System.Collections

LIGHTNING TALK 2017-09-22

BY ARNOLD BARNA

Common methods

- All collections provide methods for
  - adding,
  - removing or
  - finding items in the collection

  - And many more …

A few types

- *Access by index:*
  - ***List**: basic, versatile*
  - ***HashSet**: only unique elements*
  - ***Queue**: first-in, first-out*
  - ***Stack**: last-in, first-out*
- *Access by key:*
  - ***Dictionary:** key/value pairs*

List

```
List<string> myList = new List<string> { "one",
    "two",
    "three" };
myList.Add("four");
myList.Add("four");
myList.RemoveAt(2);
```

```
Number of elements after creating: 3
Elements after creating:
one two three

Number of elements after adding "four": 4
Elements after adding "four":
one two three four

Number of elements after adding "four" ONCE MORE: 5
Elements after adding "four" ONCE MORE:
one two three four four

Number of elements after removing element #3: 4
Elements after removing element #3:
one two four four
```

Queue

```
List<string> myList = new List<string> {
    "one",
    "two",
    "three" };
Queue<string> myQueue = new Queue<string>(myList);
myQueue.Enqueue("four");
myQueue.Enqueue("four");
string dequeuedElement = myQueue.Dequeue();
```

```
Number of elements after creating: 3
Elements after creating:
one two three

Number of elements after enqueuing "four": 4
Elements after enqueuing "four":
one two three four

Number of elements after enqueuing "four" ONCE MORE: 5
Elements after enqueuing "four" ONCE MORE:
one two three four four

Number of elements after dequeuing last element: 4
Elements after dequeuing last element:
two three four four

Variable dequeuedElement: one
```

Stack

```
List<string> myList = new List<string> {
    "one",
    "two",
    "three" };
Stack<string> myStack = new Stack<string>(myList);
myStack.Push("four");
myStack.Push("four");
string poppedElement = myStack.Pop();
```

```
Number of elements after creating: 3
Elements after creating:
three two one

Number of elements after pushing "four": 4
Elements after pushing "four":
four three two one

Number of elements after pushing "four" ONCE MORE: 5
Elements after pushing "four" ONCE MORE:
four four three two one

Number of elements after popping last element: 4
Elements after popping last element:
four three two one

Variable poppedElement: four
```

Dictionary

```
Dictionary<string, string> englishMonarchs = new Dictionary<string, string>()
{
    { "Elizabeth II", "Elizabeth Alexandra Mary" },
    { "George VI", "Albert Frederick Arthur George"},
    { "Edward VII", "Albert Edward" }
};

englishMonarchs.Add("Victoria", "I couldn't google her name");
englishMonarchs["Victoria"] = "Alexandrina Victoria";
```

```
Number of elements after creating: 3
Elements after creating:
Key: Elizabeth II Value: Elizabeth Alexandra Mary
Key: George VI Value: Albert Frederick Arthur George
Key: Edward VII Value: Albert Edward

Number of elements after adding "Victoria": 4
Elements after adding "Victoria":
Key: Elizabeth II Value: Elizabeth Alexandra Mary
Key: George VI Value: Albert Frederick Arthur George
Key: Edward VII Value: Albert Edward
Key: Victoria Value: I couldn't google her name

Number of elements after correcting "Victoria": 4
Elements after correcting "Victoria":
Key: Elizabeth II Value: Elizabeth Alexandra Mary
Key: George VI Value: Albert Frederick Arthur George
Key: Edward VII Value: Albert Edward
Key: Victoria Value: Alexandrina Victoria
```

HashSet

```
List<string> myList = new List<string> {
    "one",
    "two",
    "three" };
HashSet<string> myHashSet = new HashSet<string>(myList);
myHashSet.Add("four");
myHashSet.Add("four");
myHashSet.Remove("two");
```

```
Number of elements after creating: 3
Elements after creating:
one two three

Number of elements after adding "four": 4
Elements after adding "four":
one two three four

Number of elements after adding "four" ONCE MORE: 4
Elements after adding "four" ONCE MORE:
one two three four

Number of elements after removing "two" element: 3
Elements after removing "two" element:
one three four
```

List – Josephus Problem

```
private static int FindSurvivorSeat_WithList(int numberOfParticipants)
{
    // value: original pos
    var gamers = new List<int>(Enumerable.Range(1, numberOfParticipants));
    while (gamers.Count > 1)
    {
        gamers.RemoveAt(1); // second gets killed
        gamers.Add(gamers.First()); // first goes at the end
        gamers.RemoveAt(0); // first goes at the end
    }
    return gamers.First();
}
```

Queue (FIFO) – **Josephus Problem**

```
private static int FindSurvivorSeat_WithQueue(int numberOfParticipants)
{
    var gamers = new Queue<int>(Enumerable.Range(1, numberOfParticipants));
    while (gamers.Count > 1)
    {
        gamers.Enqueue(gamers.Dequeue()); // first goes at the end
        gamers.Dequeue(); // second gets killed
    }
    return gamers.First();
}
```

Thank you for the attention

Available on GitHub:

Code:

- https://github.com/greenfox-academy/**bramble100**/tree/master/week-03/**lightning-talk**

Slides:

- https://github.com/greenfox-academy/**bramble100**/tree/master/week-03/**lightning-talk**