Programming Basics
Exercise – WS2020/21

Exercise 07 – Stack and Queue

For the following exercises you will need an environment to program in Java. You will need the current Java Development Kit and a text editor. For better usability we recommend the use of an IDE for example "Eclipse" or "IntelliJ IDEA".

These tasks require you to make use of the data structures Stack and Queue. You can use both arrays or linked lists to build them and you may also use code from the lecture. Your Stack and Queue should implement at least the following methods:
push(data**)** for Stack, resp. **offer(**data**)** for Queues to add a new element,
pop() resp. **poll()** to remove an element,
peek() to return the first element without removing it,
isEmpty() to check whether there are any elements in your stack/queue.

Other helpful methods:
size() returns the current size of your queue or stack.

Task 1 – Well-formed Brackets

Use a Stack to determine whether an expression made of brackets **( )** and **[ ]** is well-formed.
Write a method that gets a **String made of brackets** as an argument and returns 'true' if the String is well-formed and 'false' if it is not.

Examples for <u>well-formed</u> expressions:
([])
(([][]))
(([()]))

Examples for <u>malformed</u> expressions:
(])[
[(])
)(
(()
()[))))))]

Task 2 – Sort a Queue

Create an Integer Queue that is filled with arbitrary numbers (you can generate them randomly or add them manually).
Create a method that uses this Queue and one additional Stack to sort the numbers in the Queue. At the end of the method, the sorted numbers should be contained in the Stack, with the smallest number on top of the Stack.

Task 3 – Supermarket Simulation

Create a class **Supermarket** which as an attribute has several **cashDesks** that are represented by Queues. The **cashDesks** are used by **Customers** who each have a number of **items** in their cart.

A new Customer arrives every 30 seconds and will always go to the cashDesk with the shortest queue. On the cashDesk, the cashier will need 1 second to handle an item.
Write a method that **simulates the workflow** at the cashDesks for any number of seconds that you want.

Play around with numbers. See how the workflow changes when you change the amount of available cashDesks or the amount of items a customer can fit into their cart.