Pentalog

Collections in C#

PentaStagiu Remote Brașov

November, 2019 – March, 2020

DATA ANALYSIS
UX/UI
FRONT-END
TECHNOLOGY
NEAR/OFFSHORE
AGILITY
BACK-END
SPRINT
KANBAN
CAMPAIGNS
GROWTH HACKING
SCRUM
BACKLOG
DEVOPS
DESIGN
SEO
CONTINUOUS INTEGRATION
MOBILE
QA
AUTOMATION
RESPONSIVE
UNIT TESTING

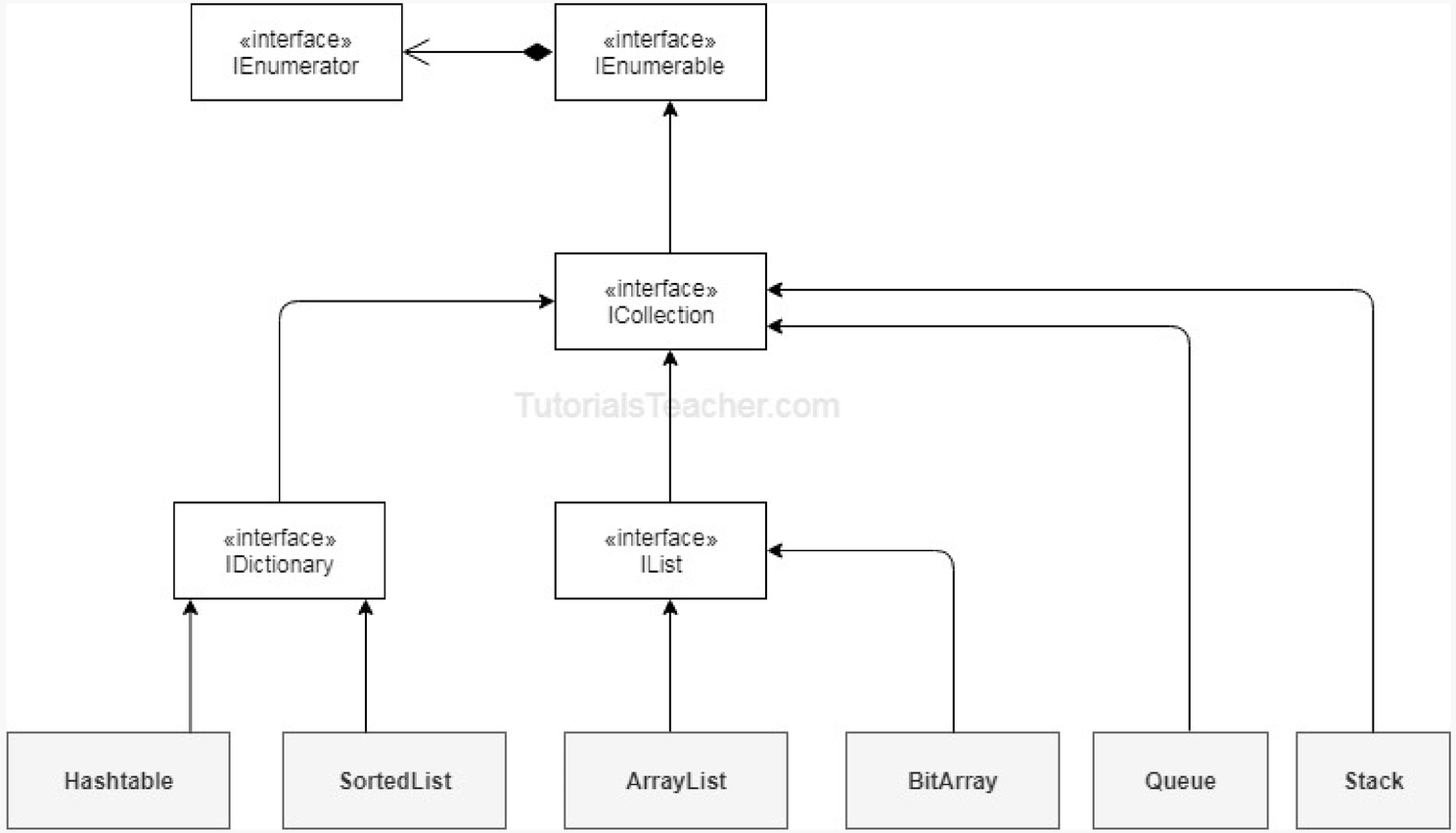
Agenda

- What collection is?
- System.Collections
- System.Collections.Generic

Collections

- collections provide a more flexible way to work with groups of objects
- the group of objects you work with can grow and shrink dynamically
- a collection is a class, so you must declare an instance of the class before you can add elements to that collection
- If your collection contains elements of only one data type, you can use one of the classes in the System.Collections.Generic namespace. A generic collection enforces type safety so that no other data type can be added to it.

System.Collections



- includes the interfaces and classes for the non-generic collections
- *IEnumerator*, *IEnumerable*, and *ICollection* are the top level interfaces for all the collections in C#

IEnumerator

- IEnumerator is the base interface for all non-generic enumerators
- Using foreach is recommended instead of directly manipulating the enumerator
- Enumerators can be used to read data in the collection, but they cannot be used to modify the underlying collection
- *Methods:*
  - MoveNext() - Sets the enumerator to the next element of the collection; returns true if the enumerator was succesfuly set to the next element and false if has reached the end of the collection
  - Reset() - Sets the enumerator to its initial position

(code example)

IEnumerable

- Exposes an enumerator, which supports a simple iteration over a non-generic collection.
- Using foreach is recommended instead of directly manipulating the enumerator

- Enumerators can be used to read data in the collection, but they cannot be used to modify the underlying collection

Methods:

- GetEnumerator() – returns an enumerator that iterates through a collection

Exension methods:

- Cast<TResult> - cast the elements of an Ienumerable to the specified type
- OfType<TResult> - filters the elements of an IEnumerable based on a specified type
- AsParallel – enable parallelization of a query
- AsQueryable – convert IEnumerable to IQueryable (System.Linq)

ICollection

- is the base interface for all the collections that defines sizes, enumerators, and synchronization methods for all non-generic collections

- The **Queue** and **Stack** collection implement ICollection inferface

Properties/Methods/Extension methods:

https://docs.microsoft.com/en-us/dotnet/api/system.collections.icollection?view=netframework-4.7.2

IList

- includes properties and methods to add, insert, remove elements in the collection and also individual element can be accessed by index

- **ArrayList** and **BitArray** collections implement IList interface

Properties/Methods/Extension methods:

https://docs.microsoft.com/en-us/dotnet/api/system.collections.ilist?view=netframework-4.7.2

(code example)

IDictionary

- represents a non-generic collection of key/value pairs
- The **Hashtable** and **SortedList** implement IDictionary interface and so they store key/value pairs

Properties/Methods/Extension methods:

https://docs.microsoft.com/en-us/dotnet/api/system.collections.idictionary?view=netframework-4.7.2

Non-generic collections

ArrayList	ArrayList stores objects of any type like an array. However, there is no need to specify the size of the ArrayList like with an array as it grows automatically.
SortedList	SortedList stores key and value pairs. It automatically arranges elements in ascending order of key by default. C# includes both, generic and non-generic SortedList collection.
Stack	Stack stores the values in LIFO style (Last In First Out). It provides a Push() method to add a value and Pop() & Peek() methods to retrieve values. C# includes both, generic and non-generic Stack.
Queue	Queue stores the values in FIFO style (First In First Out). It keeps the order in which the values were added. It provides an Enqueue() method to add values and a Dequeue() method to retrieve values from the collection. C# includes generic and non-generic Queue.
Hashtable	Hashtable stores key and value pairs. It retrieves the values by comparing the hash value of the keys.
BitArray	BitArray manages a compact array of bit values, which are represented as Booleans, where true indicates that the bit is on (1) and false indicates the bit is off (0).

System.Collections.Generic

- The System.Collections.Generic namespace contains interfaces and classes that define generic collections, which allow users to create strongly typed collections that provide better type safety and performance than non-generic strongly typed collections

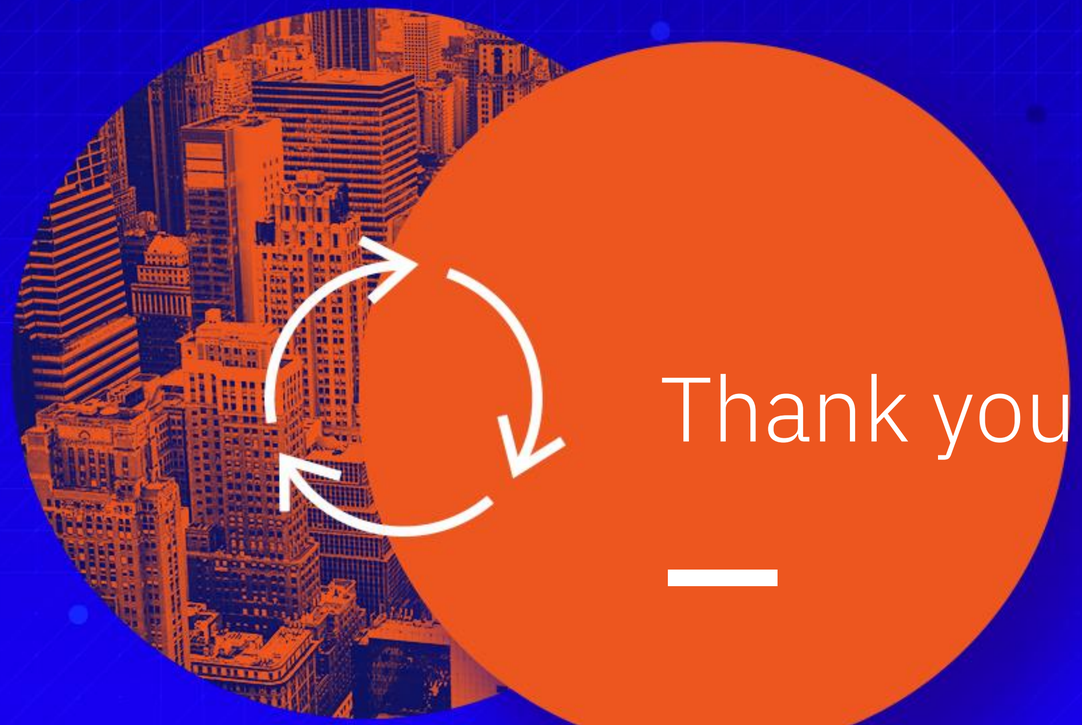
Generic collections

List<T>	Generic List<T> contains elements of specified type. It grows automatically as you add elements in it.
Dictionary<TKey,TValue>	Dictionary<TKey,TValue> contains key-value pairs.
SortedList<TKey,TValue>	Hashset<T> contains non-duplicate elements. It eliminates duplicate elements.
Hashset<T>	Hashset<T> contains non-duplicate elements. It eliminates duplicate elements.
Queue<T>	Queue<T> stores the values in FIFO style (First In First Out). It keeps the order in which the values were added. It provides an Enqueue() method to add values and a Dequeue() method to retrieve values from the collection.
Stack<T>	Stack<T> stores the values as LIFO (Last In First Out). It provides a Push() method to add a value and Pop() & Peek() methods to retrieve values.

Reference

- https://docs.microsoft.com/en-us/dotnet/api/system.collections?view=netframework-4.7.2
- https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=netframework-4.8

Pentalog



Thank you!

DATA ANALYSIS

UX/UI

FRONT-END

TECHNOLOGY

NEAR/OFFSHORE

AGILITY

BACK-END

SPRINT

KANBAN

CAMPAIGNS

GROWTH HACKING

SCRUM

BACKLOG

DEVOPS

DESIGN

SEO

CONTINUOUS INTEGRATION

MOBILE

QA

AUTOMATION

RESPONSIVE

UNIT TESTING