MODULE 1: INTRODUCTION TO PROGRAMMING

Collections Part 1 - Lists

Yesterday

- What is an object?
- What is a class?
- What is the relationship between object and class?
- What is a value type or primitive variable?
- What is a reference type variable?
- Why are there two types?

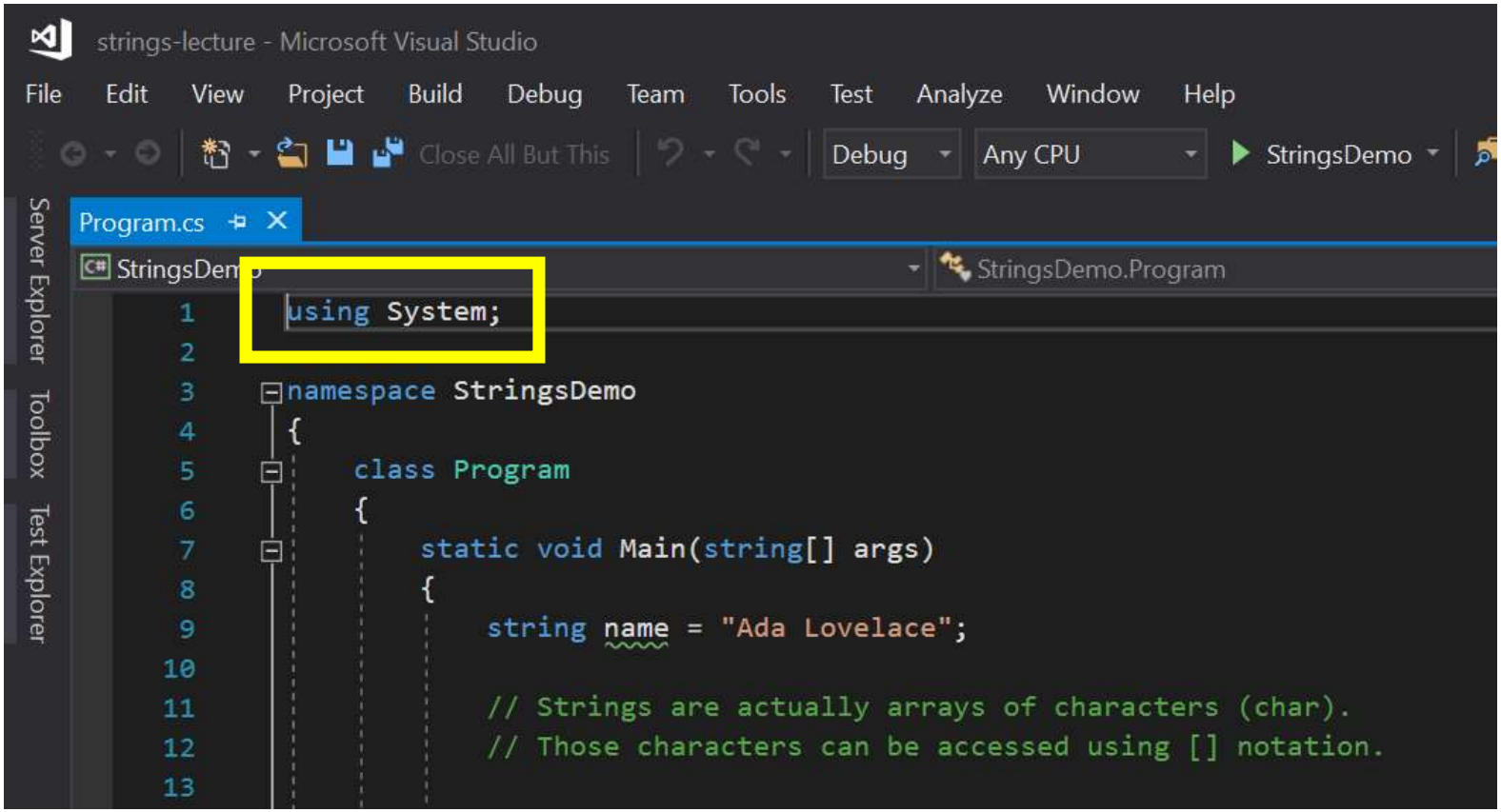
Array recap

- What is an array?
- How do we create an array?
- What are some of the limitations of arrays?
- How do we access elements in an array?
- How do we access the last element in an array?

Collections

- **Collections** classes live in a package or namespace and come from the framework's standard library of classes

Namespaces


strings-lecture - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Close All But This
Debug
Any CPU
StringsDemo
Server Explorer
Toolbox
Test Explorer
Program.cs
StringsDemo.Program
using System;
namespace StringsDemo
{
class Program
{
static void Main(string[] args)
{
string name = "Ada Lovelace";
// Strings are actually arrays of characters (char).
// Those characters can be accessed using [] notation.

Collections: List<T>

- Zero-indexed like an array
- An ordered set of elements accessible by index
- Allows duplicates
- BUT it can grow and shrink as you add and remove items
  - You can add and remove from the middle even

Declaring and Initializing Lists

- List<T>
  - T is just short hand for Type: int, string, double, etc.
- Declaration:
  - List<string> animalNames;
- Initialization:
  - animalNames = new List<string>();
- All in one:
  - List<string> animalNames = new List<string>();

Working with Lists

- List<string> animals = new List<string>();
- animals.Add(“Koala”);
- string aussieAnimal = animals[0];
- animals.Remove(“Koala”);

LET’S CODE!

Foreach

```
foreach (string word in wordsList)
        {
            Console.WriteLine(word);
        }
```

- Convenience method to iterate through collection
- Cannot modify the contents during iteration

Collections: Queue<T>

- Queues are just Lists, but used in a certain way to get a certain result
- A very common data structure in programming
- FIFO - First in, First out

FIFO – Queue<T>

```
Queue<string> animals = new Queue<string>();
```

First In:

```
animals.Enqueue("Panda")                animals.Count is 1
animals.Enqueue("Kangaroo")             animals.Count is 2
```

First Out:

```
string thisAnimal = animals.Dequeue();  animals.Count is 1
```

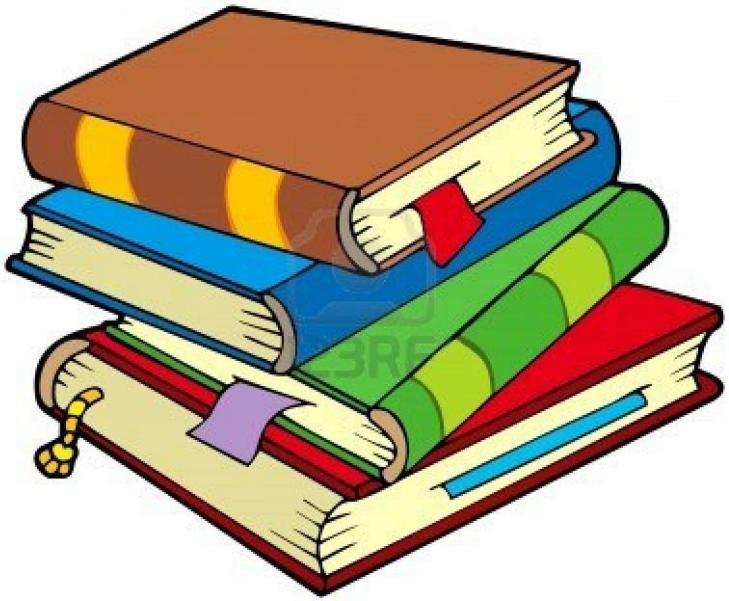
Processing a Queue

- Keep going until you are done with the Queue
- How do you know when you are done?

```
while(animals.Count > 0)
{
        string currentAnimal = animals.Dequeue();
        Console.WriteLine(currentAnimal);
}
```

Collections: Stack<T>

- Stacks are, again, Lists of elements but with different behavior
- Another very common data structure in programming
- LIFO - Last in, First out



LIFO – Stack<T>

```
Stack<string> animals = new Stack<string>();
```

Last In:

```
animals.Push("Panda")                animals.Count is 1
animals.Push("Kangaroo")             animals.Count is 2
```

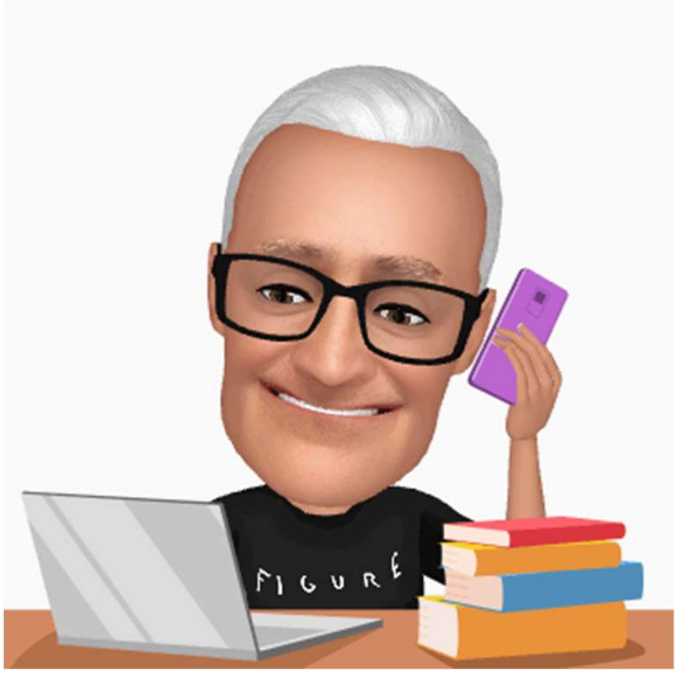
First Out:

```
string thisAnimal = animals.Pop();   animals.Count is 1
```

LET’S CODE!

Collections

- Arrays
- Lists
- Queues
- Stacks



WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:
Collections Part 2