Queues

First in, first out

Much like `List` and `Dictionary`, `Queue` is named for its ability to append data to the end of a list and remove data from the beginning.

You can think of this as a queue of people waiting to be served by a teller.

Another term for this kind of data structure is "first-in, first-out."

Not random access

Unlike a `Listor` `Dictionary`, a `Queue` does not allow random access to its elements. That is we cannot use `[]` to access any element we want.

We may only use `Enqueue` and `Dequeue` to add and remove data from the queue.

Enqueue

To add elements to the end of a queue, we use the Enqueue method. Much like `Add` appends things to a `List`, Enqueue adds things to a Queue. However, we can *only* add to the end of the queue.

```
var students = new Queue<string>();
students.Enqueue("Mary");
students.Enqueue("Bill");
students.Enqueue("Paul");
students.Enqueue("Sandra");
students.Enqueue("Thomas");
```

Looping through a queue

Much like our `List` and `Dictionary` we can use the `foreach` method to loop through the elements of a queue.

```
foreach (var student in students)
{
        Console.WriteLine($"Hello {student}");
}
```

Cannot use '[]`

If we try to access a Queue element using `[]`, we will get an error.

```
// This code will not work, it won't even compile.
var student = students[0];
```

Removing an element

We can use Dequeue to remove an element from the beginning of a queue. We can only remove the first element in a queue.

```
var student = students.Dequeue();
Console.WriteLine($"Goodbye {student}");
Console.WriteLine($"There are now {student.Count} students in the queue.");
```

This will print `Goodbye Mary` and `There are now 4 students in the queue.`

Peeking in on a queue

We can also see what element is at the beginning of a queue using the `Peek` method. While `Dequeue` removes an element from the queue, `Peek` will not.

```
var student = students.Peek();
Console.WriteLine($"You are up next {student}");
```

This will print `You are up next Bill` and `There are now 4 students in the queue.` since we didn't remove `Bill`.

Does a queue contain a specific element?

We can also use `Contains` to check if a queue contains a specific element.

```
var isThomasThere = students.Contains("Thomas");
Console.WriteLine($"Thomas is there? {isThomasThere}");
```

Clear out a queue

```
students.Clear();
Console.WriteLine($"There are now {students.Count} students in the queue.");
```

This will print `There are now 0 students in the queue.`