Collections

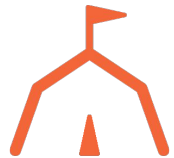
What Are Collections?

- Similar to an array, a collection is used to hold other objects.
- They are also more flexible and **less efficient** than arrays.

Collection Types

Non-Generic Collections

- Non-Generic collections are collection without strong typing. They have no safety net.
- Any type of data may go in and any type may come out, so often you will need to do parsing or casting. The results of this conversion are unpredictable



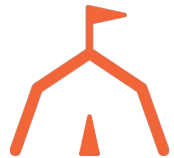
Non-Generic Collections Continued

- Many of these will appear in legacy code, but in general tend to be avoided in C# in favor of Generics.
- To put it another way, they are loosely typed in a strongly typed language.

Examples of Non-Generics

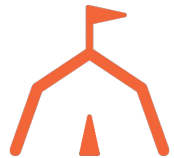
Arraylists

Hashtables



ArrayList

An array list is a collection that’s similar to an array, but it can change its capacity as elements are added or removed.



ArrayList

Example

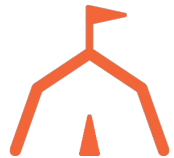
```
// The ArrayList will store elements as objects
ArrayList numbers = new ArrayList();
numbers.add(5);
        foreach (int i in numbers)
{
        Console.WriteLine(i);
}
```

ArrayList Methods

- Add(object): Adds an object to the end of the list.
- Count: Returns the number of elements in the list
- Insert(index, object): Adds an object at a specific location.

Hashtable

- A Hashtable where elements are organized based on Key-Value pairs.
- Keys are used to access elements or values.



Hashtable

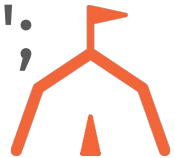
Example

```
Hashtable ht = new Hashtable();

ht.Add("001", "John");
ht.Add("002", "Paul");
Console.WriteLine(ht["001"]);
```

Generic Collections

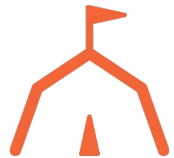
- Generics allows us to create typed collections, which are strongly typed.
- Like arrays, you may set their typing to any data type.
- To give generics a typing, use the angle brackets like so: List<int> nums;
- To include, use "System.Collections.Generic";



Generic Collections Examples

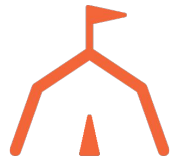
Lists

Dictionaries



List

A list is a generic collection that’s similar to an array, but it can change its capacity as elements are added or removed.
A list is the most common collection you’ll see in C# and is a generic collection.



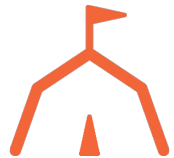
List

Example:

```
List<int> numbers = new List<int>();
numbers.Add(5);
foreach(int i in numbers)
{
    Console.WriteLine(i);
}
```

List Methods

- Add(object): Adds an object to the end of the list.
- Count: Returns the number of elements in the list
- Insert(index, object): Adds an object at a specific location.



Array Syntax on Lists

Lists in C#, on top of methods, may also use array syntax to store and retrieve elements:

```
numbers [0] = 10;
int num = numbers[0];
```

Arrays vs. Lists

Common question: when do I use an Array or a List? What's the difference?
Answer: They have one main difference - arrays are a set size, and lists may shrink or grow. Otherwise they may be used interchangeably.

Dictionaries

- A Dictionary is structured and used in the same way as a HashTable: Key Value pairs
- The key distinction is that Dictionaries are strongly typed so you always know what data types you're working with

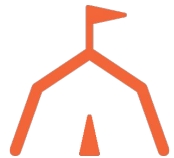
Dictionary Example

```
Dictionary<string, bool> tasty = new Dictionary<string, bool>();
tasty.Add("Chicken Curry", true);
tasty.Add("Asparagus", false);
Console.WriteLine(tasty["asparagus"]);

foreach(KeyValuePair<string, bool> kvp in tasty)
{
    if(kvp.Value == true)
    {
        Console.WriteLine($"{kvp.Key} is tasty!");
    }
    else
    {
        Console.WriteLine($"{kvp.Key} is icky!");
    }
}
```

Other Notable Collections

- A Stack is a LIFO (Last in First Out) data structure.
- A Queue is a FIFO (First in First Out) data structure.
- These are important since they are used to structure many things in tech



Recap

- Know what collections are.
- Know to put using system.collections.generics to import lists
- Generics and their role in collections
- How to define and use Lists.
- How to define and use Dictionaries.