# **GenericsExercise01**

In this exercise you should learn about using generic to make an abstract data structure which is type strong. You will make your own simple list class for general purpose.

The generic class should be named **MyList** and must contain:

**Private fields:**

* a generic array named **myArray**
* a number **count** that hold the number of element in use (and is one more than the last index in use)
* Example on content of **myArray** with 6 elements in use (**count** == 6)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| value | Bjørk | Tove | Lene | Simon | Hans | Allan | ? | ? |

Elements will always be by index between 0 and **count**-1

**Public constructors:**

* default constructor:  
  postcondition:  
  **- myArray** contains an array with space for 100 elements of the generic type  
  - **count** is zero
* constructor with a number **size** as argument  
  precondition: **size** > 0  
  postcontition:   
  **- myArray** contains an array with space for a number of elements of the generic type given by **size  
  - count** is zero

**Public property:**

* **Count:** only public get witch return the number of elements in use   
  postcondition: nothing changed in the class

**Public methods:**

* **Add:** takes one generic element (**elm**) as input parameter   
  precondition: **elm** != null and there is at least space for one more element (**count** < **size** of **myArray**)

postcondition: **elm** is inserted after the element with highest index and **count** is adjusted by +1, nothing else is changed in the class

* **GetElementByIndex**: take a number (**idx**)as parameter an returns the generic element at the index given by **idx**  
  precondition: there is at least one element (0 < **count)**, and **idx** is a legal place (0 <= **idx** < **count)**

postcondition: element (reference) is returned but nothing changed in the class

**GetLastElement**: returns the generic element (reference) at the last index in use  
precondition: there is at least one element (0 < **count**)  
postcondition: last element (reference) is returned but nothing changed in the class

**GetFirstElement**: returns the generic element (reference) at the first index in use (0)   
precondition: there is at least one element (0 < **count**)   
postcondition: first element (reference) is returned but nothing changed in the class

* **RemoveLastElement**: remove the element with highest index (returns nothing)  
  precondition: there is at least one element (0 < **count**)  
  postcondition: **count** is adjusted by -1
* **RemoveFirstElement**: remove the element with lowest index (returns nothing)  
  precondition: there is at least one element (0 < **count**)  
  postcondition: all elements from index 1 is moved to a place one index lower and **count** is adjusted by -1
* **InsertElementAt**: takes an index (**idx**) and a generic element (**elm**) as parameter, inserts the element at the index   
  precondition: **elm** != null, there is at least space for one more element (**count** < **size** of **myArray**), idx is a legal place (0 <= **idx** < **count**)

postcondition: **count** is adjusted by +1, **myArray** contains all old elements from before and also the new one in the following order: all elements with an old index < **idx** are still at same index, new element (**elm**) is at index **idx**, all elements with an old index >= **idx** are moved to a place one index higher.

**More advanced (if time):**

* **Implement an Indexer**: that can be used for replace and retrieving an element using the syntax like for an array and a list.

Example: Student s = studentList[4]; studentList [5] = new Student(………);  
The indexer should take a number **idx** as parameter.  
precondition: **idx** is a legal element index (0 <= **idx** < **count**)  
- used on the left side in an assignment: right hand side must not be null  
postcondition:   
- used on the right side in an assignment: element at **idx** (reference) is returned but nothing changed in the class.

- used on the left side in an assignment: element at **idx** (reference) is replaced by the right hand side element, but nothing else is changed in the class.

* **Implement an Iterator**: that can be used for iterate through a MyList object by using a foreach loop.