# Exercises: Generics

This document defines the exercises for [“Java OOP Advanced” course @ Software University](https://softuni.bg/modules/59/java-advanced). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/1527)

## Generic Box

Create a **generic class Box** that can store any type. **Override** the **toString()** method to print the type and the value of the stored data in the format "**{class full name}: {value}**"**.**

Use the class that you've created and test it with the class **java.lang.String**. On the first line, you will get **n** - the number of strings to read from the console. On the next **n** lines, you will get the actual strings. For each of them create a box and call its **toString()** method to print its data on the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  life in a box  box in a life | java.lang.String: life in a box  java.lang.String: box in a life |
| 1  I am a programmer | java.lang.String: I am a programmer |

## Generic Box of Integer

Use the description of the previous problem but now, test your generic box with **Integers**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  7  123  42 | java.lang.Integer: 7  java.lang.Integer: 123  java.lang.Integer: 42 |
| 5  12  13  14  15  16 | java.lang.Integer: 12  java.lang.Integer: 13  java.lang.Integer: 14  java.lang.Integer: 15  java.lang.Integer: 16 |

## Generic Swap Method Strings

Create a generic method that receives a list containing **any type of data** and swaps the elements at two given indexes.

As in the previous problems, read **n** number of boxes of type **String** and add them to the list. On the next line, however, you will receive a **swap** command consisting of **two indexes**. Use the method you've created to swap the elements that correspond to the given indexes and **print** **each** element in the list.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Peter  George  Swap me with Peter  0 2 | java.lang.String: Swap me with Peter  java.lang.String: George  java.lang.String: Peter |
| 2  Uni  Soft  0 1 | java.lang.String: Soft  java.lang.String: Uni |

## Generic Swap Method Integers

Use the description of the previous problem but now, test your list of generic boxes with **Integers**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  7  123  42  0 2 | java.lang.Integer: 42  java.lang.Integer: 123  java.lang.Integer: 7 |
| 5  12  13  14  15  16  3 4 | java.lang.Integer: 12  java.lang.Integer: 13  java.lang.Integer: 14  java.lang.Integer: 16  java.lang.Integer: 15 |

## Generic Count Method Strings

Create a **method** that receives as an argument a **list of any type that can be compared** and an **element of the given type**. The method should **return the count of elements that are greater than the value of the given element**. **Modify your Box class** to support **comparing by the value** of the data stored.

On the first line, you will receive **n** - the number of elements to add to the list. On the next **n** lines, you will receive the actual elements. On the last line, you will get the value of the element to which you need to compare every element in the list.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 3  aa  aaa  bb  aa | 2 | 6  a  b  c  d  e  f g | 0 |

## Generic Count Method Doubles

Use the description of the previous problem but now, test your list of generic boxes with **Doubles**.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 3  7.13  123.22  42.78  7.55 | 2 | 1  1231542.123  1 | 1 |

## Custom List

Create a generic data structure that can store **any type** that can be **compared.** Implement functions:

* **void add(T element)**
* **T remove(int index)**
* **boolean contains(T element)**
* **void swap(int index, int index)**
* **int countGreaterThan(T element)**
* **T getMax()**
* **T getMin()**

Create a command interpreter that reads commands and modifies the custom list that you have created. Implement the commands:

* **Add {element}** - Adds the given element to the end of the list.
* **Remove {index}** - Removes the element at the given index.
* **Contains {element}** - Prints if the list contains the given element (**true** or **false**)**.**
* **Swap {index1} {index2}** - Swaps the elements at the given indexes.
* **Greater {element}** - Counts the elements that are greater than the given element and prints their count
* **Max** - Prints the maximum element in the list.
* **Min** - Prints the minimum element in the list.
* **Print** - Prints all elements in the list, each on a separate line.
* **END** - stops the reading of commands.

**Note**: For the **Judge tests,** use **String** as **T.**

**Examples**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| Add aa  Add bb  Add cc  Max  Min  Greater aa  Swap 0 2  Contains aa  Print  END | cc  aa  2  true  cc  bb  aa | Add Peter  Add George  Swap 0 0  Swap 1 1  Swap 0 1  Swap 1 0  Swap 0 1  Print  END | George Peter |

## Custom List Sorter

Extend the previous problem by creating an additional **Sorter class**. It should have a single static **method** **sort()** which can sort objects of type **CustomList** containing any type that can be compared. **Extend the command list** to support one additional command **Sort**:

* **Sort** - Sort the elements in the list in ascending order.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| Add cc  Add bb  Add aa  Sort  Print  END | aa  bb  cc | Add Peter  Add George  Max  Sort  Print  END | Peter  George  Peter |

## \*Custom List Iterator

For the print command, you have probably used a **for** a loop. Extend your custom list class by making it implement **Iterable.** This should allow you to iterate your list in a **foreach** statement.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| Add aa  Add bb  Add cc  Max  Min  Greater aa  Swap 0 2  Contains aa  Print  END | cc  aa  2  true  cc  bb  aa | Add Peter  Add George  Swap 0 0  Swap 1 1  Swap 0 1  Swap 1 0  Swap 0 1  Print  END | George  Peter |