DTU Compute
Department of Applied Mathematics and Computer Science
Advanced Programming (02324), Spring 2025
Carlos E. Budde and Ekkart Kindler

Assignment 2a: Generic Stacks

Assignment 2 will be concerned with defining and implementing the stack from Assignment 1 in a more elegant way. The stack definition and implementations from Assignment 1 (Tuesday, February 3) had two major design flaws:

a. The stack from Assignment 1 was working for elements of type **`Integer`** only. If we wanted to use it with elements of other types (e.g. for **`String`**), we would need to copy the code and change some of the types, which would result in lots of redundant code.
b. The methods from Assignment 1 did not properly deal with exceptional situations (for example when the **`pop()`** or **`top()`** operation cannot return a proper value since the stack is empty. To improve this, the stack needs to define possible exceptions, and the implementations need, in the relevant situations, throw exceptions, and the rest of the application needs to properly catch them and deal with them.

In this assignment (Assignment 2a: Friday, February 7), you will implement a generic version of the stack, which can deal with any kind of elements. In Assignment 2b (Tuesday, February 11), you will implement the exceptions. For both assignments, you can start from code which is provided as an IntelliJ/Maven project **`assignment2.zip`** on DTU Learn (and you can of course use some of the code from your Assignment 1).

Note that the interface **`Stack`**, the (empty) implementation classes, the **`IntegerStackApp`** and **`IntegerStackGUI`** and the test are adapted and slightly expanded for generics and showing exceptions already.

In the following, Assignment 2a will be broken down into smaller steps; more details on Assignment 2b will be provided in separate document on Tuesday, February 11.

1. Before you start with implementing something for Assignment 2, make sure that your code for Assignment 1 is working properly (and, if you did not do it yet, show it to the TAs or teachers).
2. Download the project (**`assignment2.zip`**) from DTU Learn to your computer. Unpack this zip-file to some new sub-folder on your computer (at best in a common folder for all IntelliJ projects for this course) and import it to your IntelliJ (via the pom file as discussed in Assignment 1).
3. Implement the classes **`LinkedListStack<E>`** and **`ArrayStack<E>`** in a generic way and make sure that all test (there are now automatic tests for Integers and Strings, but the GUI is still using Integers only). You can and should reuse code from your solution of Assignment 1 for that purpose.
4. Run all the tests provided with **`assignment2.zip`** and see whether all tests run successfully. Some tests, actually, check whether your implementation throws the expected exceptions in the relevant situations; these tests do not yet need to run successfully. Also manually test the different stack implementations with the **`IntegerStackApp`** – which as discussed uses integers only (but since the implementation is generic, this should be fine).

5. In the end, reflect on what the advantage of using generic interfaces and generic classes are.
6. The solution should be shown groupwise to the teachers or teaching assistants during the tutorial sessions on Tuesdays or Fridays at latest on Tuesday, February 11). The code of this task will to some extend be used during the next tutorials, which you later will need to be submit (in week 3) together with the other tasks (details to be announced).
7. If you should have time left, you can already start with implementing throwing the relevant exceptions in relevant situations in your implementations (see the throws declarations in the code) and also try to handle them in the client (GUI) when they occur.

References and further reading

[02324 f25 L02.1] Carlos E. Budde and Ekkart Kindler: Lecture Notes for the course Advanced programming. Spring 2025. (DTU Learn).

[Eck 2022] David J. Eck: Introduction to Programming Using Java. Version 9.0, JavaFX Edition, May, 2022. Section 10.5