**Part 82 - Generic queue collection class**

**Queue is a generic FIFO (First In First Out) collection class** that is present in System.Collections.Generic namespace. The Queue collection class is analogous to a queue at the ATM machine to withdraw money. The order in which people queue up, will be the order in which they will be able to get out of the queue and withdraw money from the ATM. The Queue collection class operates in a similar fashion. The first item to be added (enqueued) to the queue, will be the first item to be removed (dequeued) from the Queue.

o add items to the end of the queue, use **Enqueue**() method.  
  
To remove an item that is present at the beginning of the queue, use **Dequeue**() method.  
  
A foreach loop iterates thru the items in the queue, but will not remove them from the queue.  
  
To check if an item, exists in the queue, use **Contains**() method.  
  
**What is the difference between Dequeue() and Peek() methods?**  
**Dequeue**() method removes and returns the item at the beginning of the queue, where as **Peek**() returns the item at the beginning of the queue, without removing it.  
  
**Let us undestand the Queue collection class with an example.**  
public class Program  
{  
    public static void Main()  
    {  
        Customer customer1 = new Customer()  
        {  
            ID = 101,  
            Name = "Mark",  
            Gender = "Male"  
        };  
  
        Customer customer2 = new Customer()  
        {  
            ID = 102,  
            Name = "Pam",  
            Gender = "Female"  
        };  
  
        Customer customer3 = new Customer()  
        {  
            ID = 103,  
            Name = "John",  
            Gender = "Male"  
        };  
  
        Customer customer4 = new Customer()  
        {  
            ID = 104,  
            Name = "Ken",  
            Gender = "Male"  
        };  
  
        Customer customer5 = new Customer()  
        {  
            ID = 105,  
            Name = "Valarie",  
            Gender = "Female"  
        };  
  
        // Create a Queue  
        Queue<Customer> queueCustomers = new Queue<Customer>();  
        // To add an item to the queue, use Enqueue() method.

   // customer1 is added first, so this customer, will be the first to get out of the queue  
        queueCustomers.Enqueue(customer1);  
        // customer2 will be queued up next, so customer2 will be second to get out of the queue  
        queueCustomers.Enqueue(customer2);  
        queueCustomers.Enqueue(customer3);  
        queueCustomers.Enqueue(customer4);  
        queueCustomers.Enqueue(customer5);  
  
        // To retrieve an item from the queue, use Dequeue() method. Notice that the   
        // items are dequeued in the same order in which they were enqueued.  
        // Dequeue() method removes and returns the item at the beginning of the Queue.  
        Customer c1 = queueCustomers.Dequeue();  
        Console.WriteLine(c1.ID + " -  " + c1.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
  
        Customer c2 = queueCustomers.Dequeue();  
        Console.WriteLine(c2.ID + " -  " + c2.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
  
        Customer c3 = queueCustomers.Dequeue();  
        Console.WriteLine(c3.ID + " -  " + c3.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
  
        Customer c4 = queueCustomers.Dequeue();  
        Console.WriteLine(c4.ID + " -  " + c4.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
  
        Customer c5 = queueCustomers.Dequeue();  
        Console.WriteLine(c5.ID + " -  " + c5.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
        Console.WriteLine("-----------------------------------------------------------");  
  
        // After customer5 is dequeued, there will be no items left in the   
        // queue. So, let's enqueue the five objects once again  
        queueCustomers.Enqueue(customer1);  
        queueCustomers.Enqueue(customer2);  
        queueCustomers.Enqueue(customer3);  
        queueCustomers.Enqueue(customer4);  
        queueCustomers.Enqueue(customer5);  
  
        // If you need to loop thru items in the queue, foreach loop can be used in the   
        // same way as we use it with other collection classes. The foreach loop will  
        // only iterate thru items in the queue, but will not dequeue them.  
        foreach (Customer customer in queueCustomers)  
        {  
            Console.WriteLine(customer.ID + " -  " + customer.Name);  
            Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
        }  
        Console.WriteLine("-----------------------------------------------------------");  
  
        // To retrieve an item that is present at the beginning of the   
        // queue, without removing it use Peek() method.  
        Customer c = queueCustomers.Peek();  
        Console.WriteLine(c.ID + " -  " + c.Name);  
        Console.WriteLine("Items left in the Queue = " + queueCustomers.Count);  
        Console.WriteLine("-----------------------------------------------------------");  
  
        // To check if an item, exists in the queue, use Contains() method.  
        if (queueCustomers.Contains(customer1))  
        {  
            Console.WriteLine("customer1 is in Queue");  
        }  
        else  
        {  
            Console.WriteLine("customer1 is not in Queue");  
        }  
    }  
}  
  
public class Customer  
{  
    public int ID { get; set; }  
    public string Name { get; set; }  
    public string Gender { get; set; }  
}