# Exercises: Unit Testing

This document defines the exercises for ["Java OOP Advanced" course @ Software University](https://softuni.bg/java-basics-oop).

## Database

Create a simple class - **Database**. It should **store Integers**. You should set the initial integers by constructor. Store them **in array**. Your Database should have a functionality to **add**, **remove** and **fetch all stored items**. Your task is to **test the class**. In other words, create the class and **write tests**, so you are sure its methods are working as intended.

### Constraints

* Storing array's **capacity** must be **exactly 16 integers**.
  + If the size of the array is not 16 integers long, throw **OperationNotSupportedException**.
* **Add** operation, should **add an element at the next free cell.** (just like a stack)
  + If passed element is null throw **OperationNotSupportedException**.
  + If there are already 16 elements and add is called, throw **OperationNotSupportedException.**
* **Remove** operation, should support only removing an element **at the last index**. (just like a stack)
  + If you try to remove element from empty Database throw **OperationNotSupportedException**
* **Constructors** should take integers only, and store them in **array.**
* **Fetch method** should return the elements as **array.**

### Hint

Do not forget to **test the constructor(s)**. They are methods too!

## Extended Database

You already have a class - **Database**. Now your task is to extend it. It should support, adding, removing and finding People. In other words, it should store **People**. There should be two types of finding methods - first: **findById** (long id) and the second one: **findByUsername** (String username). As you may guess, each person should have its own unique id, and unique username. Your task is to implement these functions and test them.

### Constraints

Database should have methods:

* **Add**
  + If there are multiple users with this id, throw **OperationNotSupportedException.**
  + If negative nor null ids are present, throw **OperationNotSupportedException**.
  + If there are already 16 elements and add is called, throw **OperationNotSupportedException.**
* **Remove**
  + If no elements are present when remove is called, throw **OperationNotSupportedException**.
* **findByUsername**
  + If no user is present by this username, throw **OperationNotSupportedException**.
  + If username parameter is null, throw **OperationNotSupportedException.**
  + Arguments are all CaseSensitive!
* **findById**
  + If no user is present by this id, throw **OperationNotSupportedException**.

### Hint

Do not forget to test the constructor(s). They are methods too!

## Iterator Test

Create a class "**ListIterator**", it should receive the collection (Strings) which it will iterate, through its constructor. You should store the elements in a List and get them initially through its constructor. If there is null passed to the constructor, throw new **OperationNotSupportedException**. The class should have three main functions:

* **Move** - should move an internal index position to the next index in the list, the method should return true if it successfully moved and false if there is no next index.
* **HasNext** - should return true if there is a next index and false if the index is already at the last element of the list.
* **Print** - should print the element at the current internal index, calling Print on a collection without elements should throw an appropriate exception with the message "**Invalid Operation!**".

By default, the internal index should be pointing to the **0th index** of the List. Your program should support the following commands:

|  |  |  |
| --- | --- | --- |
| **Command** | **Return Type** | **Description** |
| Move | boolean | This command should move the internal index to the next index. |
| Print | void | This command should return the element at the current internal index. |
| HasNext | boolean | Returns whether the collection has a next element. |

### Test

Create tests, so you are sure all methods in the class ListIterator are working as intended.

### Constraints

* There will always be only **1** **Create** command and it will always be the first command passed.
* The last command will always be the only **END** command.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Create  Print  END | Invalid Operation! |
| Create Stefcho Goshky  HasNext  Print  Move  Print  END | true  Stefcho  true  Goshky |
| Create 1 2 3  HasNext  Move  HasNext  HasNext  Move  HasNext  END | true  true  true  true  true  false |