DEVELHOPE

Data Collections

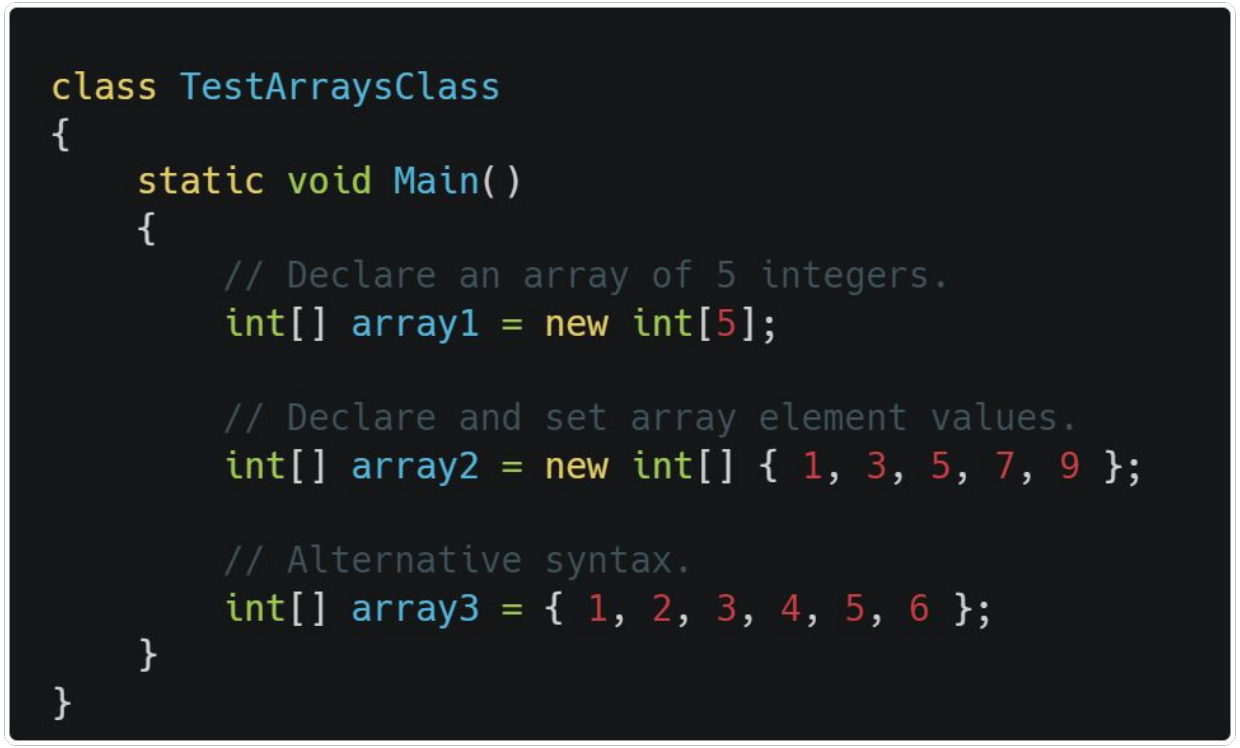
Unit Slides

Matthijs van der Veer

What is an array

- Fixed size data structure
- Accessed by an index
- 0 index based. The first item is index 0!
    - In an array of 5 items, the last item has index 4.
- Specify the type of array you’re making.
    - Array of string
    - Array of int
    - Array of anything!

Arrays

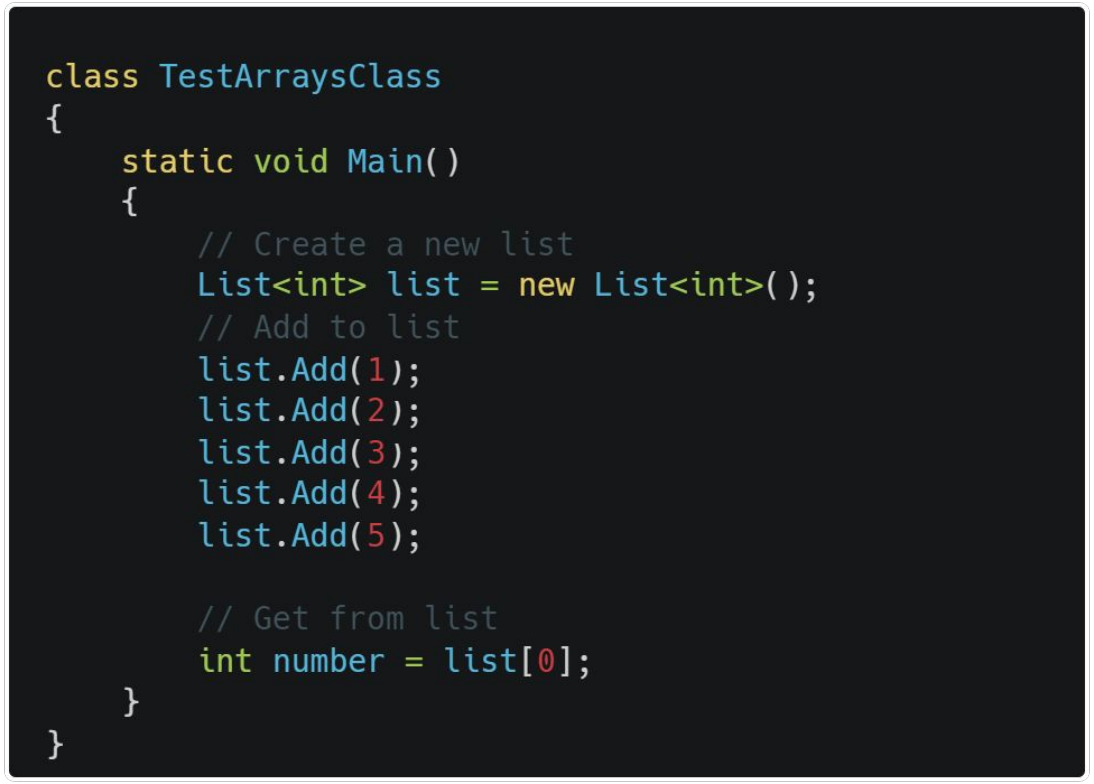


```
class TestArraysClass
{
    static void Main()
    {
        // Declare an array of 5 integers.
        int[] array1 = new int[5];

        // Declare and set array element values.
        int[] array2 = new int[] { 1, 3, 5, 7, 9 };

        // Alternative syntax.
        int[] array3 = { 1, 2, 3, 4, 5, 6 };
    }
}
```

What is a List?

- Internally it stores items in an array.
- Has Add/Remove methods.
- Size can change, the list will resize the array.
- You can access items by index, like an array.
- `List<string>` or `List<int>` or `List<Calculator>`.
- Optimised for sorting and searching.

Lists



```
class TestArraysClass
{
    static void Main()
    {
        // Create a new list
        List<int> list = new List<int>();
        // Add to list
        list.Add(1);
        list.Add(2);
        list.Add(3);
        list.Add(4);
        list.Add(5);

        // Get from list
        int number = list[0];
    }
}
```

Data Collections

- List
- Dictionary
- Queue
- Stack
- StringCollection
- StringDictionary
- SortedDictionary
- ConcurrentDictionary
- ListDictionary
- ArrayList
- SortedList
- NameValueCollection
- KeyedCollection
- ImmutableDictionary
- ImmutableSet
- HashTable
- HashSet
- ImmutableSortedSet
- ImmutableSortedDictionary
- LinkedList
- ConcurrentQUeue
- ImmutableQueue
- ConcurrentStack
- ImmutableStack

Commonly Used Data Collections

- Array
- List
- ArrayList
- Queue
- ConcurrentQueue
- Stack
- ConcurrentStack
- LinkedList
- Hashtable
- Dictionary
- ConcurrentDictionary
- HashSet

What is an ArrayList?

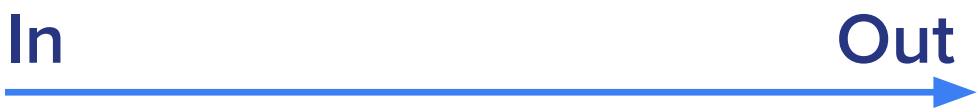
- Like a List<T>, however:
- No type safety
- Slightly less performant
- Only used in very specific scenarios

What is a Queue?

- First-In-First-Out
- Uses Generics
- Enqueue()
- Dequeue()
- Peek()

What is a Queue?

What is a Queue?



Enqueue!

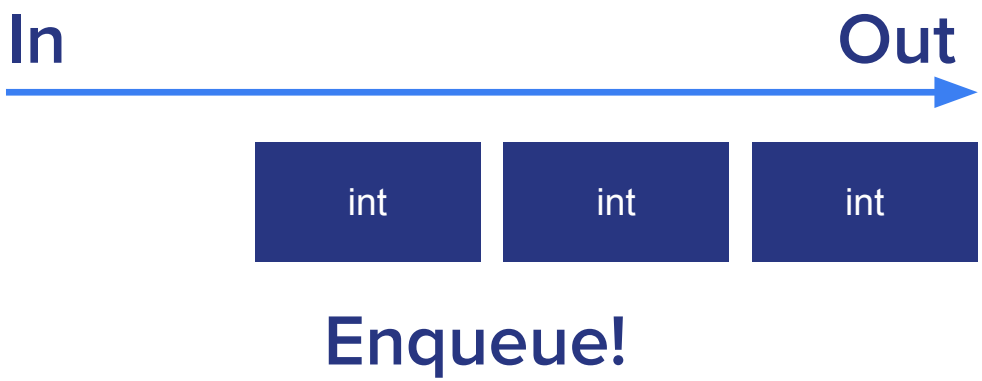
What is a Queue?

Enqueue!

What is a Queue?

Enqueue!

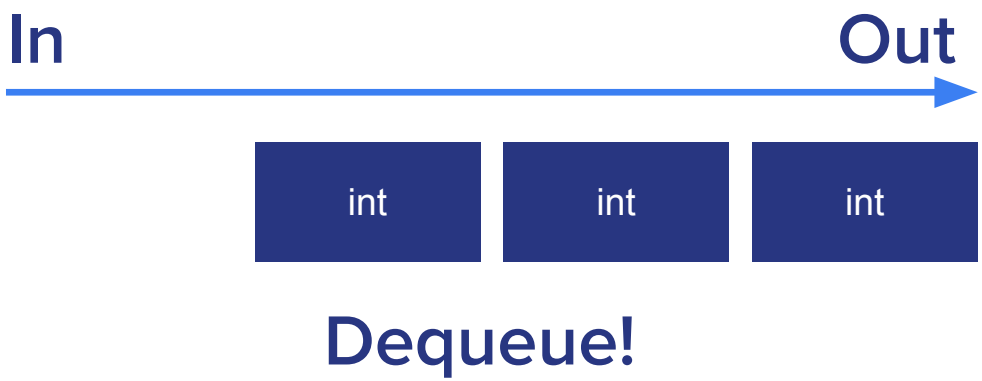
What is a Queue?



What is a Queue?

Enqueue!

What is a Queue?



What is a Queue?

What is a Queue?

Dequeue!

What is a Queue?

What is a ConcurrentQueue?

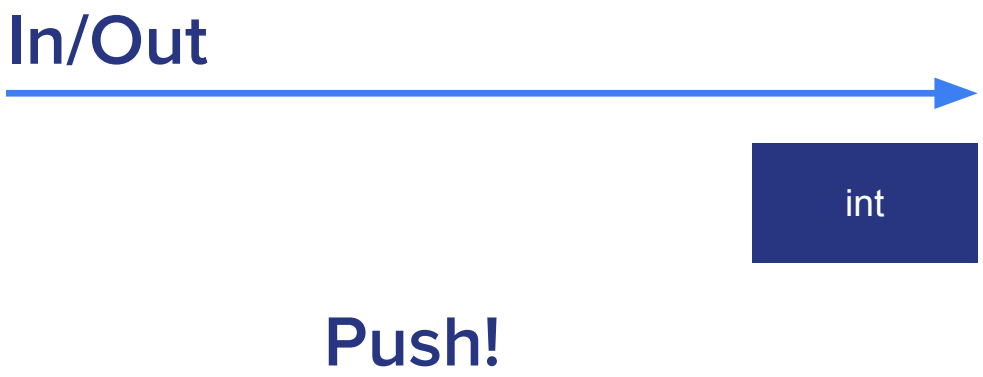
- Similar to a Queue
- Thread safety
- Slightly slower than a Queue
- TryDequeue

What is a Stack?

- Last-In-Last-Out (LIFO)
- Generic
- Easily to reverse items
- Pop/Push/Peek

What is a Stack?

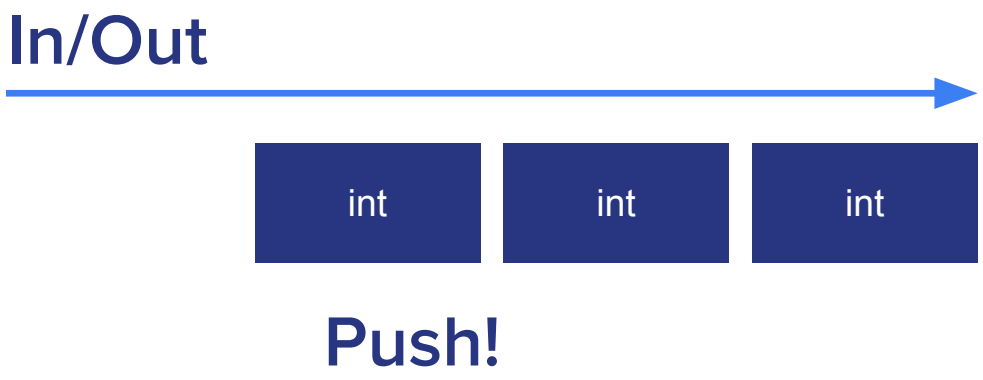
What is a Stack?



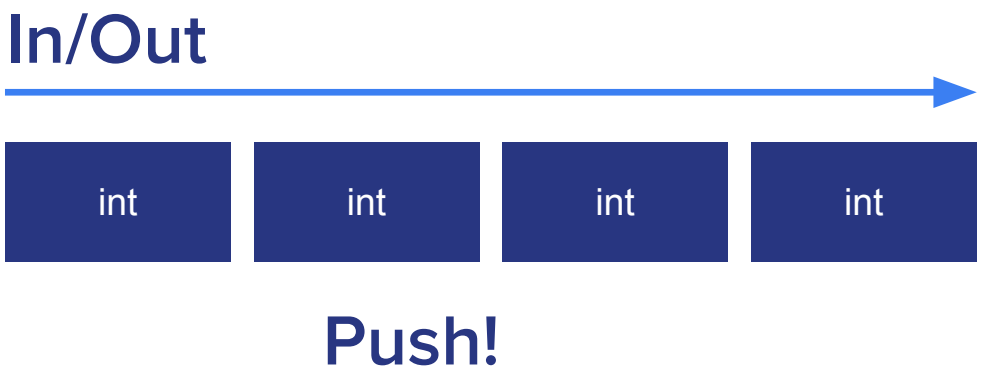
Push!

What is a Stack?

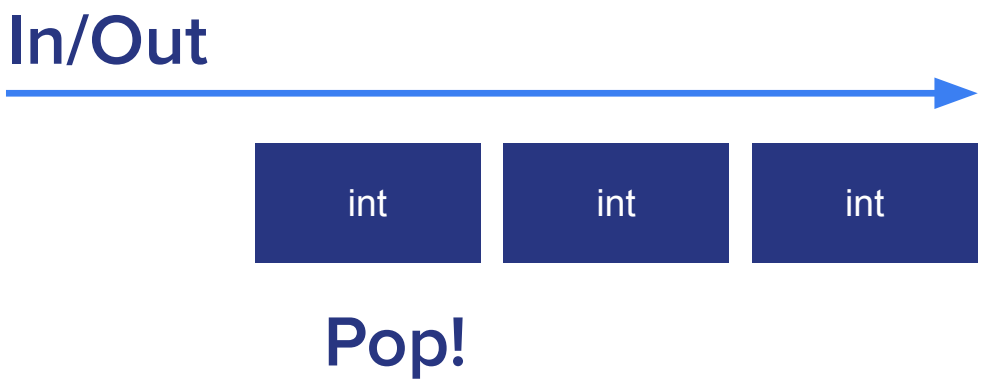
What is a Stack?



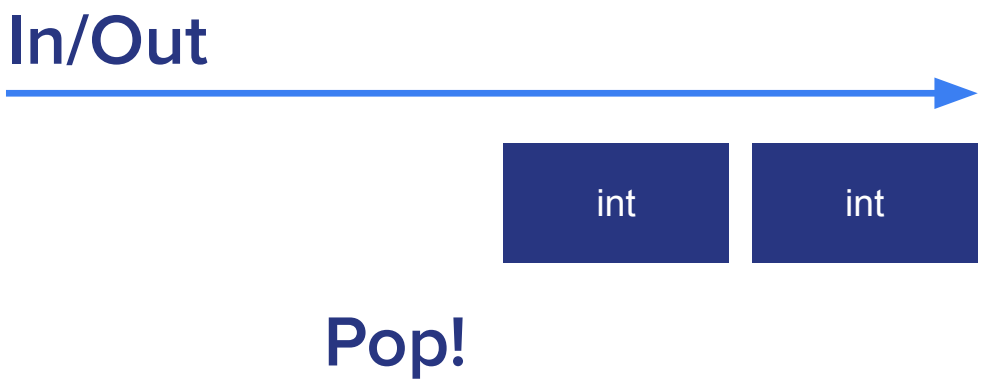
What is a Stack?



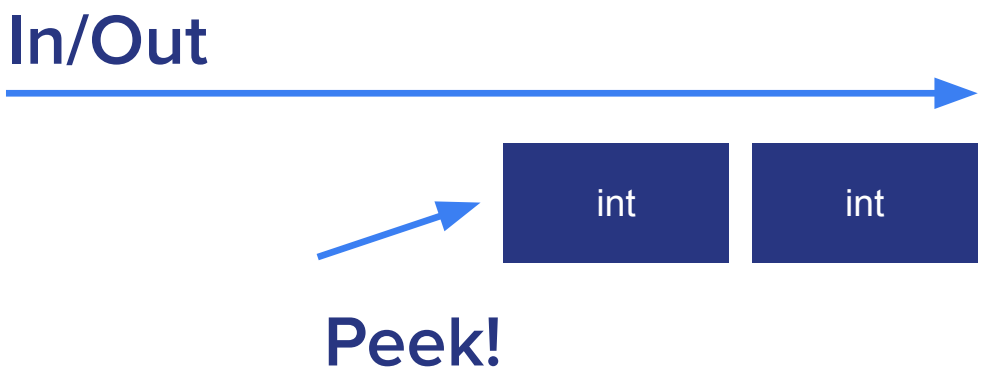
What is a Stack?



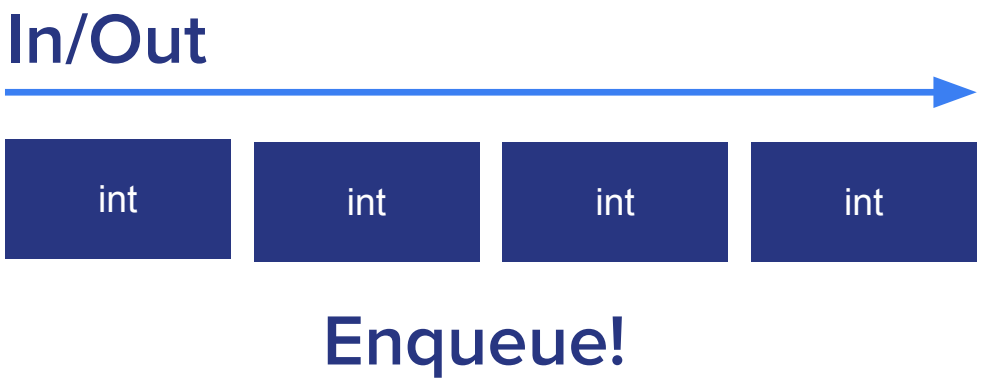
What is a Stack?



What is a Stack?



What is a Stack?



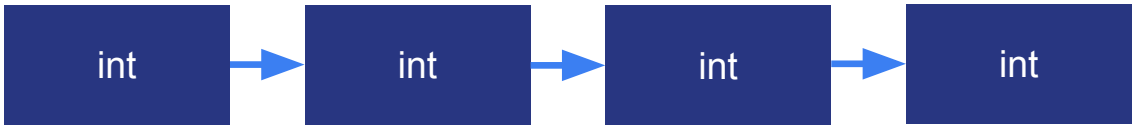
What is a ConcurrentStack?

- Built for concurrency

What is a LinkedList?

- Nodes linking to each other
- Size can change
- Size changing is very efficient!
- Can be less efficient when searching

What is a LinkedList?



What is a HashTable?

- Key/Value pairs
- Efficient lookup
- Unordered
- Duplicates not allowed

What is a Dictionary?

- A type safe HashTable
- Key-Value pairs
- Fast lookups
- There’s also a ConcurrentDictionary

What is a HashSet?

- Only values, no keys
- No duplicates allowed
- Very fast lookups
- Set operations
  - Union
  - Intersection
  - Difference

What kind of loops do we have?

- for
  - Has an index that can be updated throughout the loop
  - Contains a condition for when to stop
- foreach
  - Automatically iterate through a data collection
  - Creates a variable for the current item in the collection
- while
  - “run this code while a condition is true”
  - Don’t forget to break out of the loop or it will run forever.
- do while
  - “run this code while a condition is true, but at least once!”
  - Don’t forget to break out of the loop or it will run forever.

for loop

foreach loop

```
string[] names = { "John", "Alice", "Bob" };

foreach (string name in names)
{
    Console.WriteLine("Name: " + name);
}
```

while loop

```
int count = 0;
while (count < 5)
{
    Console.WriteLine("Iteration: " + count);
    count++;
}
```