C# Fundamentals (2)

Hack Your Future (https://hackyourfuture.be/)

Axxes Coaches

Hannes Lowette
Joren Vandekerckhove
Kevin Boets
Stijn Roscam
Thibaut Humblet
Francis Didden

Collections

“When you need X of the same variable, you are using a collection”

- **Arrays**
  Indexed, fixed-length lists of variables
- **Multi-dimensional arrays**
  Indexed, multi-dimensional, fixed-length lists of variables
- **Lists**
  Ordered, flexible-length lists of variables
- **Dictionary**
  key-value pairs

Classes

“A blueprint to create objects, which hold state (fields, properties) and behavior (methods).”

- **Defining classes**
  The `class` keyword can be used to create new classes
- **Creating objects**
  The `new` keyword can be used to create new objects
- **Fields**
  Inside the class, you can create data fields that will be unique to the object
- **Properties**
  Using properties, you get more control over the manipulation of internal fields from the outside
- **Constructors**
  When creating new objects, you can define the instantiation behavior

Inheritance

“When you have a class that refines the concepts of another class”

- **Inheritance**
  Parent-child relations between classes
- **Single inheritance**
  There is only 1 parent class, ever!
- **Override & virtual**
  Replacing concepts from the parent class
- **Abstract classes**
  Classes that cannot be instantiated, must be inherited from first.
  They can force implementation of certain members

Access modifiers

“Control where things can be used”

Classes

- **Internal:** in the same assembly
- **Public:** everywhere

Class members

- **Private:** Only inside the class
- **Protected:** Inside the class, and classes that inherit from it
- **Public:** From everywhere

Static, Const & Readonly

- **Static (method, property, field)**
  There is only 1 instance of this, shared across all instance
- **Static (class)**
  This class only contains static members. Cannot be instantiated.
- **Const**
  The value we assign to this gets locked in by the compiler, and cannot be changed in runtime.
- **Readonly**
  The value we assign to this gets locked in by the constructor, and cannot be changed after.

Further study

Using Objects & classes:

https://docs.microsoft.com/en-us/shows/csharp-101/csharp-object-oriented-programming-objects-and-classes

Defining Methods & Members:

https://docs.microsoft.com/en-us/shows/csharp-101/csharp-object-oriented-programming-methods-and-members