# Lab: Generics

Problems for the [["C# OOP Advanced" course @ Software University](https://softuni.bg/courses/csharp-oop-advanced-high-quality-code)](https://softuni.bg/courses/java-oop-advanced).

You can check your solutions in [Judge](https://judge.softuni.bg/Contests/706/Generics-Lab).

# Part I: Generics

## Box of T

**NOTE**: You need a public StartUp class with the namespace BoxOfT.

Create a class **Box<>** that can store anything.

It should have two public methods:

* void Add(element)
* element Remove()
* int Count { get; }

Adding should add on top of its contents. Remove should get the topmost element.

### Examples

|  |
| --- |
| public static void Main(string[] args)  {  Box<int> box = new Box<int>();  box.Add(1);  box.Add(2);  box.Add(3);  Console.WriteLine(box.Remove());  box.Add(4);  box.Add(5);  Console.WriteLine(box.Remove());  } |

### Hints

Use the syntax Box<T> to create a generic class

## Generic Array Creator

**NOTE**: You need a public StartUp class with the namespace GenericArrayCreator.

Create a class ArrayCreator with a method and a single overload to it:

* static T[] Create(int length, T item)

The method should return an array with the given length and every element should be set to the given default item.

### Examples

|  |
| --- |
| static void Main(string[] args)  {  string[] strings = ArrayCreator.Create(5, "Pesho");  int[] integers = ArrayCreator.Create(10, 33);  } |

# Part II: Generic Constraints

## Generic Scale

**NOTE**: You need a public StartUp class with the namespace GenericScale.

Create a class **Scale<T>** that holds two elements - left and right. The scale should receive the elements through its single constructor:

* Scale(T left, T right)

The scale should have a single method:

* T GetHeavier()

The greater of the two elements is the heavier. The method should return null if the elements are equal.