C# Advanced – Class 1

Material recap: Loops, Collections, Enums, Classes, Getters and Setters, Constructors, Exception Handling

New material: Indexers, Copy constructors

Trainer: Miodrag Cekikj – cekicmiodrag@gmail.com

Assistant: Andrej Chichakovski – andrejchichak@gmail.com

SEAVUS EDUCATION *and* DEVELOPMENT CENTER

Warm up

- Variables, data types
- Constants, Enums
- Branching
- Iterations (Loops)
- Collections
- Classes
- Indexers (new)
- Constructors (recap + new)

Variables and data types

- Variables
  - A variable is nothing but a name given to a storage area that our programs can manipulate.
  - Each variable has a specific type, which determines the size and layout of the memory, the range of values that can be stored within that memory and the set of operations that can be applied to the variable.
  - int i = 0;
  - string x = ”Hello World”;
- Data types
  - Data type tells a C# compiler what kind of value a variable can hold. C# includes many in-built data types for different kinds of data (string, int, double, decimal …)

Variables and data types (2)

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0 \text{ to } 28}$	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

Constants and Enums

- Constants
  - Constants are fields whose values are set at compile time and can never be changed.
  - const double Pi = 3.14;
- Enums
  - Sets of distinct named constants
  - Logical connection among various constants
  - enum DaysOfWeek {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
- DEMO

Branching

- Unconditional - happens every time the branch point is reached (unconditionally)
  - Method Call()
  - Break
  - Continue
  - GoTo
- Conditional - based on the evaluation of certain conditions that occur at runtime
- Short-Circuit Evaluation - If first term is false in the complex condition, the compiler will short-circuit the evaluation, the second test will never be performed
  - If
  - If Else
  - Else If
  - Switch

Iterations (Loops)

- Loops
  - While
  - Do While
  - For
  - Loop with GO - Spaghetti Code, most experienced programmers avoid using goto to create loops

Collections

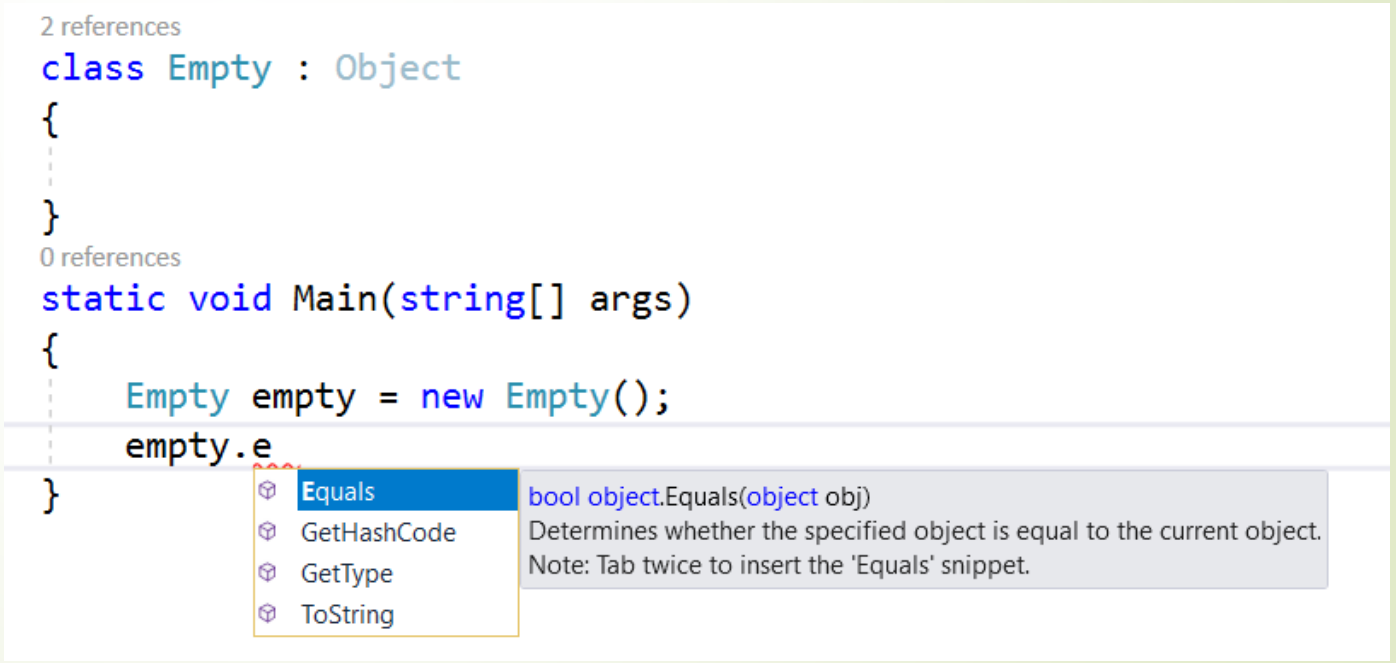
- Generic
  - Array – fixed size
  - List – dynamically increased
  - Dictionary – key, value pair, (unordered)
  - Queue – FIFO, (unordered)
  - Stack – LIFO, (unordered)

  - **Exercise** – Write a program that will receive an input from a console application that will give the capital city of a country. Previously the list should be built-in into a suitable structure (choose one from above).

The Object class in C#

- Supports all classes in the .NET Framework class hierarchy and provides low-level services to derived classes. This is the ultimate base class of all classes in the .NET Framework; it is the root of the type hierarchy.
- Demo

1. Equals()
2. GetHashCode()
3. GetType()
4. ToString()



Getters and Setters - recap

- Demo
  - Class1 – shortest way of using getters and setters
  - Class2 – shorter way of using getters and setters
  - Class3 – the traditional/old way of using getters and setters

Indexers for classes

- Indexers allow instances of a class or struct to be indexed just like arrays. The indexed value can be set or retrieved without explicitly specifying a type or instance member.
- *When a class has an indexer it can be used in a similar manner as array. Objects of the class can use array-style notation to present multiple values.*
- An **indexer** allows an object to be indexed such as an array. When you define an indexer for a class, this class behaves similar to a **virtual array**.

- *Demo1*
- *Demo2*

Constructors - overview

- Instance constructor
  - The “default” constructor for the classes
- Private constructor
  - It’s a special instance of the instance constructor. You cannot instantiate an object because the constructor is private.
- Static constructor
  - This constructor is used to initialize any static data, or to perform one-time action, like setting constants values.
- Copy constructor
  - There is no “Copy” constructor, but a way to make simulation. That constructor is called Copy constructor.