Microsoft

LESSON 7.1

10754 Microsoft .NET Fundamentals

Understand Resource Allocation

Lesson Overview

Understand resource allocation.

In this lesson, you will learn about:

- The function and benefits of the .NET garbage collector.
- Fundamental memory concepts.
- Stacks and heaps.

Guiding Questions

1. What system manages memory in a .NET application?
2. What prompts garbage collection?
3. What is the difference between stacks and heaps?

Anticipatory Set

Most students have some container that they use to store all the materials and resources needed for school. It might be a backpack, a binder, a locker, or all three.

- What makes a student decide to clean out his or her container?
- When students clean the container, how do they decide what to keep and what to recycle?

Discuss your ideas with a partner.

.NET Garbage Collection

- The Microsoft® .NET Framework's garbage collector manages the allocation and release of memory for your application.
- The garbage collector provides the following benefits:
  - It enables you to develop an application without having to free memory.
  - Allocates objects on the managed heap efficiently.
  - Reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations.
    - Managed objects automatically get clean content to start with, so their constructors do not have to initialize every data field.
  - Provides memory safety by making sure that an object cannot use the content of another object.
- .NET Framework provides the `GC` (garbage collector) class for manually controlling garbage collection.

.NET Memory Fundamentals

- Each process has its own space, referred to as **virtual memory**. All processes on the same computer share the same physical memory.
- By default, on 32-bit computers, each process has a 2-GB user-mode virtual address space.
- **Application developers work only with virtual address space** and never manipulate physical memory directly.
  - The garbage collector allocates and frees virtual memory for you on the managed heap.

Virtual Memory

- Virtual memory can be in three states:
  - **Free:** the block of memory has no references to it and is available for allocation.
  - **Reserved:** the block of memory is available for your use and cannot be used for any other allocation request. However, you cannot store data to this memory block until it is committed.
  - **Committed:** the block of memory is assigned to physical storage.
- Virtual address space can get fragmented. This means that there are free blocks, also known as *holes,* interspersed with reserved and committed blocks in the address space. Even if you have 2 GB of free space, an allocation request for 2 GB will be unsuccessful unless all that space is in a single address block.
- You can run out of memory if you run out of virtual address space to reserve or physical space to commit.

Initiating Garbage Collection

- Garbage collection occurs when one of the following conditions is true:
  - The system has low physical memory.
  - The memory that is used by allocated objects on the managed heap surpasses an acceptable threshold. This threshold is adjusted continuously as the process runs.
  - The `GC.Collect` method is called.
- Developers rarely need to call `GC.Collect` because the garbage collector runs continuously. This method is used primarily for unique situations and testing.

The Collection Process

Garbage collection consists of the following steps:

1. The garbage collector searches for managed objects that are referenced in managed code.
2. It tries to finalize objects that are not referenced.
3. It frees objects that are not referenced and reclaims their memory.

Note: The garbage collector does not recognize references to an object from unmanaged code and might free objects that are being used exclusively in unmanaged code unless explicitly prevented from doing so using the `GC.KeepAlive` method.

Stacks and Heaps

- Variables are stored in either a stack or heap based on their type:
  - Value types (e.g., `int`, `double`, `float`) go on the stack.
  - Reference types (e.g., `String`, `Object`) go on the heap.
    - Value types in classes are stored with the instance of the class on the heap.
- In managed code, the heap is managed by the garbage collector; therefore, it is referred to as the **managed heap**.
- The stack is not handled by the garbage collector—it is reclaimed automatically when a variable passes out of scope (i.e., when a method of code block finishes executing).

Stack Overflows

- Space on the stack is allocated each time that a method is called.
- Nesting too many method calls results in a `StackOverflowException`.
- Beginning programmers often make this error when incorrectly executing recursion, as shown in this code:

```
public void recursiveMethod()
{
    recursiveMethod();
}
```

Assignment

- Create a Console Application project in Microsoft Visual Studio® and enter the Example code found at

  http://msdn.microsoft.com/en-us/library/system.gc.aspx.

- After running the program, discuss the output with a partner. What does it reveal about garbage collection?