Exercise 01: Stack and palindrome

In this exercise, we will be implementing the *Stack* data structure (see `CharStack` interface), and then using it to check strings for palindrome properties.

Preparation

1. Create a new folder `ex01` in your local repository in both `src` and `test` folder or as a separate folder (if you have a separate folder for every exercise)
2. Get the new Exercise from our Repository in any local directory by following one of the next steps:
   a. `git clone` *`LINK SEE LEARNING CAMPUS`*
   b. Go to our GitLab repository and download the project under *`Code`* as a zip file and unpack it.
3. Copy the class files from the `src` and `test` folder or the whole code (if you have a separate folder for every exercise) to the newly created folders in your local repository.

This is how your folder structure could look like:

Option 1

```
Exercise01
   src
   test
   …

Exercise02
   src
   test
   …
```

Option 2

```
src
   ex01
   ex02
   …

test
   ex01
   ex02
   …
```

Task 1: Stack Datastructure (using char-Array)

1. Complete the `CharStackImpl` class by implementing the `push`, `pop` and `size` methods
   - `push` places an element on top of the stack,
     `pop` removes the top element from the stack;
     The stack data structure is thus called LIFO -- *last in - first out*.
   - Also, remember to implement a helper class `CharElement` to model an Element of the stack.
   - Verify that the `CharStackTest` test runs without errors.
2. Add the modified `CharStackImpl.java` file as well as the helper class to your commit, in IntelliJ with right click -> Git -> Add, or in the console with `git add CharStackImpl.java`
3. Commit and push the changes

   - IntelliJ: VCS -> Commit Changes -> Commit and push
   - Terminal:
     ```
     git commit -m "Your commit message"
     git push
     ```

Task 2: Palindrome

1. Implement the static method `Palindrome.isPalindrome()` in which you now use your Stack to test arbitrary strings for palindrome properties.

- A string is a palindrome if it has the same sequence of letters when read both forwards and backwards, i.e. the text is "mirrored". The capitalization of characters should be ignored.
  How can the stack be used to check this?
- The `String.replaceAll` method can be used to remove all spaces.
- The `String.toLowerCase` (or `String.toUpperCase`) method converts all characters to lower case or upper case respectively.
- The `String.toCharArray` method returns the string as an array of `chars`.
- Verify that the `testPalindrome()` test runs without errors.

2. Commit and push your changes.

Please note: the upcoming tasks will always use Git in the same way. You should get used to this workflow.