**Software Development Data Structures H16Y 35**

**Assessment Task 3 Exercise 1**

**Assessment task instructions**

You are required to create two interfaces as per the requirements detailed.

You should develop the interfaces as C# Console Applications.

Include appropriate XML comments.

Your Interfaces should compile.

Copy and paste your code into the appropriate boxes.

Develop an interface for integer stack with the following methods:

**StackADT**

* void push ( int value ) //adds value to list appropriately for a stack
* int pop ( ) //remove and return value from list
* boolean isEmpty() //returns true if the stack is empty
* int size()//returns number of items in stack

|  |
| --- |
| Interace 1 Code |
| //Robert Lothian  //Software Development yr 2  //25.04.22  //This excercise is to create an interface for an Integer Stack  using System;  namespace Assessment3.\_1  {  internal class Program  {  interface StackADT  {  void push(int value);  int pop();  Boolean isEmpty();  int size();  }  class IntStack : StackADT  {  static readonly int max = 10;  int top;  int count;  int [] stack = new int[max];  /// <summary>This adds an element to the stack </summary>  /// <param name="value"></param>  public void push(int value)  {  Console.WriteLine("Item added");  value++;  count++;  }  /// <summary>This removes item from stack </summary>  /// <returns>Returns stack with last element \*\*top\*\* removed</returns>  public int pop()  {  int value = stack[top--];  count--;  Console.WriteLine("Item removed");  return value;  }    /// <summary>Determines if stack is empty </summary>  /// <returns>If top is less than or equal to 0 this will return TRUE </returns>  public Boolean isEmpty()  {  return (top<=0);  }  /// <summary>Calculates the size of the Stack </summary>  /// <returns>The count kept in pop and push methods</returns>  public int size()  {    Console.WriteLine("The number of items stacked is: " + count);  return count;  }  }  static void Main(string[] args)  {  IntStack myStack = new IntStack();  myStack.push(5);  myStack.push(1);  myStack.push(7);  myStack.pop();  myStack.size();  }  }  } |

Develop an interface for a Queue with the following methods:

**QueueADT**

* void enqueue ( int value ) //adds an element to the queue
* int dequeue() //return an element from the front of the queue
* boolean isEmpty() //check whether the queue is at maximum
* boolean ifFull() //check whether the queue is at maximum capacity

|  |
| --- |
| Interface 2 Code |
| //Robert Lothian  //Software Development yr 2  //25.04.22  //This excercise is to create an interface for a Queue  using System;  using System.Collections;  namespace Assessment3.\_1.\_2  {  internal class Program  {  /// <summary>  /// This interface determines the methods we will use for queue  /// </summary>  interface QueueADT  {  void enqueue(int value);  int dequeue();  Boolean isEmpty();  Boolean ifFull();  }  /// <summary>  /// This class creates a queue from elements determined in the interface  /// </summary>  class Queue : QueueADT  {  static readonly int max = 3;  int [] queue = new int[max];  int count = 0;  /// <summary>  /// This method is used to remove items from the queue, if queue has no element the isEmpty method is called  /// </summary>  /// <returns>The result of removing the first element and moving remaining element indexes down 1 place</returns>  public int dequeue()  {  int res = queue[0];  int i;  for (i = 0; i < count -1; i++)  {  queue[i] = queue[i + 1];  }  count--;  if (count <0)  {  isEmpty();  }  else  Console.WriteLine("Item removed from queue");    return res;  }  /// <summary>  /// This method is used to add an element to the queue, if queue is full if full class is called  /// </summary>  /// <param name="value"></param>  public void enqueue(int value)  {  if (count < max)  {  queue[count] = value;  count++;  Console.WriteLine("Item added to queue");  }  else  ifFull();  }  /// <summary>  /// This method is called if the queue is full and informs user with message  /// </summary>  /// <returns></returns>  public bool ifFull()  {  Console.WriteLine("Queue is full!");    return true;  }  /// <summary>  /// This method is called if queue is empty and informs user with message  /// </summary>  /// <returns></returns>  public bool isEmpty()  {  if (count <=0)  {  Console.WriteLine("Queue is empty!");  }  return true;  }  }  /// <summary>  /// Main class which creates an obect of type Queue and calls various methods within the queue class  /// </summary>  /// <param name="args"></param>  static void Main(string[] args)  {  Queue myQueue = new Queue();  myQueue.enqueue(1);  myQueue.dequeue();  myQueue.dequeue();  }  }  } |