

Assignment 4

Tianle Yuan

04/21/2023

04 Generics

Test your knowledge

1. Describe the problem generics address.

A: Generics allow you to use class/methods that should work with any data type.

2. How would you create a list of strings, using the generic List class?

A: `List<string> stringList = new List<string>();`

3. How many generic type parameters does the Dictionary class have?

A: 2. TKey and TValue.

4. True/False. When a generic class has multiple type parameters, they must all match.

A: False.

5. What method is used to add items to a List object?

A: `Add(T item)`.

6. Name two methods that cause items to be removed from a List.

A: `Remove(T item)` and `RemoveAt(int index)`.

7. How do you indicate that a class has a generic type parameter?

A: With `<T>` after the class name.

8. True/False. Generic classes can only have one generic type parameter.

A: False.

9. True/False. Generic type constraints limit what can be used for the generic type.

A: True.

10. True/False. Constraints let you use the methods of the thing you are constraining to.

A: True

Practice working with Generics

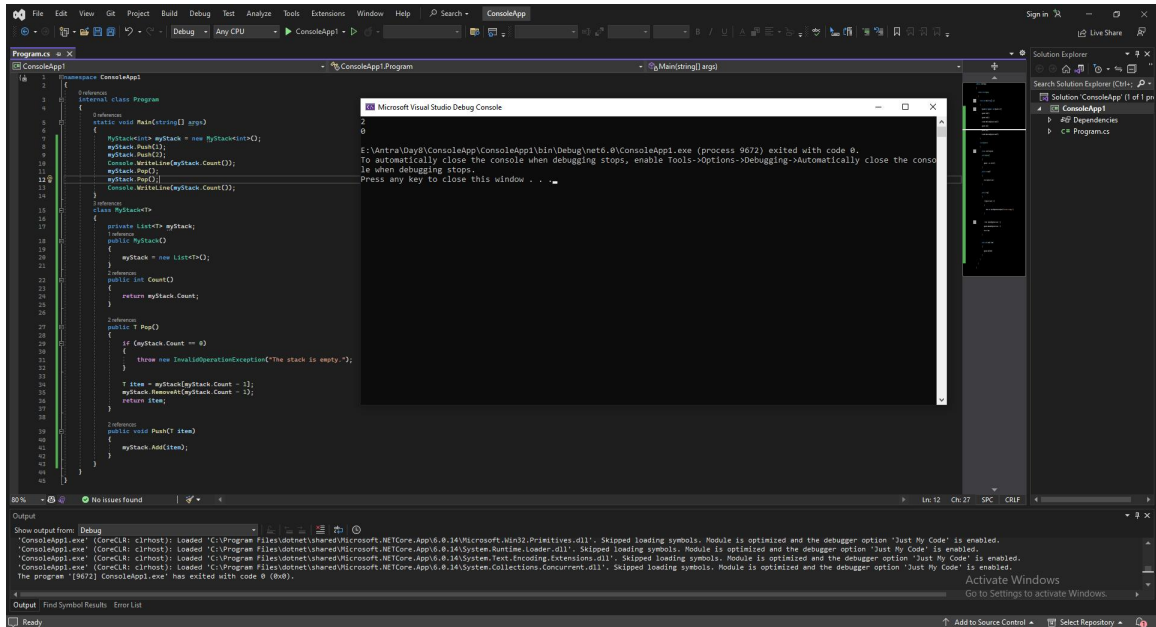
1. Create a custom Stack class `MyStack<T>` that can be used with any data type which has following methods

1. `int Count()`

2. `T Pop()`

3. `Void Push()`

A: Here below is my code

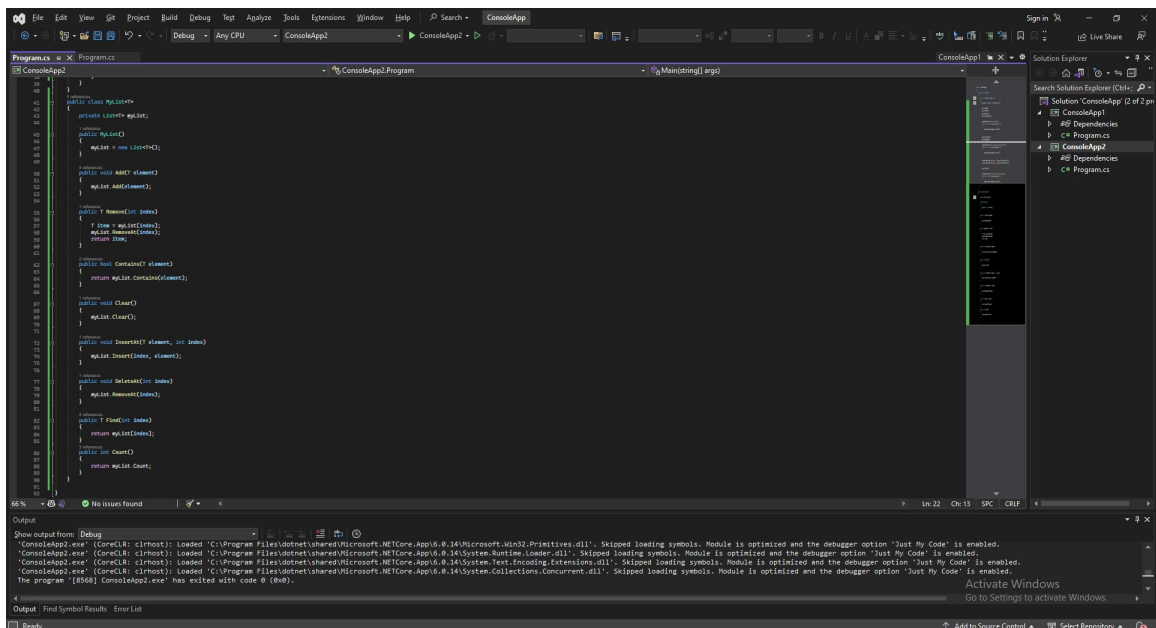


2. Create a Generic List data structure MyList<T> that can store any data type.

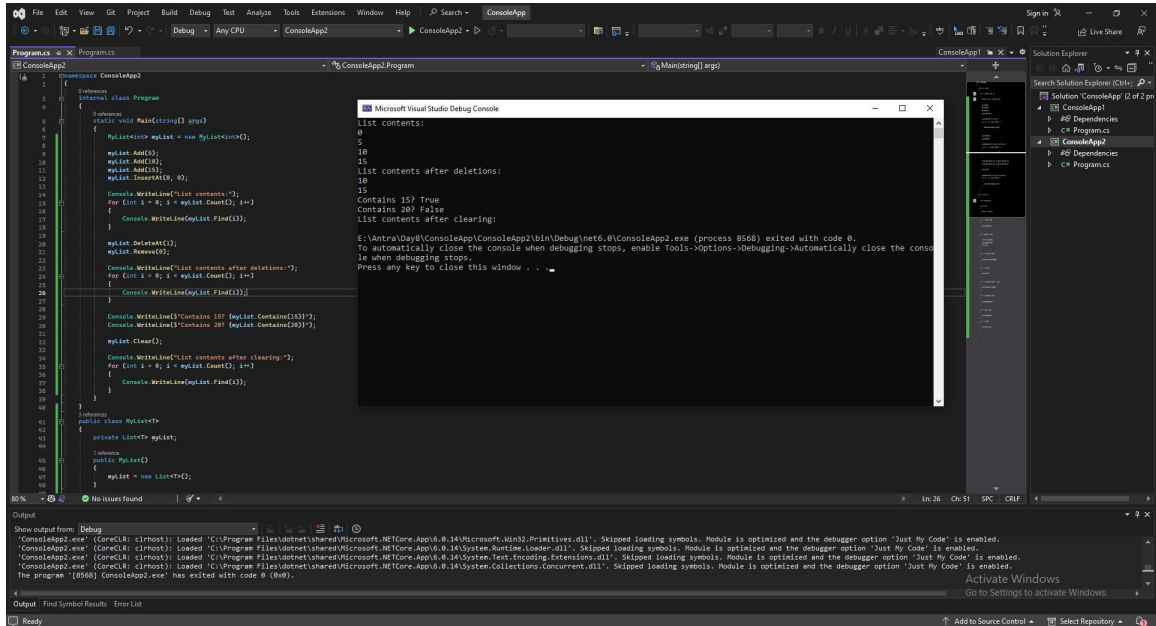
Implement the following methods.

1. void Add (T element)
2. T Remove (int index)
3. bool Contains (T element)
4. void Clear ()
5. void InsertAt (T element, int index)
6. void DeleteAt (int index)
7. T Find (int index)

A: Here below is the data structure I defined:



Here below is the code of usage of `MyList<T>`:



```
1  namespace ConsoleApp2
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              MyList<int> myList = new MyList<int>(5);
8
9              myList.Add(5);
10             myList.Add(10);
11             myList.Add(15);
12             myList.InsertAt(0, 0);
13
14             Console.WriteLine("List contents:");
15             for (int i = 0; i < myList.Count(); i++)
16             {
17                 Console.WriteLine(myList.Find(i));
18             }
19
20             myList.DeleteAt(0);
21             myList.Reverse();
22             Console.WriteLine("List contents after deletions:");
23             for (int i = 0; i < myList.Count(); i++)
24             {
25                 Console.WriteLine(myList.Find(i));
26             }
27
28             Console.WriteLine(myList.Find(1));
29
30             Console.WriteLine($"Contains 10? {myList.Contains(10)}?");
31             Console.WriteLine($"Contains 20? {myList.Contains(20)}?");
32
33             myList.Clear();
34             Console.WriteLine("List contents after clearing:");
35             for (int i = 0; i < myList.Count(); i++)
36             {
37                 Console.WriteLine(myList.Find(i));
38             }
39         }
40     }
41 }
42
43 public class MyList<T>
44 {
45     private List<T> myList;
46
47     internal MyList()
48     {
49         myList = new List<T>();
50     }
51 }
```

Microsoft Visual Studio Debug Console

```
List contents:
0
5
10
15
List contents after deletions:
0
10
15
Contains 10? True
Contains 20? False
List contents after clearing:

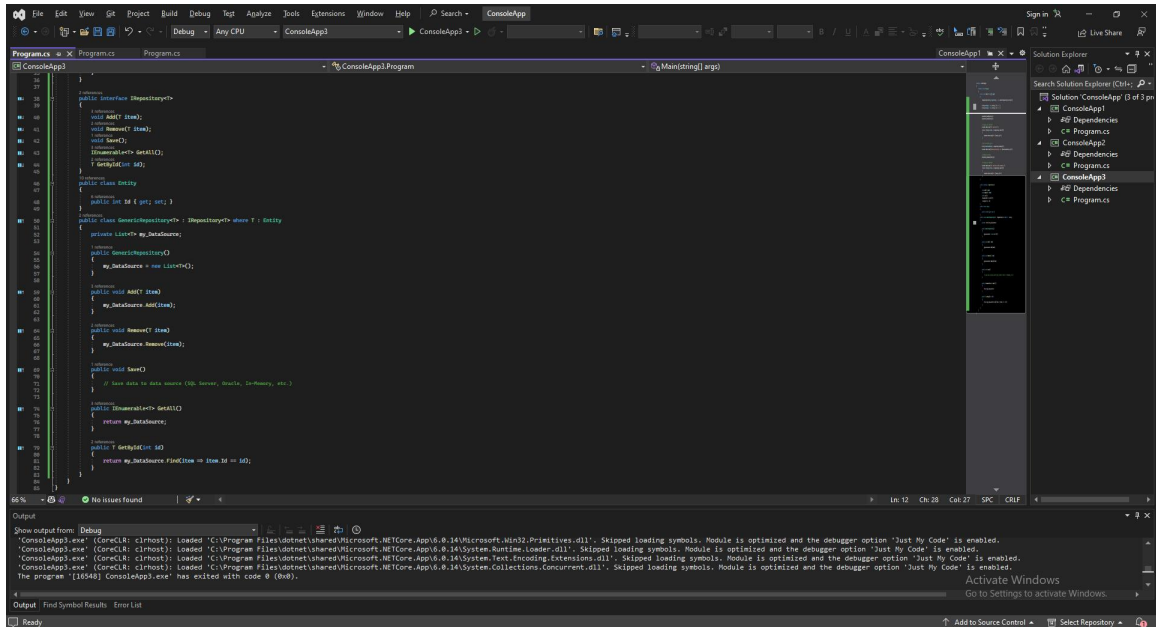
```

E:\Viveka\Days1\ConsoleApp2\ConsoleApp2\bin\Debug\net6.0\ConsoleApp2.exe (process 8568) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
le when debugging stops.
Press any key to close this window . . .

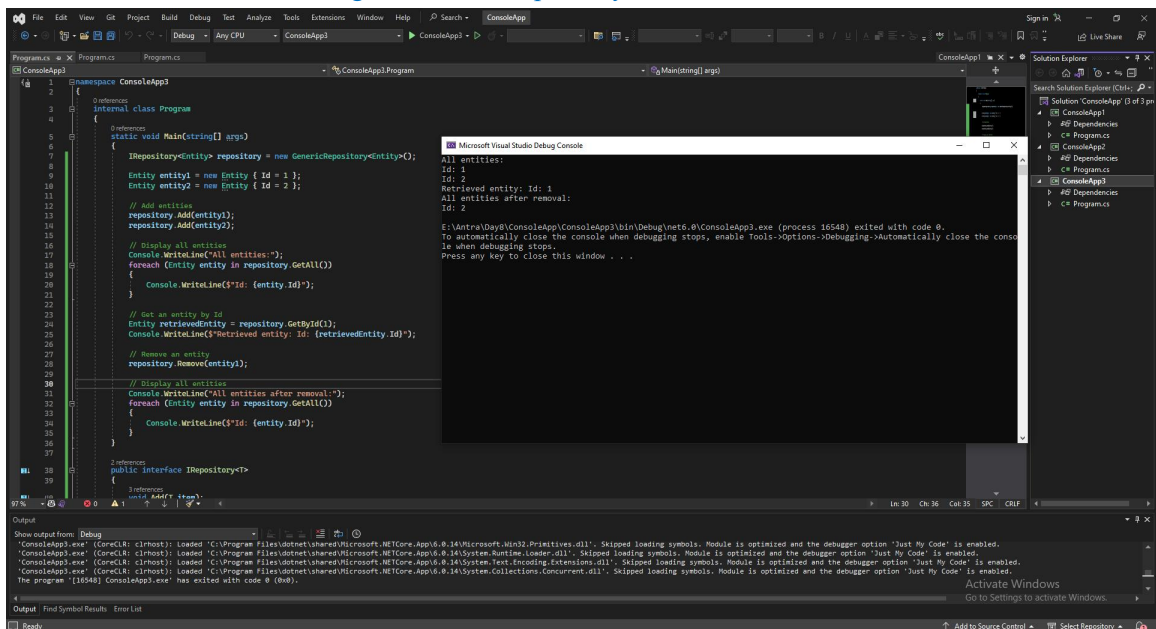
3. Implement a `GenericRepository<T>` class that implements `IRepository<T>` interface that will have common /CRUD/ operations so that it can work with any data source such as SQL Server, Oracle, In-Memory Data etc. Make sure you have a type constraint on `T` where it should be of reference type and can be of type Entity which has one property called `Id`. `IRepository<T>` should have following methods

- 1. void Add(`T` item)**
- 2. void Remove(`T` item)**
- 3. Void Save()**
- 4. `IEnumerable<T>` GetAll()**
- 5. `T` GetById(int id)**

A: Here below is the data structure I defined:



Here below is the code of usage of GenericRepository<T>:



Explore following topics

- Generics in .NET
- Generic classes and methods
- Collections and Data Structures
- Commonly Used Collection Types