The Learning House

427 S. 4th Street #300

Louisville KY 40202

The C# Collections

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Why Collections?

For many applications, we need to create and manage groups of related objects.

There are two ways we can group objects. The first is to create an array of objects, and the second is to create a collection of objects.

Arrays vs Collections

Arrays are useful for…

Creating and working with a fixed number of objects.

Cases when you will not need to insert items at the beginning or middle of the list.

Collections are useful for…

Creating and working with an dynamically changing or unknown number of objects.

When you want to use a key for quick retrieval of a specific item, instead of an array index.

A Collection is a Class Object

This means you must declare a new collection before you can add any elements to the collection.

The most recent versions of the .NET framework include generic collections which enforce type safety. Prior versions had collections like the ArrayList which were not type-safe.

Boxing and Unboxing

DEMO : ARRAYLIST, HASHTABLE, QUEUE, STACK

System.Collections.Generic Namespace

While the System.Collections namespace is useful, all of that casting and converting from objects is pretty cumbersome.

Wouldn’t it be nice if we could tell our collections what type was in it and then not have to convert things?

With Generic Collections (.NET 3.0), we can!

Classes in System.Collections.Generic

Class	Purpose
Dictionary<TKey, TValue>	Represents a generic collection of keys and values. We have a unique key for each value that we want to search for quickly. It protects us from adding the same key/pair twice.
List<T>	A sequential list of items that will automatically resize itself. This is by far the most popular collection.
Queue<T>	First-in, first-out list
Stack<T>	Last-in, first-out list

List<T>, Dictionary<T>, Stack<T> and Queue<T>

DEMO

Conclusion on Generic Collections

- They flat-out rock! We can manage many different sizes of collections, add and remove items at will, and don’t have to write any plumbing code to do it.

Lab Exercise

Write an application that rolls two dice 100 times. Keep track of the number of times each value (2-12 with standard dice) comes up. Print all of the values and the number of times they were rolled.

Lab Exercise

Write the guessing game, but this time keep track of all of the guesses. Do not let the user guess the same number twice. Print all the guessed numbers at the end of the game.