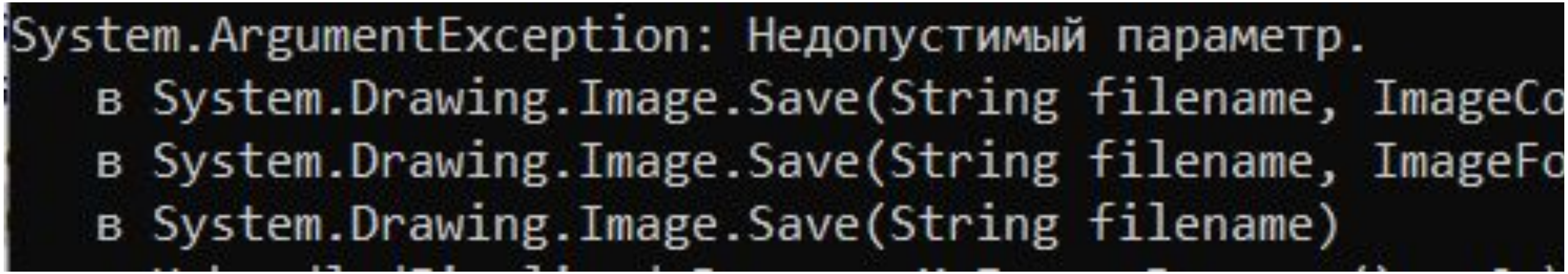
〰	Total .NET	200,8 KB
🟦	Allocated in Gen0 since GC	104,4 KB
🟥	Heap generation 1	79,0 KB
🟩	Heap generation 2	12,0 B
🟪	Large object heap (LOH)	17,3 KB
▨	Allocated in LOH since GC	0,0 B
〰	Allocation data collected	


```
System.ArgumentException: Недопустимый параметр.
   в System.Drawing.Image.Save(String filename, ImageCodecInfo encoder, EncoderParamete
rs encoderParams)
   в System.Drawing.Image.Save(String filename, ImageFormat format)
   в System.Drawing.Image.Save(String filename)
   в UnhandledFinalized.Program.Main(String[] args) в C:\Users\alexa\RiderProjects\Test
```

Unhandled Finalized

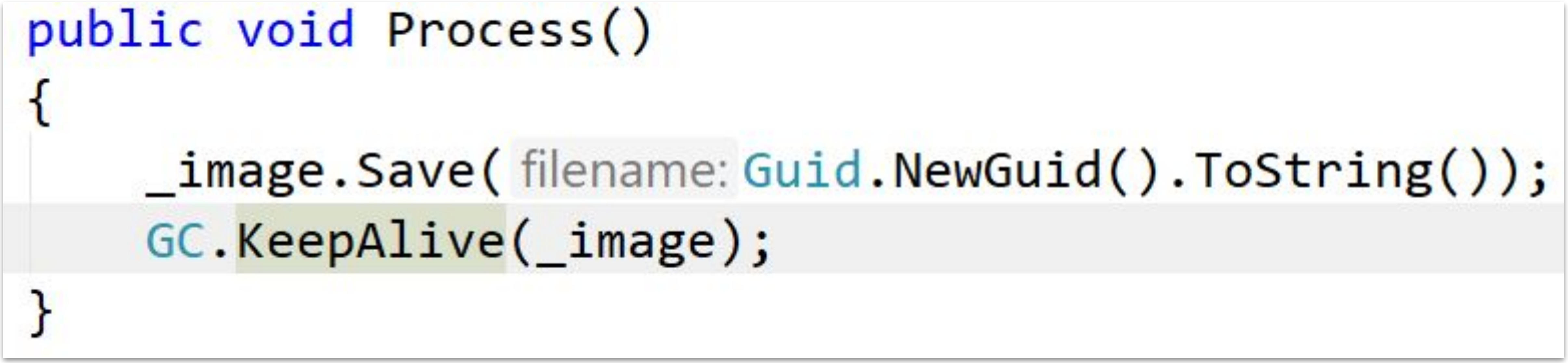


```
public void Process()
{
    Task.Factory.StartNew(() => _image.Dispose());
    _image.Save( filename: Guid.NewGuid().ToString());
}
```



```
System.ArgumentException: Недопустимый параметр.
   в System.Drawing.Image.Save(String filename, ImageCo
   в System.Drawing.Image.Save(String filename, ImageFo
   в System.Drawing.Image.Save(String filename)
```

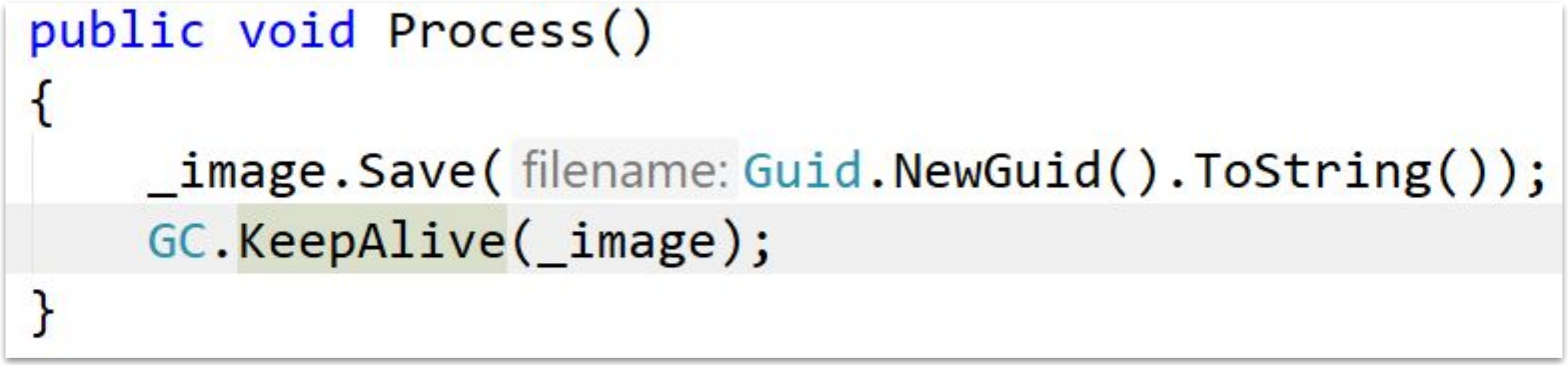
Unhandled Finalized



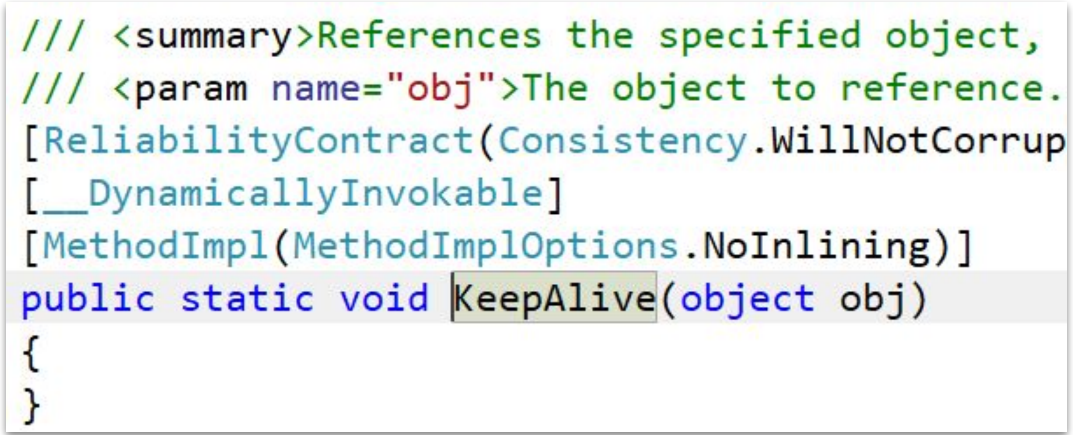
```
public void Process()
{
    _image.Save( filename: Guid.NewGuid().ToString());
    GC.KeepAlive(_image);
}
```

https://referencesource.microsoft.com/#mscorlib/system/gc.cs,279

Unhandled Finalized



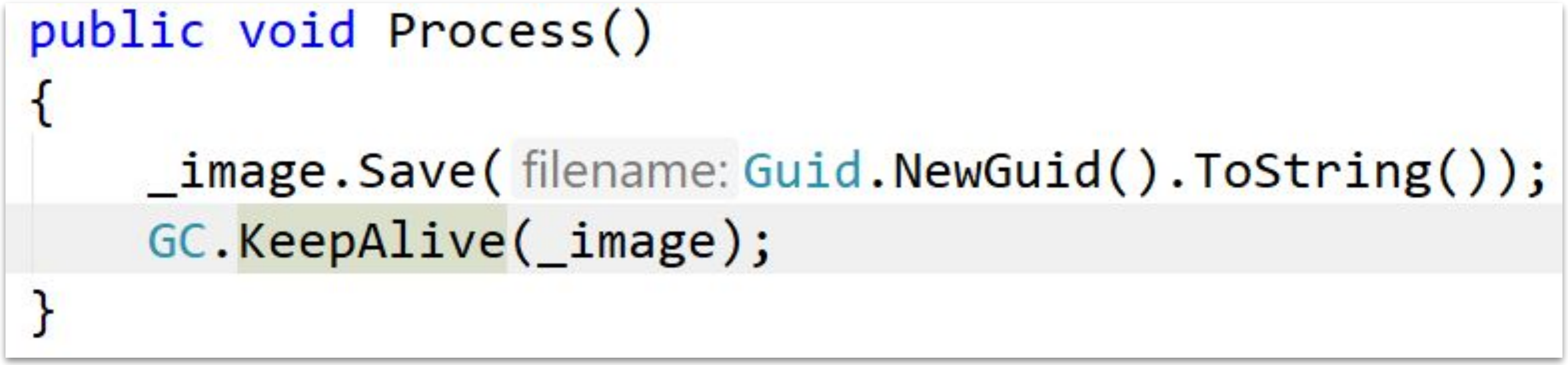
```
public void Process()
{
    _image.Save( filename: Guid.NewGuid().ToString());
    GC.KeepAlive(_image);
}
```



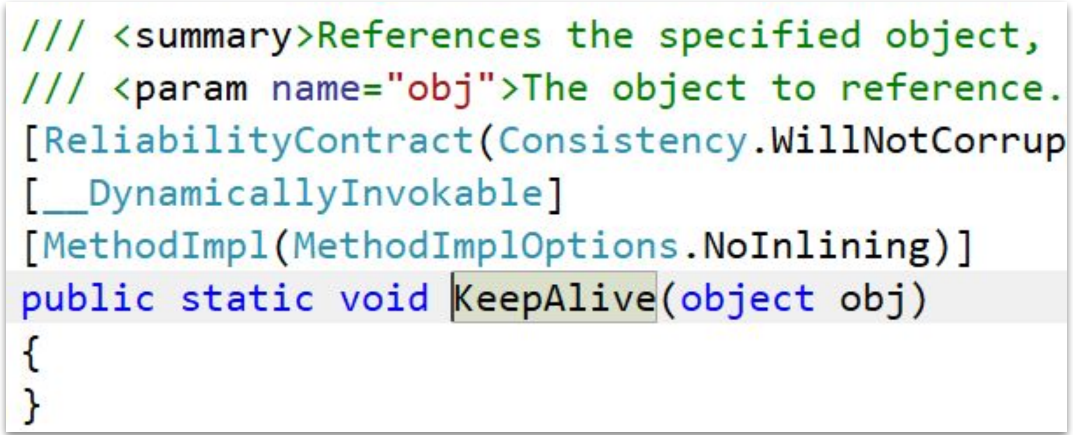
```
/// <summary>References the specified object,
/// <param name="obj">The object to reference.
[ReliabilityContract(Consistency.WillNotCorrup
[__DynamicallyInvokable]
[MethodImpl(MethodImplOptions.NoInlining)]
public static void KeepAlive(object obj)
{
}
```

https://referencesource.microsoft.com/#mscorlib/system/gc.cs,279

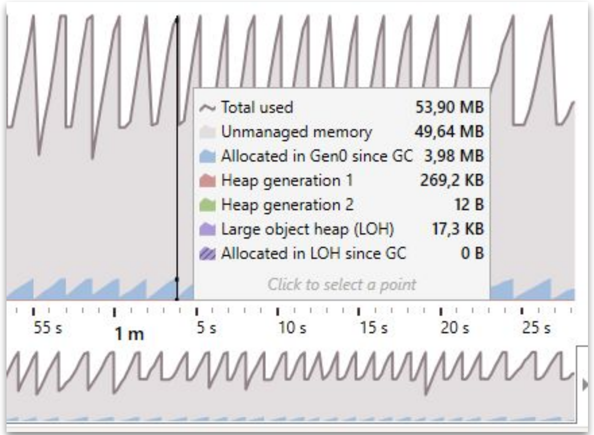
Unhandled Finalized



```
public void Process()
{
    _image.Save( filename: Guid.NewGuid().ToString());
    GC.KeepAlive(_image);
}
```



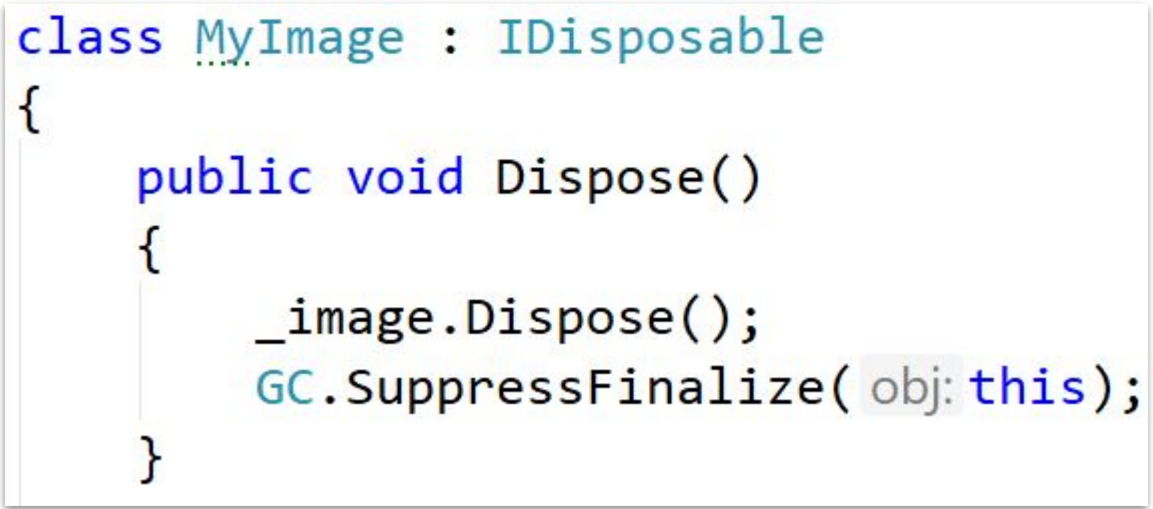
```
/// <summary>References the specified object,
/// <param name="obj">The object to reference.
[ReliabilityContract(Consistency.WillNotCorrup
[__DynamicallyInvokable]
[MethodImpl(MethodImplOptions.NoInlining)]
public static void KeepAlive(object obj)
{
}
```



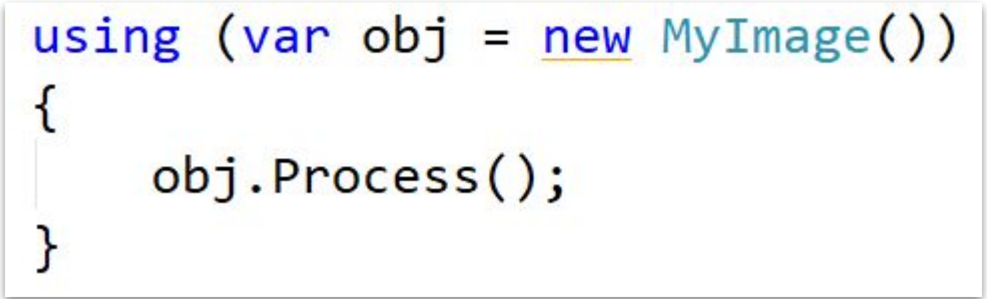
```
Total used                  53,90 MB
Unmanaged memory            49,64 MB
Allocated in Gen0 since GC   3,98 MB
Heap generation 1           269,2 KB
Heap generation 2               12 B
Large object heap (LOH)      17,3 KB
Allocated in LOH since GC        0 B
Click to select a point
```

https://referencesource.microsoft.com/#mscorlib/system/gc.cs,279

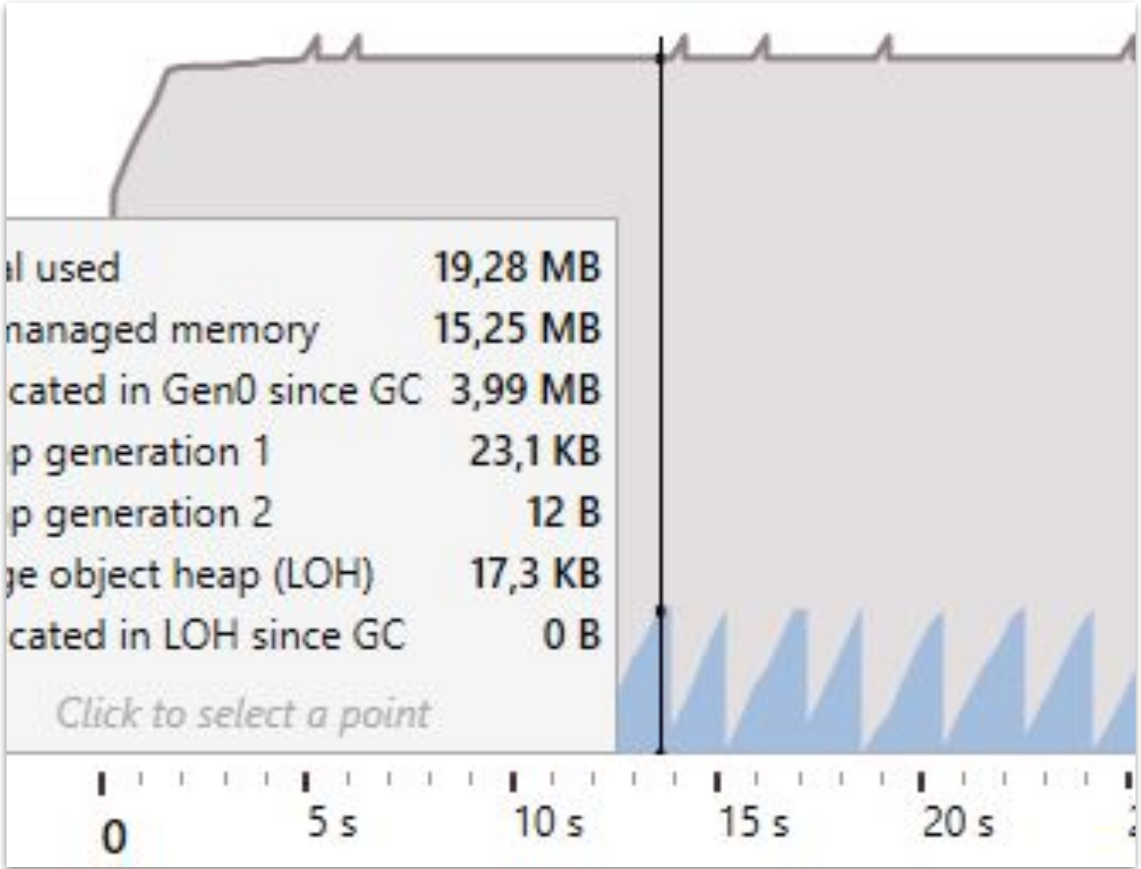
Unhandled Finalized



```
class MyImage : IDisposable
{
    public void Dispose()
    {
        _image.Dispose();
        GC.SuppressFinalize( obj: this);
    }
```



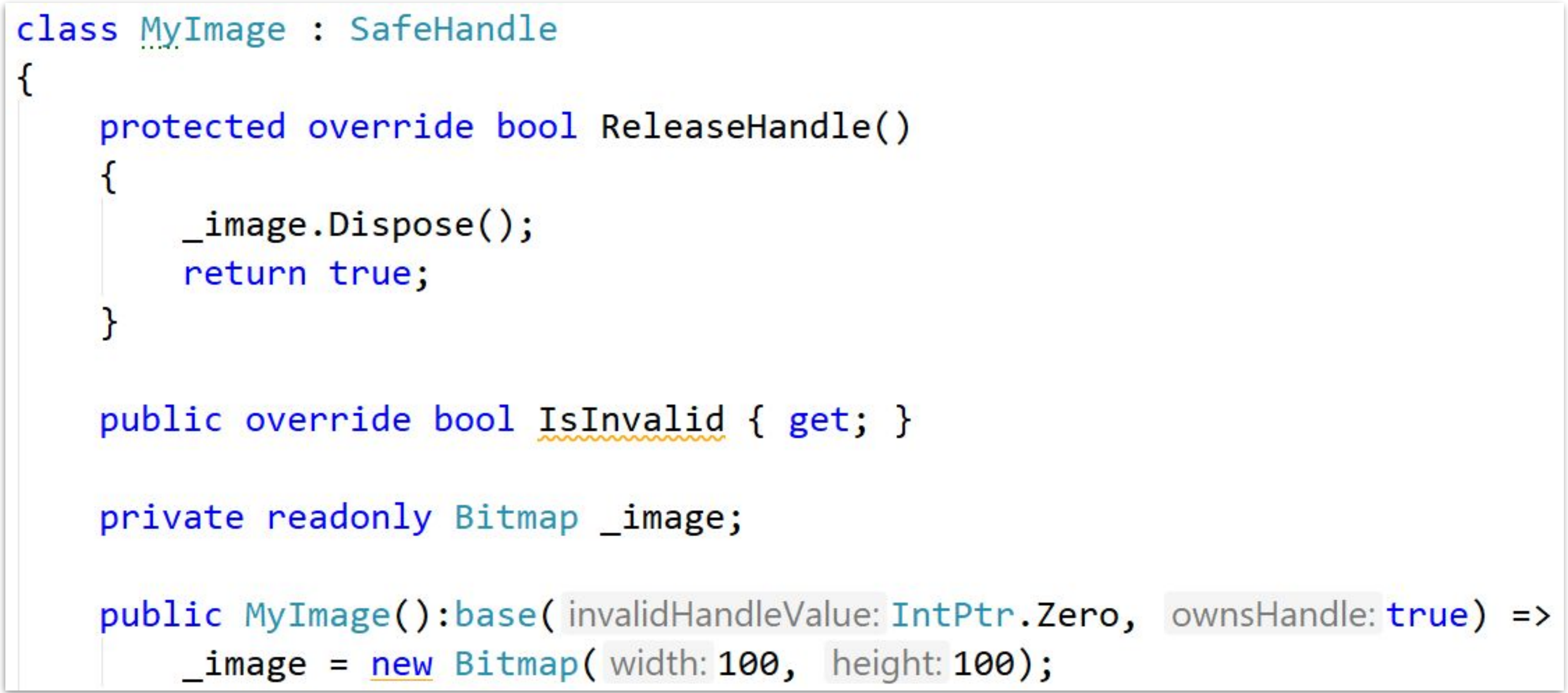
```
using (var obj = new MyImage())
{
    obj.Process();
}
```



```
l used                      19,28 MB
nanaged memory              15,25 MB
cated in Gen0 since GC       3,99 MB
p generation 1               23,1 KB
p generation 2                  12 B
ge object heap (LOH)         17,3 KB
cated in LOH since GC            0 B

 Click to select a point
```

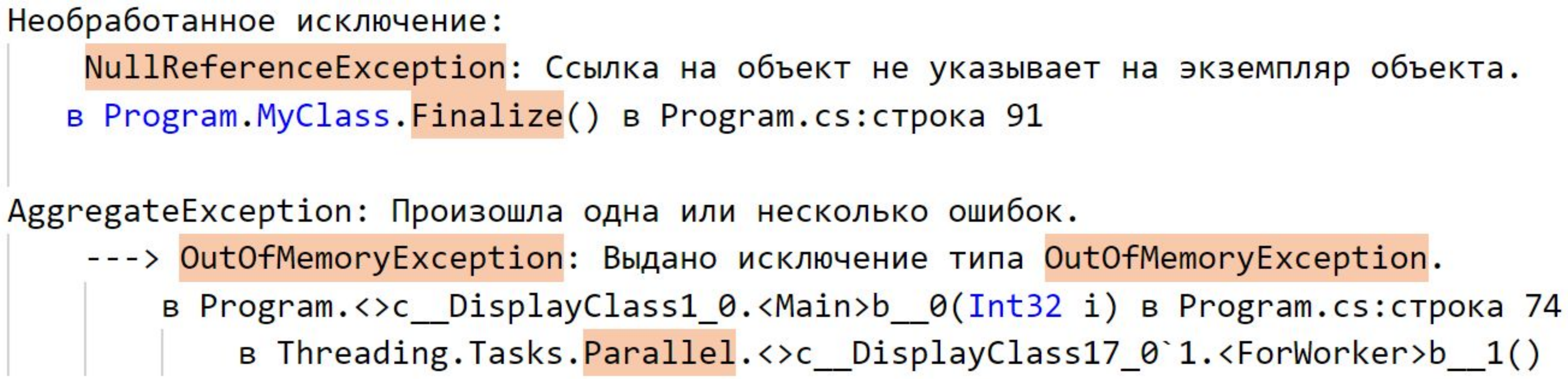
CriticalFinalizerObject

- SafeHandle
- CriticalHandle



```
class MyImage : SafeHandle
{
    protected override bool ReleaseHandle()
    {
        _image.Dispose();
        return true;
    }

    public override bool IsInvalid { get; }

    private readonly Bitmap _image;

    public MyImage():base( invalidHandleValue: IntPtr.Zero,  ownsHandle: true) =>
        _image = new Bitmap( width: 100,  height: 100);
```

Exception in Finalyzer


```
Необработанное исключение:
     NullReferenceException: Ссылка на объект не указывает на экземпляр объекта.
    в Program.MyClass.Finalize() в Program.cs:строка 91

AggregateException: Произошла одна или несколько ошибок.
    ---> OutOfMemoryException: Выдано исключение типа OutOfMemoryException.
        в Program.<>c__DisplayClass1_0.<Main>b__0(Int32 i) в Program.cs:строка 74
            в Threading.Tasks.Parallel.<>c__DisplayClass17_0`1.<ForWorker>b__1()
```

LOH

1. Объекты 85000
   a. x32 - new byte[84987]
   b. x64 - new byte[84976]

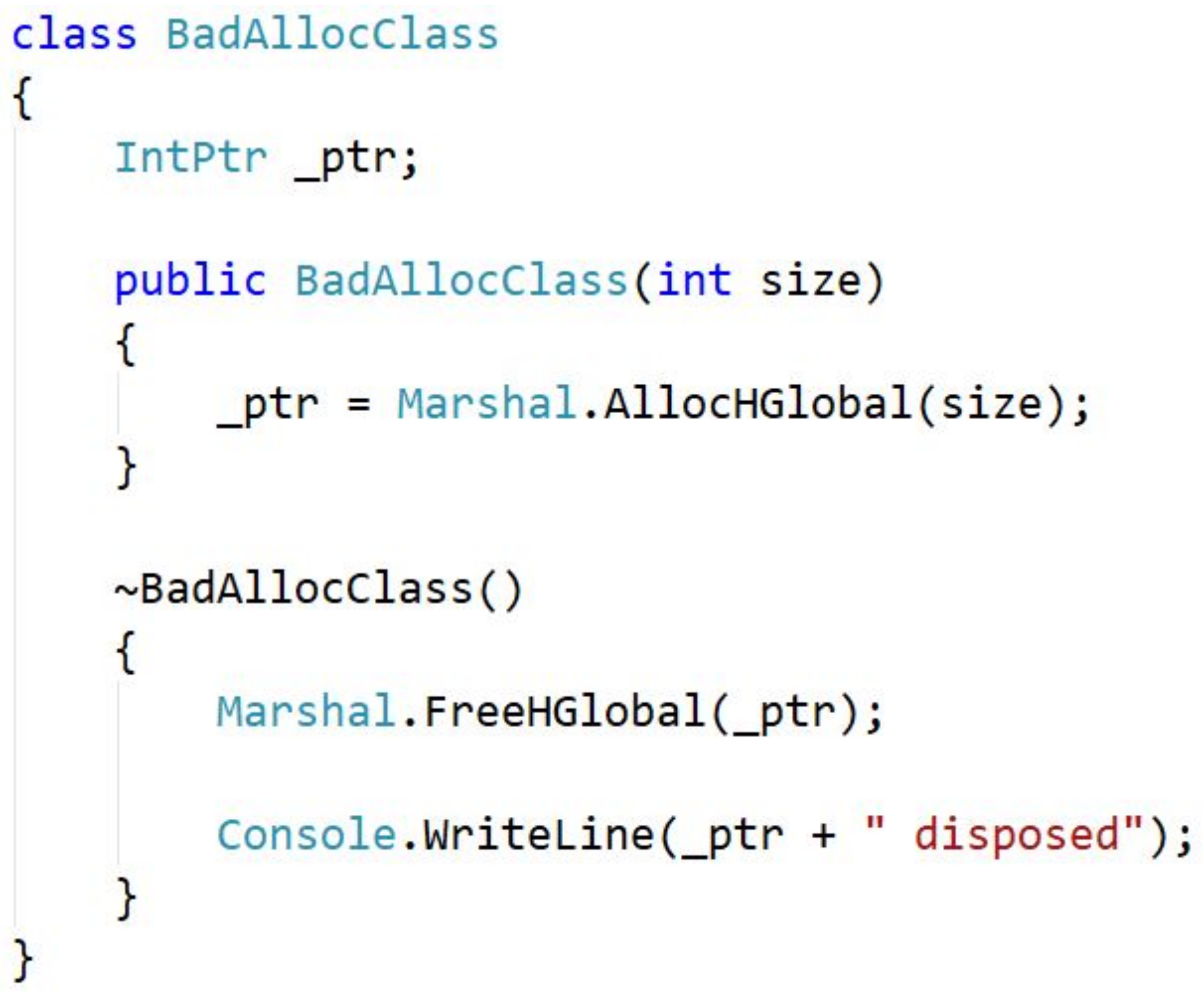
2. Отдельное адресное пространство

3. Не применяется сжатие
   a. см. GCLargeObjectHeapCompactionMode.CompactOnce

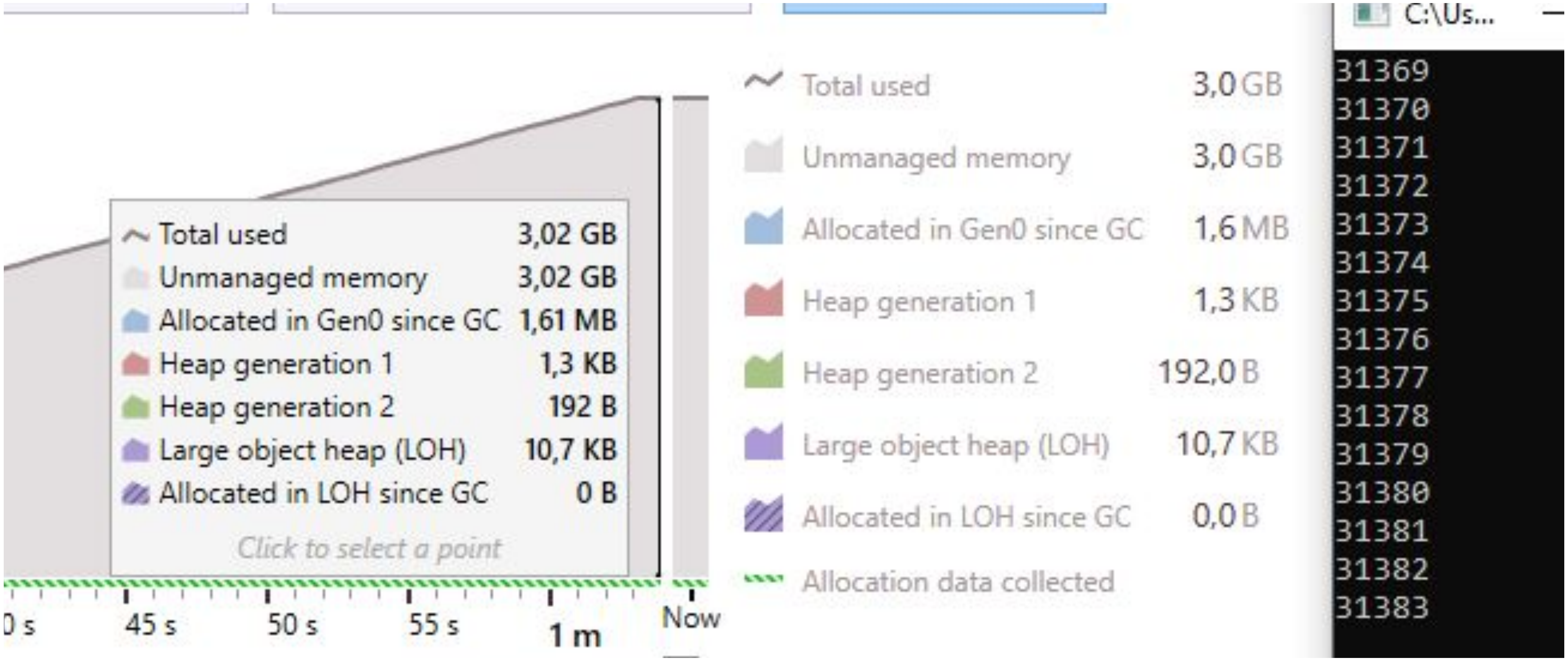
4. Фрагментация может привести к OutOfMemory

5. Относится к поколению 2

Нужно малость Unmanaged памяти



```
class BadAllocClass
{
    IntPtr _ptr;

    public BadAllocClass(int size)
    {
        _ptr = Marshal.AllocHGlobal(size);
    }

    ~BadAllocClass()
    {
        Marshal.FreeHGlobal(_ptr);

        Console.WriteLine(_ptr + " disposed");
    }
}
```

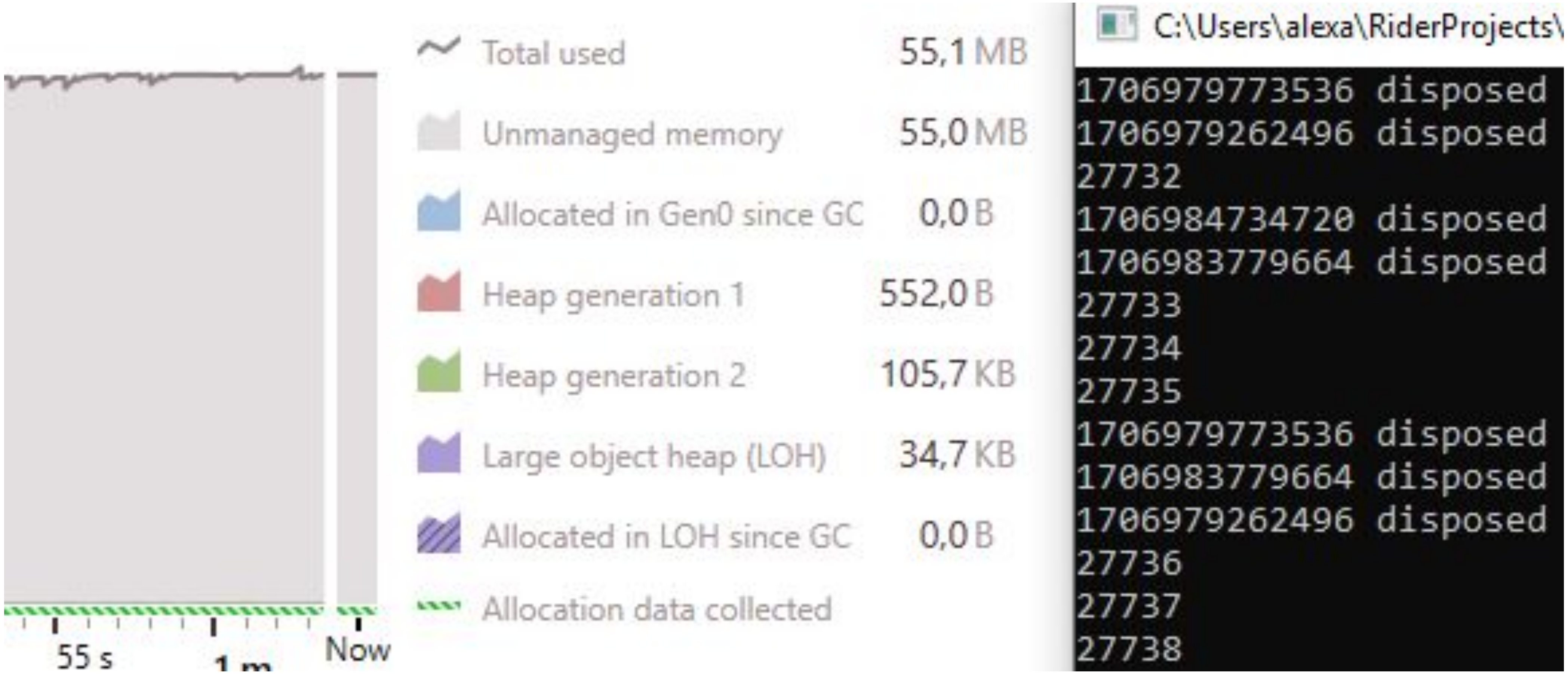


~	Total used	3,02 GB
	Unmanaged memory	3,02 GB
	Allocated in Gen0 since GC	1,61 MB
	Heap generation 1	1,3 KB
	Heap generation 2	192 B
	Large object heap (LOH)	10,7 KB
▨	Allocated in LOH since GC	0 B

~	Total used	3,0 GB
	Unmanaged memory	3,0 GB
	Allocated in Gen0 since GC	1,6 MB
	Heap generation 1	1,3 KB
	Heap generation 2	192,0 B
	Large object heap (LOH)	10,7 KB
▨	Allocated in LOH since GC	0,0 B
	Allocation data collected	

```
31369
31370
31371
31372
31373
31374
31375
31376
31377
31378
31379
31380
31381
31382
31383
```

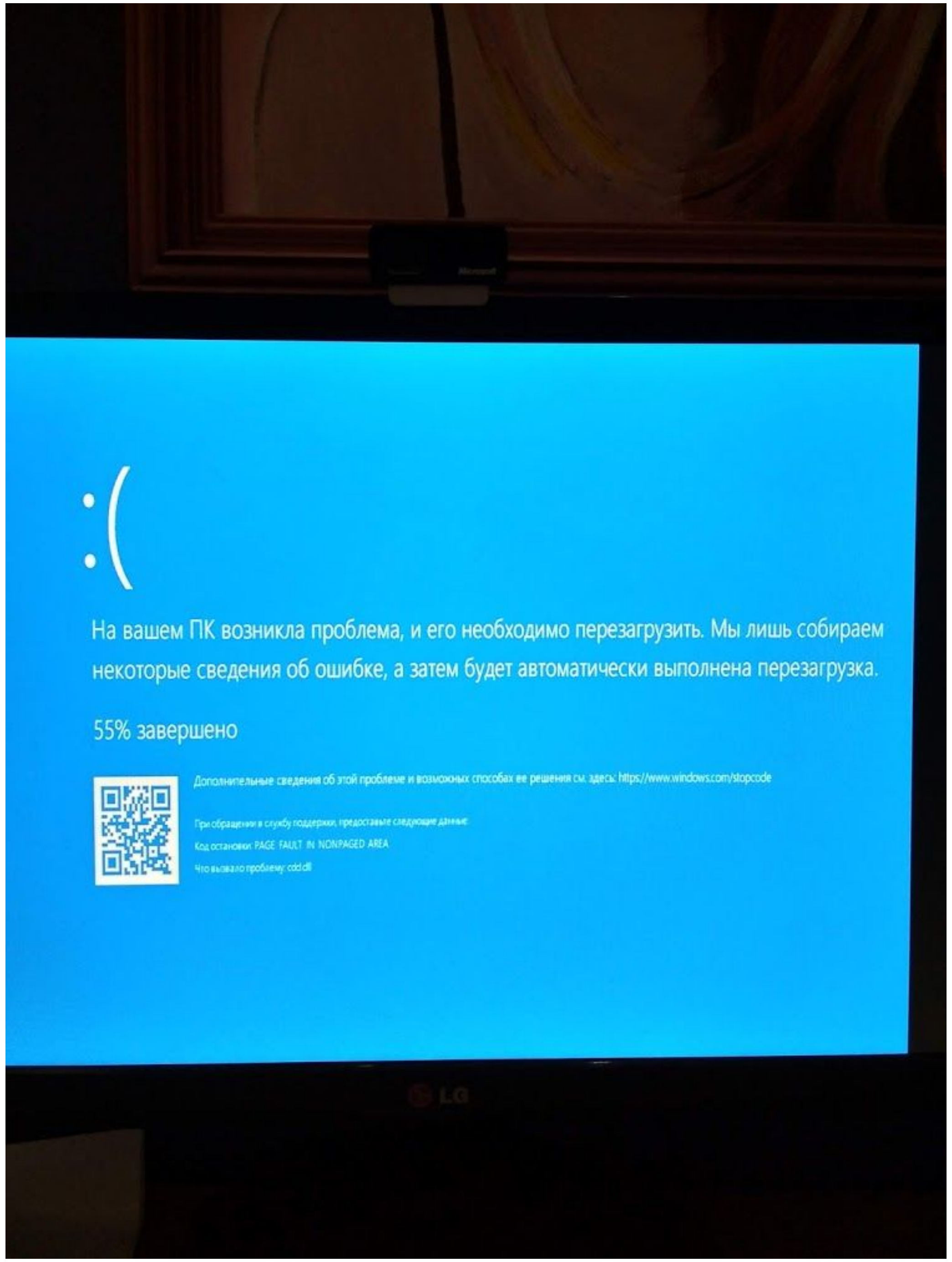
```
class GoodAllocClass
{
    IntPtr _ptr;
    int _size;

    public GoodAllocClass(int size)
    {
        _ptr = Marshal.AllocHGlobal(size);

        GC.AddMemoryPressure(size);

        _size = size;
    }

    ~GoodAllocClass()
    {
        Marshal.FreeHGlobal(_ptr);

        GC.RemoveMemoryPressure(_size);

        Console.WriteLine(_ptr + " disposed");
    }
}
```



∼	Total used	55,1 MB
▨	Unmanaged memory	55,0 MB
▨	Allocated in Gen0 since GC	0,0 B
▨	Heap generation 1	552,0 B
▨	Heap generation 2	105,7 KB
▨	Large object heap (LOH)	34,7 KB
▨	Allocated in LOH since GC	0,0 B
▨	Allocation data collected	

C:\Users\alexa\RiderProjects\

```
1706979773536 disposed
1706979262496 disposed
27732
1706984734720 disposed
1706983779664 disposed
27733
27734
27735
1706979773536 disposed
1706983779664 disposed
1706979262496 disposed
27736
27737
27738
```



:(

На вашем ПК возникла проблема, и его необходимо перезагрузить. Мы лишь собираем некоторые сведения об ошибке, а затем будет автоматически выполнена перезагрузка.

55% завершено

Дополнительные сведения об этой проблеме и возможных способах ее решения см. здесь: https://www.windows.com/stopcode

При обращении в службу поддержки, предоставьте следующие данные:

Код остановки: PAGE FAULT IN NONPAGED AREA

Что вызвало проблему: cdd.dll

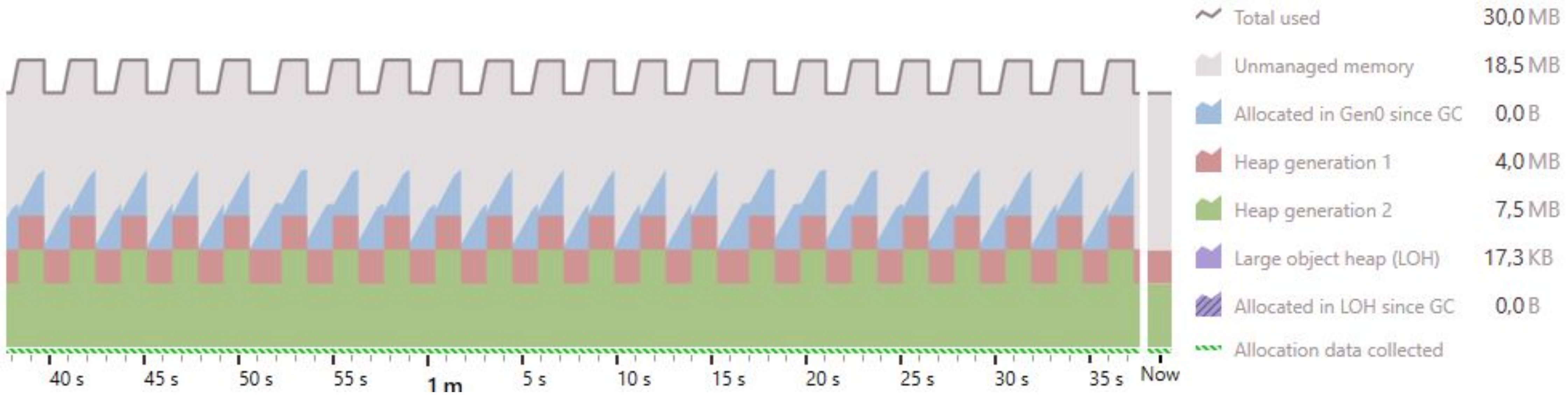
GC.TryStartNoGCRegion

Attempts to disallow garbage collection during the execution of a critical path.

https://docs.microsoft.com/en-us/dotnet/api/system.gc.trystartnogcregion?view=netframework-4.8

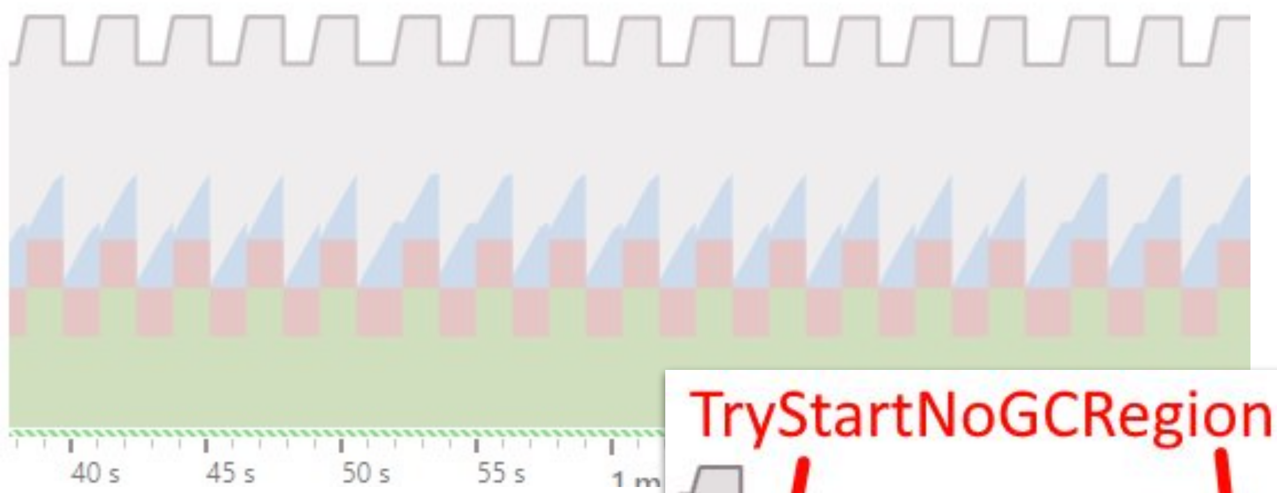
GC.TryStartNoGCRegion

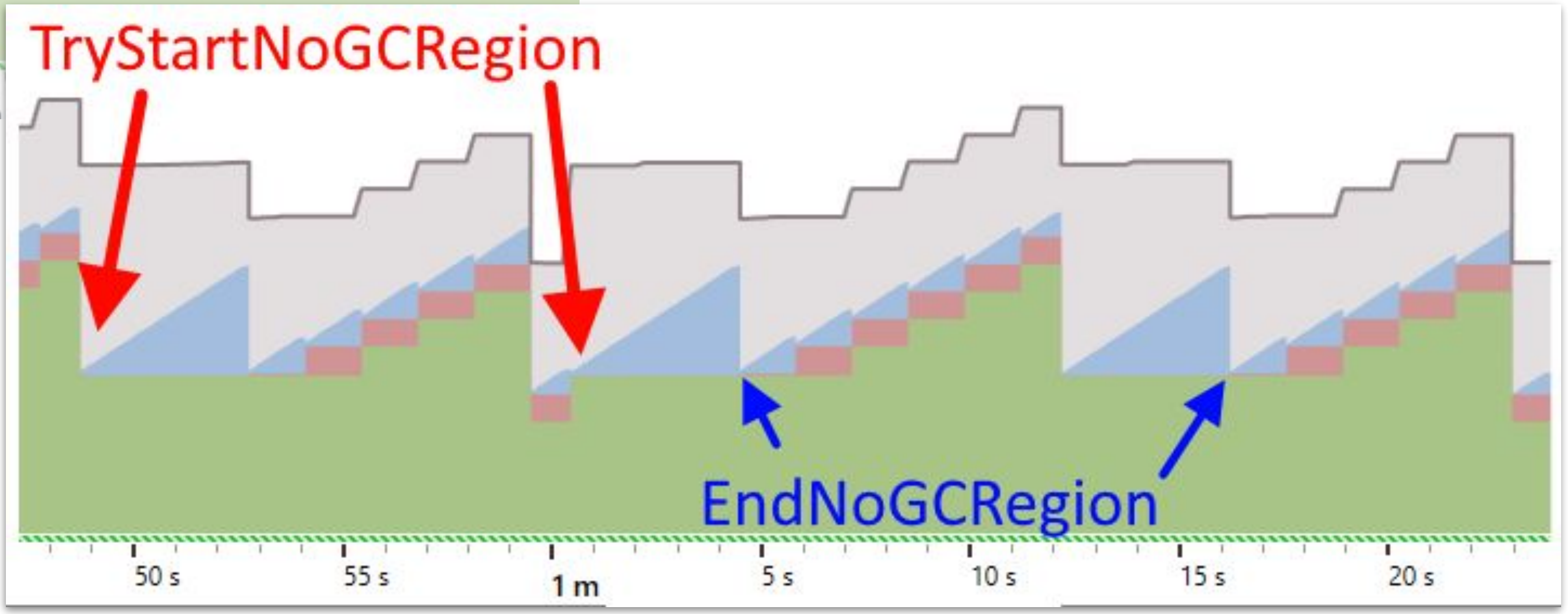
Attempts to disallow garbage collection during the execution of a critical path.



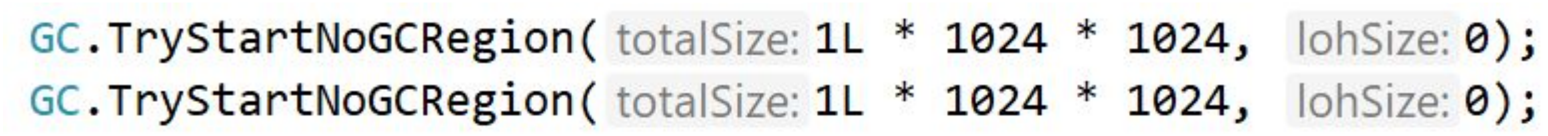
https://docs.microsoft.com/en-us/dotnet/api/system.gc.trystartnogcregion?view=netframework-4.8

GC.TryStartNoGCRegion





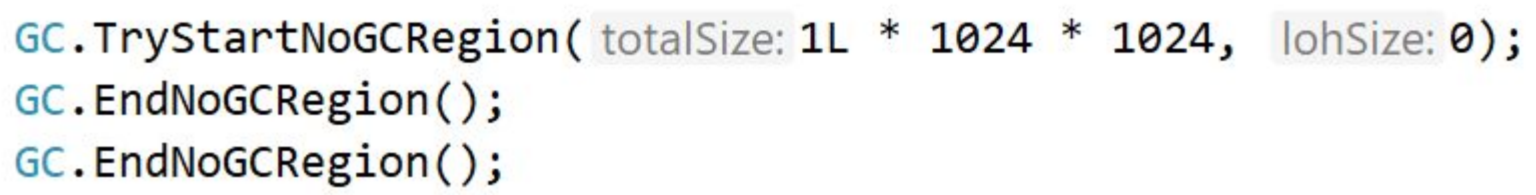
GC.TryStartNoGCRegion



```
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
```

```
InvalidOperationException: The NoGCRegion mode was already in progress
```

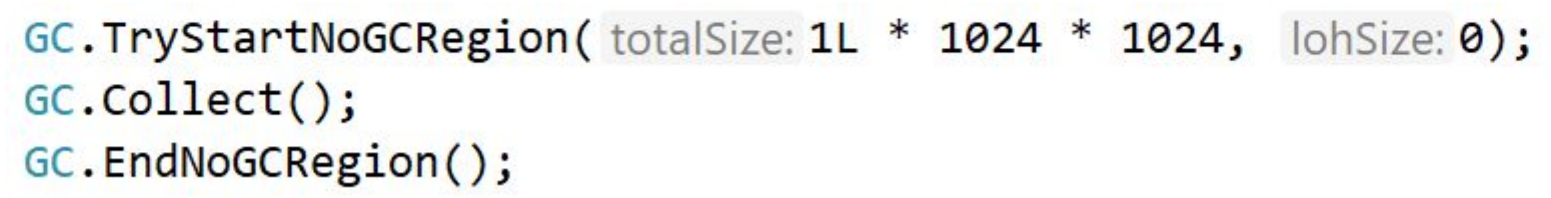
GC.TryStartNoGCRegion



```
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
GC.EndNoGCRegion();
GC.EndNoGCRegion();
```

```
InvalidOperationException: NoGCRegion mode must be set
```

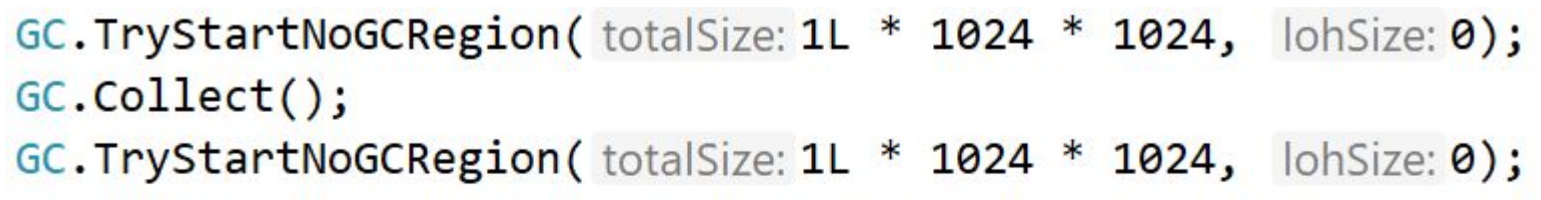
GC.TryStartNoGCRegion



```
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
GC.Collect();
GC.EndNoGCRegion();
```

```
InvalidOperationException: Garbage collection was induced in NoGCRegion mode
```

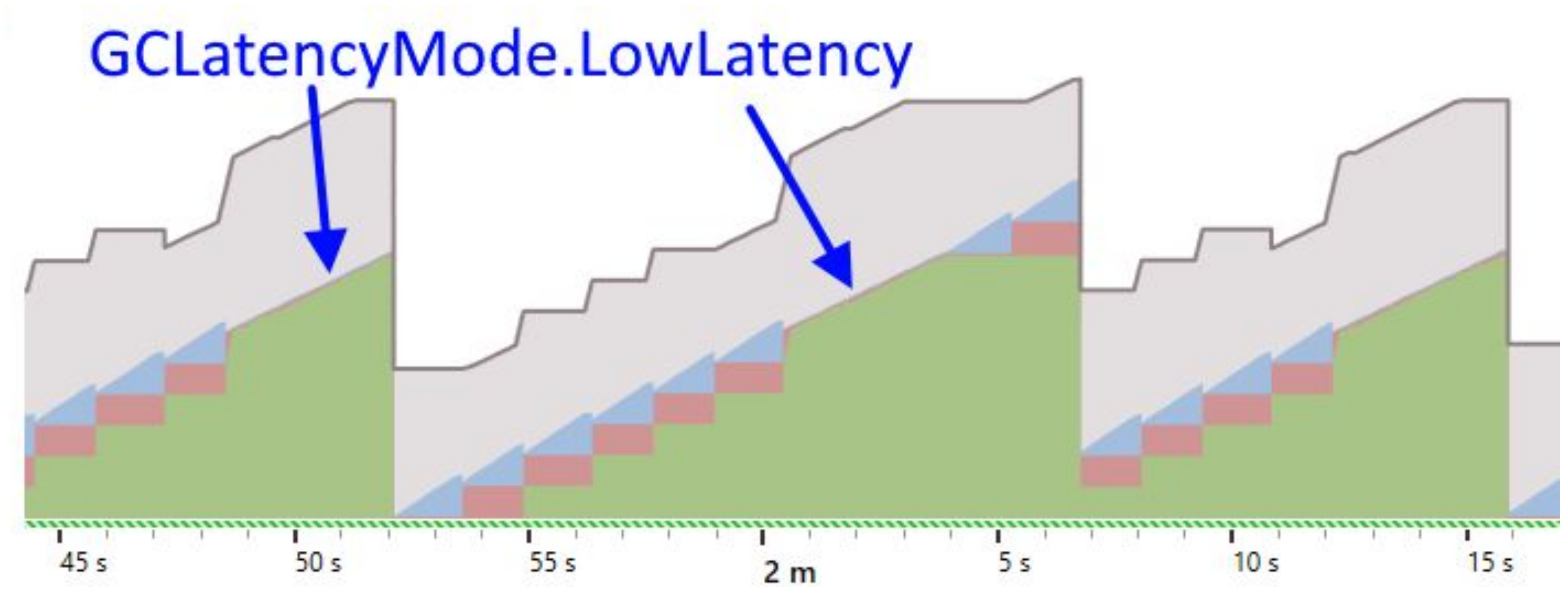
GC.TryStartNoGCRegion



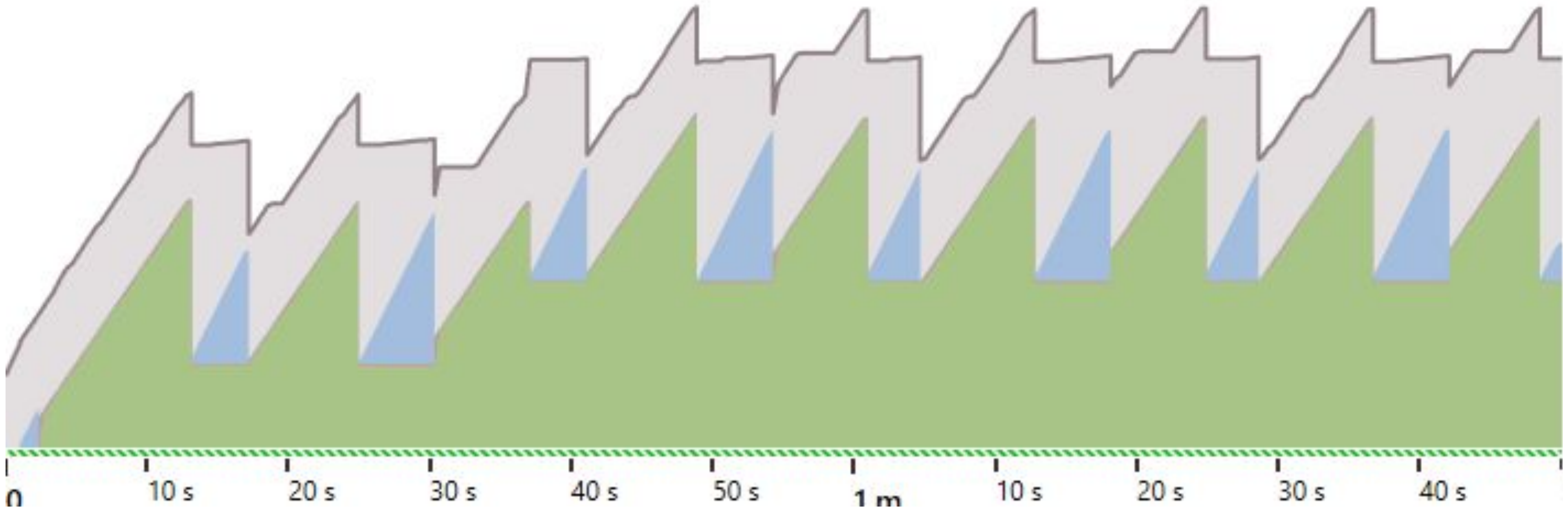
```
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
GC.Collect();
GC.TryStartNoGCRegion( totalSize: 1L * 1024 * 1024,  lohSize: 0);
```

```
InvalidOperationException: The NoGCRegion mode was already in progress
```

Сборщик мусора LatencyMode



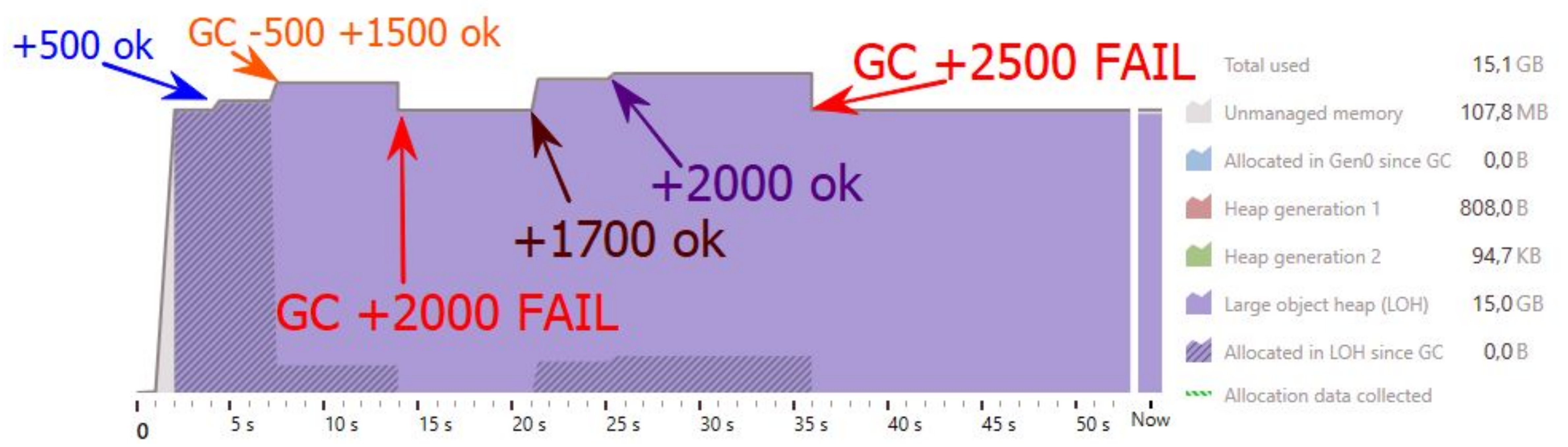
LowLatency + TryStartNoGCRegion



GC Server: SustainedLowLatency

MemoryFailPoint

Checks for sufficient memory resources before executing an operation

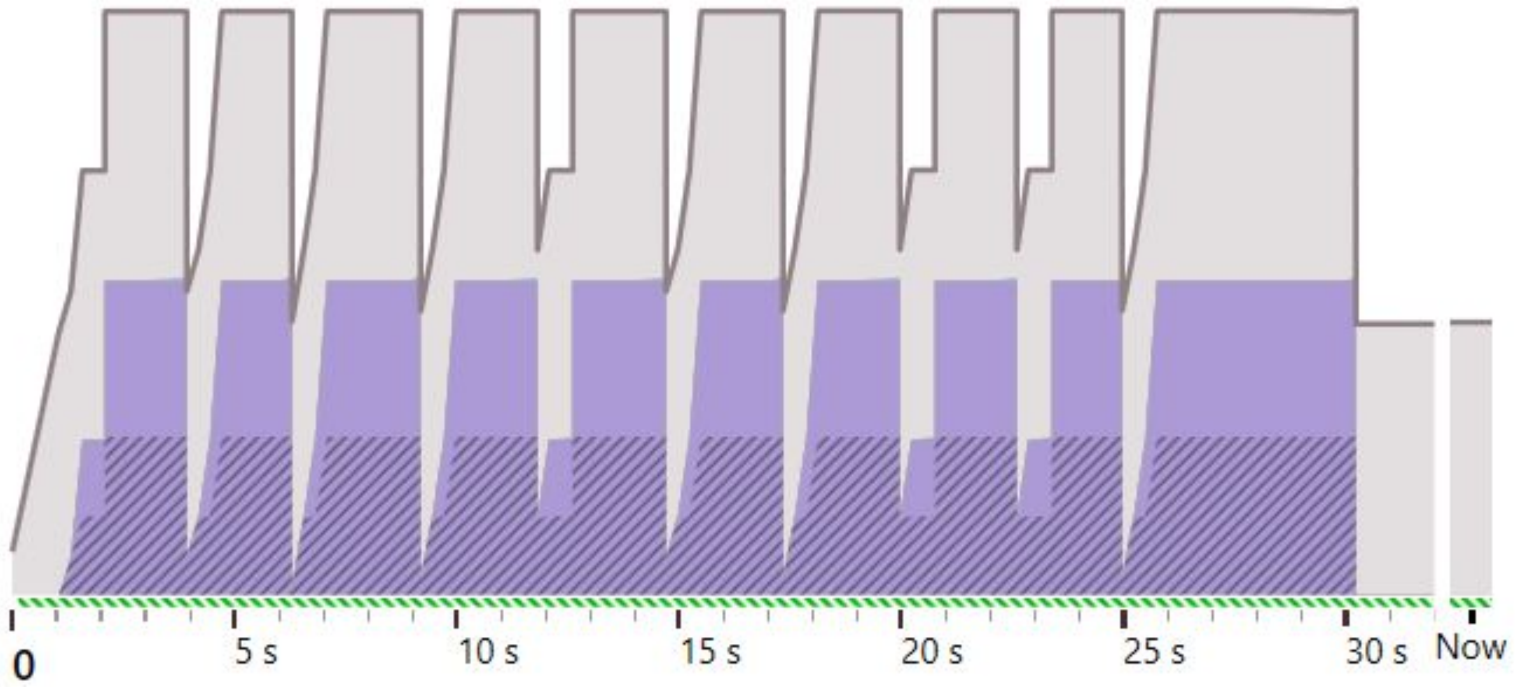


Total used	15,1 GB
Unmanaged memory	107,8 MB
Allocated in Gen0 since GC	0,0 B
Heap generation 1	808,0 B
Heap generation 2	94,7 KB
Large object heap (LOH)	15,0 GB
Allocated in LOH since GC	0,0 B
Allocation data collected	

https://referencesource.microsoft.com/#mscorlib/system/runtime/memoryfailpoint.cs,161

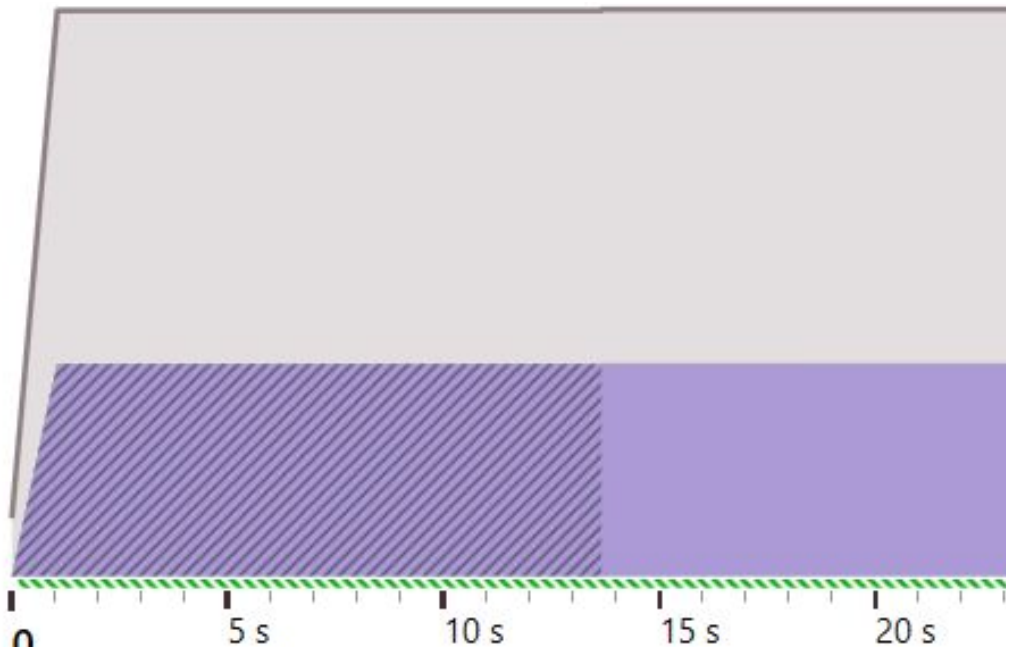
Arrays

```
var list = new List<double>();

for (int i = 0; i < ...; i++)
{
   list.Add(i);
```



ArrayPool (nuget System.Buffers)

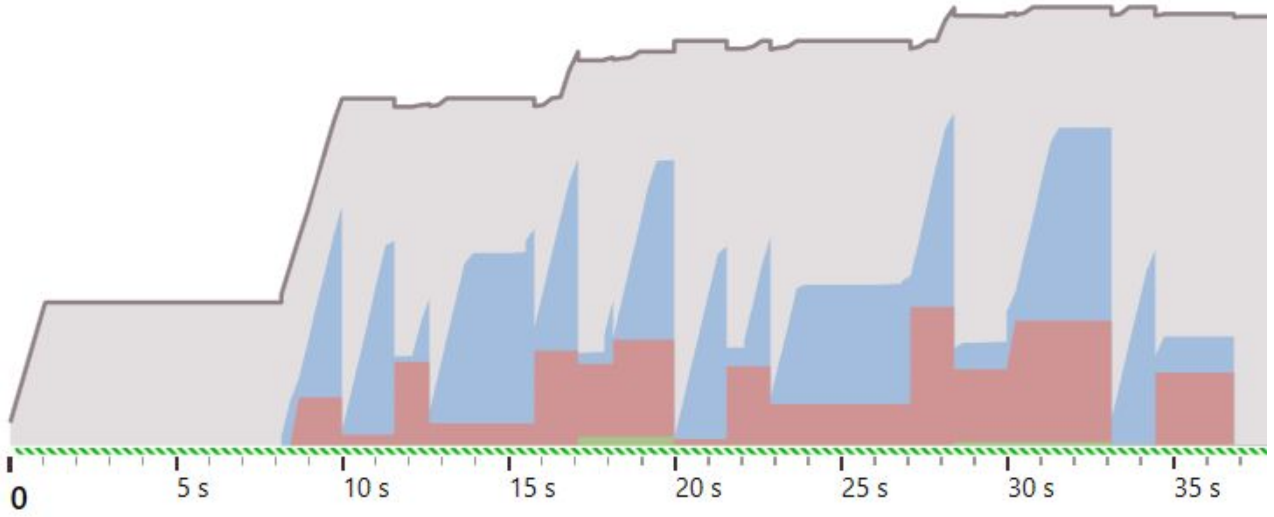
```
var pool = ArrayPool<double>.Shared;

IList<double> list = pool.Rent(size);

for (int i = 0; i < ...; i++)
{
    list[i] = i;
}

pool.Return((double[])list, true);
```



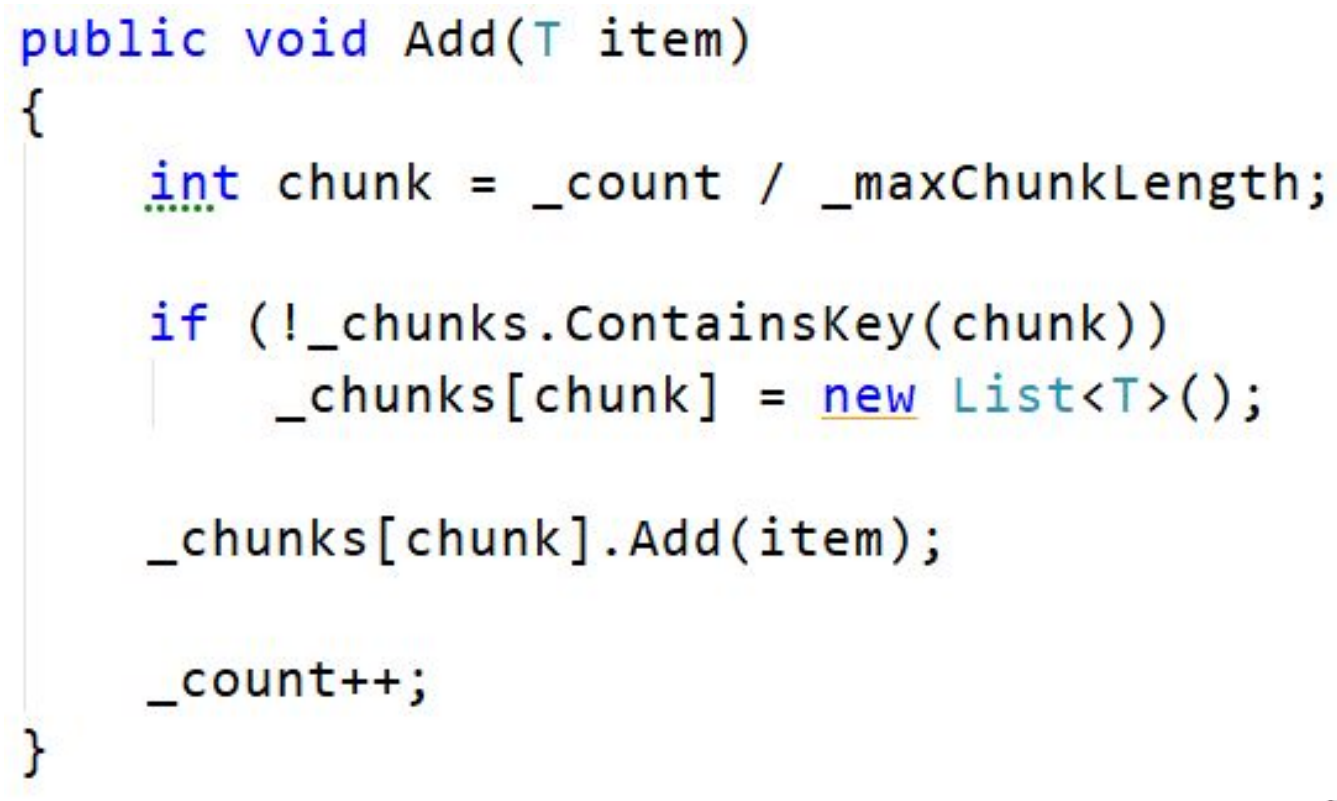
https://adamsitnik.com/Array-Pool/

Chuncked Arrays

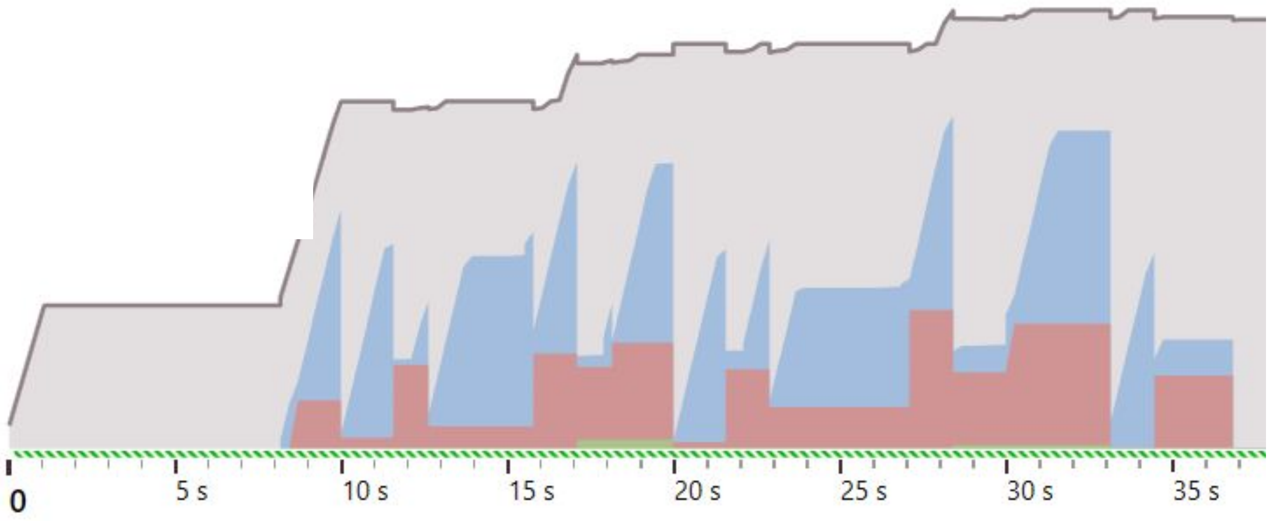
```
var list = new MyChunckedList<double>(1000);

for (int i = 0; i < ...; i++)
{
    list.Add(i);
}
```



Chuncked Arrays



```
public void Add(T item)
{
    int chunk = _count / _maxChunkLength;

    if (!_chunks.ContainsKey(chunk))
        _chunks[chunk] = new List<T>();

    _chunks[chunk].Add(item);

    _count++;
}
```



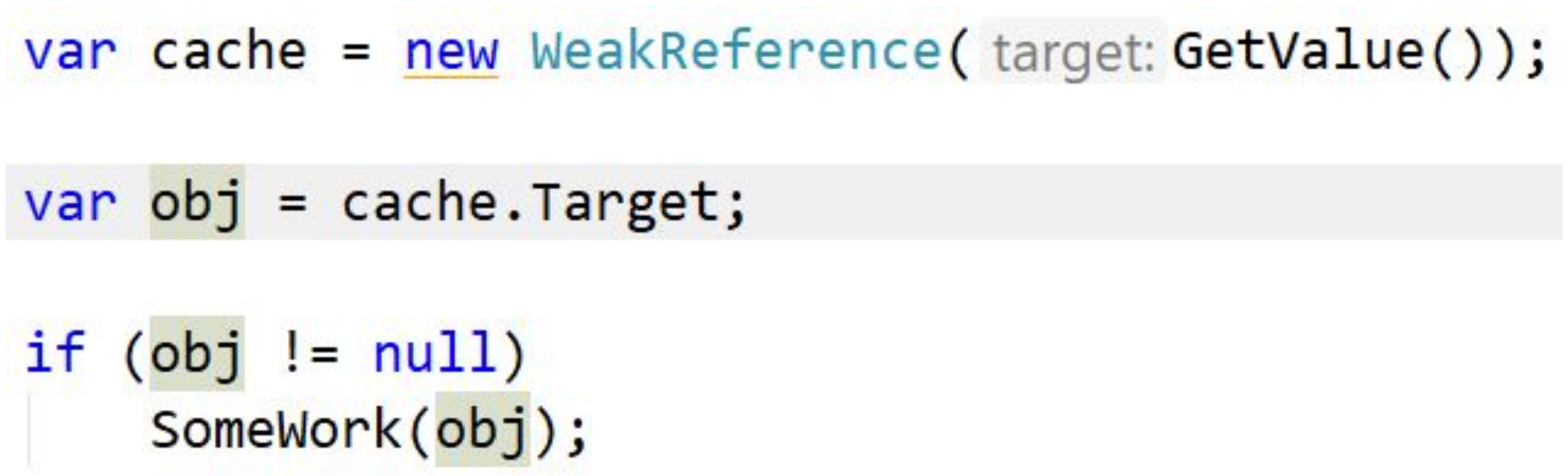
Слабые ссылки

Позволяют создать ссылку на объект 😄

Держать эту ссылку

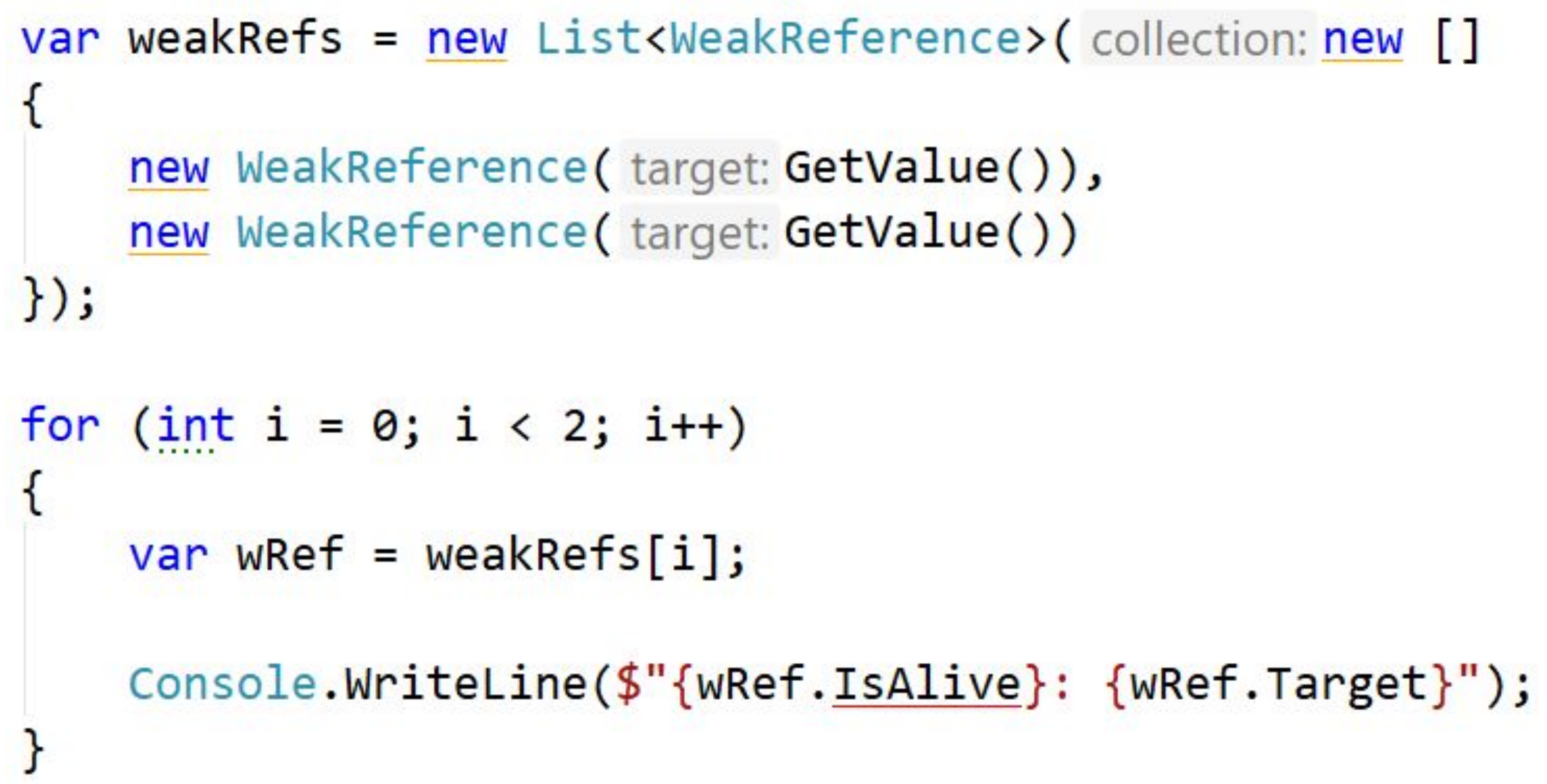
Но GC может удалить объект

Слабые ссылки



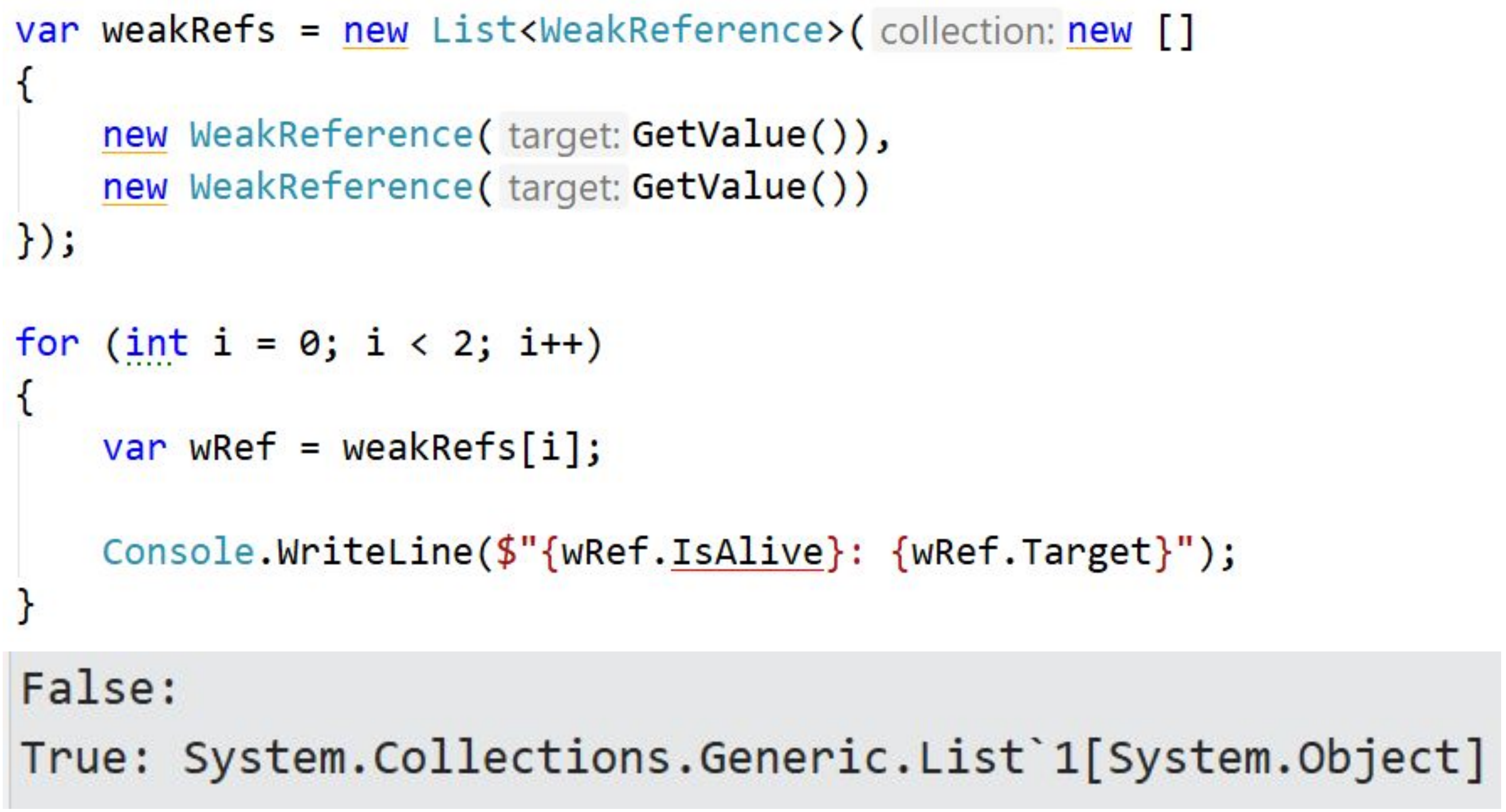
```
var cache = new WeakReference( target: GetValue());

var obj = cache.Target;

if (obj != null)
    SomeWork(obj);
```

Слабые ссылки



```
var weakRefs = new List<WeakReference>( collection: new []
{
    new WeakReference( target: GetValue()),
    new WeakReference( target: GetValue())
});

for (int i = 0; i < 2; i++)
{
    var wRef = weakRefs[i];

    Console.WriteLine($"{wRef.IsAlive}: {wRef.Target}");
}
```

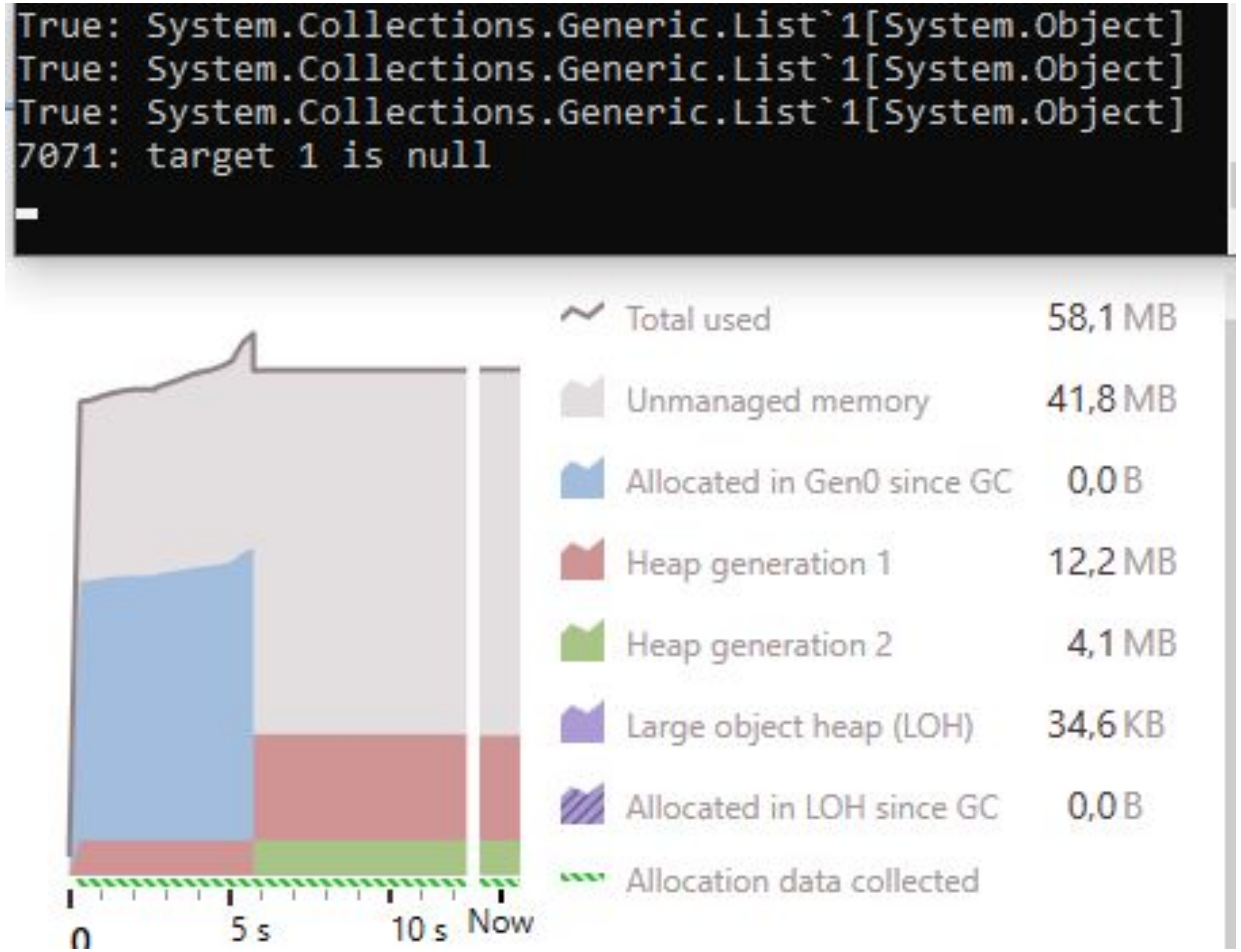
Слабые ссылки



```
var weakRefs = new List<WeakReference>( collection: new []
{
    new WeakReference( target: GetValue()),
    new WeakReference( target: GetValue())
});

for (int i = 0; i < 2; i++)
{
    var wRef = weakRefs[i];

    Console.WriteLine($"{wRef.IsAlive}: {wRef.Target}");
}
```

```
False:
True: System.Collections.Generic.List`1[System.Object]
```

Слабые ссылки



```
True: System.Collections.Generic.List`1[System.Object]
True: System.Collections.Generic.List`1[System.Object]
True: System.Collections.Generic.List`1[System.Object]
7071: target 1 is null
```

〰	Total used	58,1 MB
⬜	Unmanaged memory	41,8 MB
🟦	Allocated in Gen0 since GC	0,0 B
🟥	Heap generation 1	12,2 MB
🟩	Heap generation 2	4,1 MB
🟪	Large object heap (LOH)	34,6 KB
▨	Allocated in LOH since GC	0,0 B
▨	Allocation data collected	

Ручная сборка

```
GC.Collect()
    =>
        GC.Collect(GC.MaxGeneration,
                   GCCollectionMode.Optimized)
```

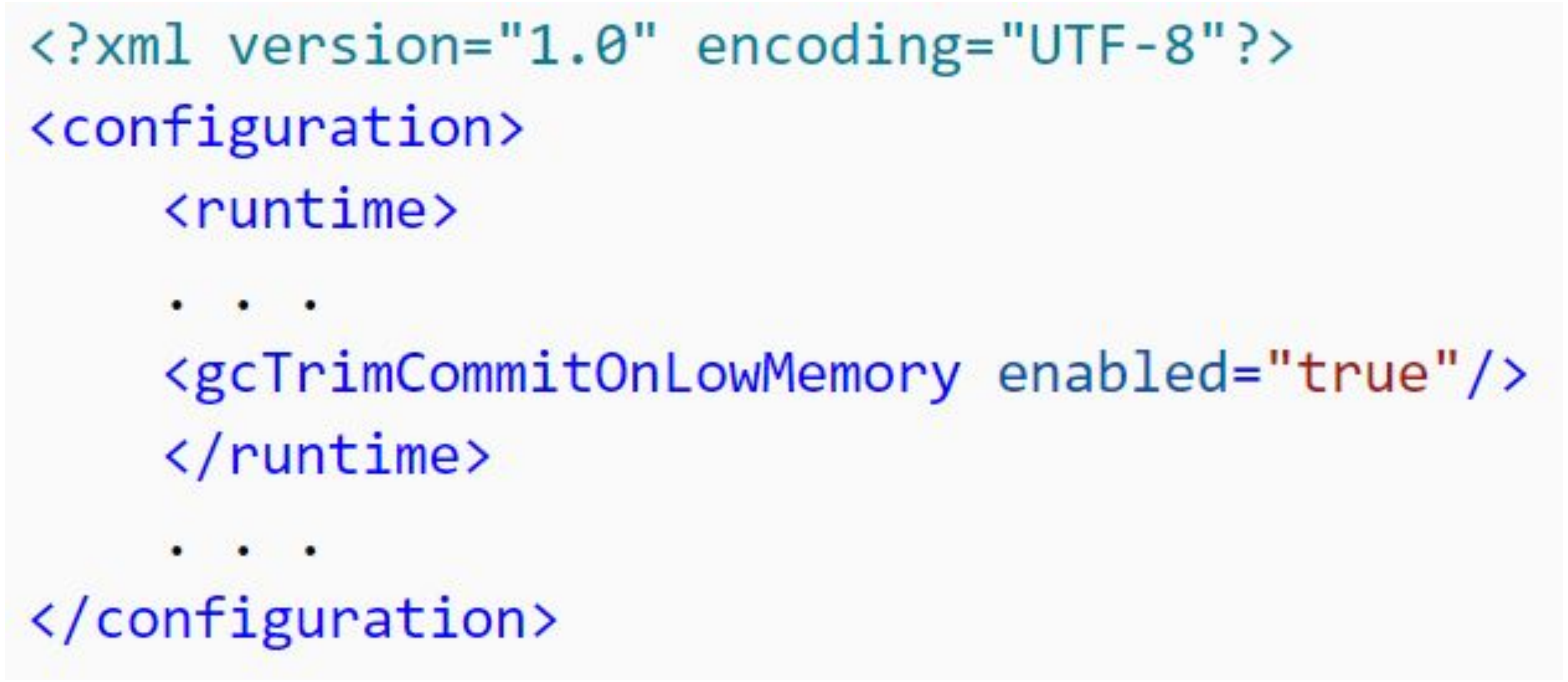
Ручная сборка

```
GC.Collect(GC.MaxGeneration,
           GCCollectionMode.Forced,
           blocking: true,
           compacting: true);
```

Ручная сборка

```
GCSettings.LargeObjectHeapCompactionMode =
    GCLargeObjectHeapCompactionMode.CompactOnce;

GC.Collect(GC.MaxGeneration,
             GCCollectionMode.Forced,
             blocking: true,
             compacting: true);
```

gcTrimCommitOnLowMemory (Aspnet.config)



```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <runtime>
    . . .
    <gcTrimCommitOnLowMemory enabled="true"/>
    </runtime>
    . . .
</configuration>
```

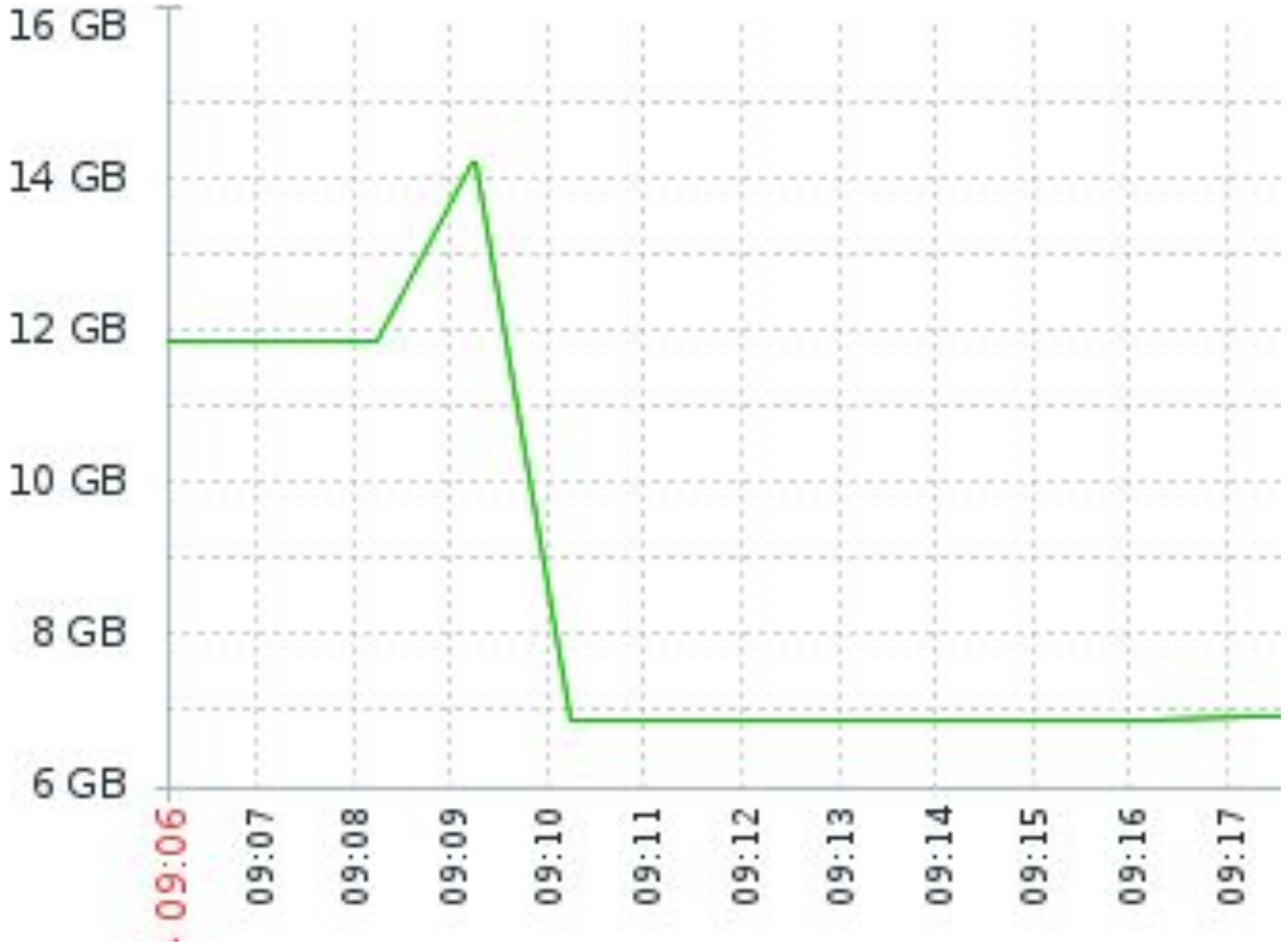
https://bit.ly/31kUbCc

Админы староверы

```
> testlimit64 -d -c 5000
```

Админы староверы

> testlimit64 -d -c 5000



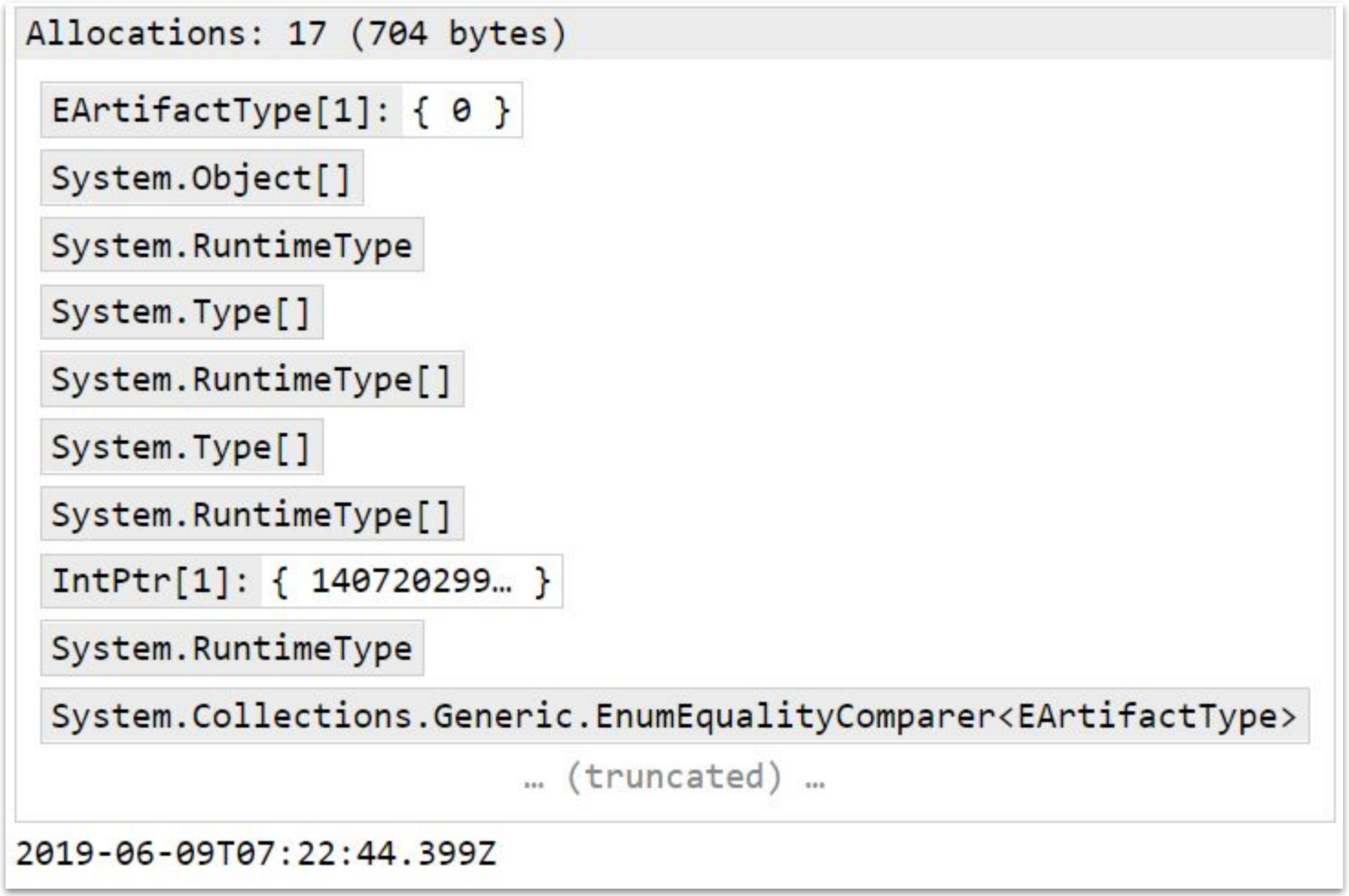
Помогайте GC

```
string.Format(
  $"{{0:yyyy-MM-ddTHH:mm:ss.fff}}{
    (artifactType.In(EArtifactType.None) ? "" : "Z")
  }", createDate
);
```

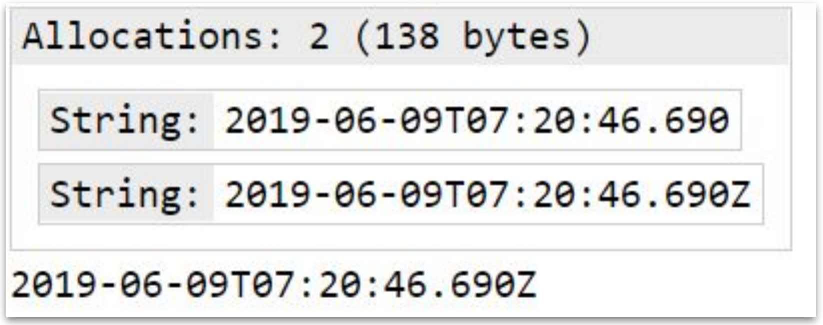
Помогайте GC

```
string.Format(
  $"{{0:yyyy-MM-ddTHH:mm:ss.fff}}{
    (artifactType.In(EArtifactType.None) ? "" : "Z")
  }", createDate
);
```

```
res = createDate.ToString("yyyy-MM-ddTHH:mm:ss.fff");

if (artifactType != EArtifactType.None)
   res += "Z";
```



```
Allocations: 17 (704 bytes)

EArtifactType[1]: { 0 }
System.Object[]
System.RuntimeType
System.Type[]
System.RuntimeType[]
System.Type[]
System.RuntimeType[]
IntPtr[1]: { 140720299… }
System.RuntimeType
System.Collections.Generic.EnumEqualityComparer<EArtifactType>
                 … (truncated) …

2019-06-09T07:22:44.399Z
```



```
Allocations: 2 (138 bytes)

String: 2019-06-09T07:20:46.690
String: 2019-06-09T07:20:46.690Z

2019-06-09T07:20:46.690Z
```

https://sharplab.io/#gist:18bf7a776cb79aa19400654e5df9becd
https://sharplab.io/#gist:299577c2fa542fa086d8aed399f971f9
https://sharplab.io/#gist:c946710d9728db254ae29e17def61c1d

Ссылки

https://bit.ly/2WdXNWS - официальная дока

https://sharplab.io - online inspect C#, F#, VB.NET

https://bit.ly/2HApjEK - GC - друг или враг (RavenDb)

https://bit.ly/2YFcm3w - PerfView

https://bit.ly/2QY3IKB - TestLimit

https://bit.ly/2K5MhHq - dotnetbook

https://github.com/SanSYS/trygc

Microsoft

МАСТЕР КЛАСС

CLR via C#

ПРОГРАММИРОВАНИЕ НА ПЛАТФОРМЕ MICROSOFT .NET FRAMEWORK 4.5 НА ЯЗЫКЕ C#

4-е издание Джеффри Рихтер ПИТЕР

@SanSYS