



Module 1 Day 7

Collections, Part 1

Arrays

- A group of similarly typed items
- Elements are accessed by an integer index
- Fixed in size once created

Collection Classes

- Defined in the [System.Collections.Generic](#) namespace
 - A namespace is just an organization mechanism with a hierarchical naming structure
- List: an Array on steroids
- Stack: a last-in, first-out collection
- Queue: a first-in, first-out collection
- ... and many more, some of which we will cover tomorrow...
 - and some of which you will investigate on your own

List

- The collection most like an array
- To create, like any other variable:
 - Declare, Allocate (Instantiate), Initialize

```
// Declare
List<string> daysOfWeek;
// Allocate and initialize
daysOfWeek = new List<string>()
{ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
```

- <xxx> syntax is called a “generic”, and ANY type can be placed there
- List<int>, List<double>, List<Car>
- You can even do a list of lists! (but we’ll spare you that)

List Methods

- Access elements using `listName[index]` syntax, just like arrays
- Add elements
 - `listName.Add(elementToAdd)`
 - `elementToAdd` must be of the appropriate type
 - `listName.Insert(index, elementToAdd)`
- Remove elements
 - `listName.Remove(elementToRemove)`
 - Removes the first occurrence where `(listElement == elementToRemove)`
 - `listName.RemoveAt(index)`

Iterating a List

- The number of elements is called Count
- Since [index] works, we can iterate as usual

```
for (int i = 0; i < daysOfWeek.Count; i++)  
{  
    Console.WriteLine(daysOfWeek[i]);  
}
```

- But there is another way ... foreach

```
foreach (string day in daysOfWeek)  
{  
    Console.WriteLine(day);  
}
```

- So, when to use foreach?

Stack

- Last-in, First-out
- Methods
 - Push
 - Pop
 - Peek
- Foreach
- NO index access!
- NO initializer

```
Stack<int> stack = new Stack<int>();  
stack.Push(1);  
stack.Push(2);  
stack.Push(3);
```

```
while (stack.Count > 0)  
{  
    int i = stack.Pop();  
    Console.WriteLine(i);  
}
```

```
foreach (int i in stack)  
{  
    Console.WriteLine(i);  
}
```

Driveway parking, Undo, Browser Back

Queue

- First-in, First-out
- Methods
 - Enqueue
 - Dequeue
 - Peek
- Foreach
- NO index access!
- NO initializer

```
Queue<int> queue = new Queue<int>();  
queue.Enqueue(1);  
queue.Enqueue(2);  
queue.Enqueue(3);
```

```
while (queue.Count > 0)  
{  
    int i = queue.Dequeue();  
    Console.WriteLine(i);  
}
```

Store checkout, Print queue