ASE Exercise 4 (Fall 2021)

Task 1

Completing the following task requires to import some existing files first.

1. Go to the RWTHmoodle learning room and download *Lecture4_src*. Extract .zip file to any convenient location.
2. Start your eclipse and copy the extracted folders to a newly created project. (`lib`, `warehouse_files` and `lego_files`)
3. Don't forget to add the additional libraries (jar files in lib-folder) to your Java Build Path).
4. Implement the ERP system presented in lecture 4.
5. The project contains a `warehouse_stock.csv` file containing Part ids and their respective quantities. You need to modify the project to serve the following tasks:
    a) The warehouse currently does not contain a way to look up a part based on its id. Implement a method that enables this functionality.
    b) Overload the `isAvailable` method to use the `part_id` instead of a `Part`.
    c) Populate the warehouse from the `warehouse_stock.csv` file and print all the parts that have a stock of less than 20 pcs.
    d) How many AAT (75080-1) or Tie Fighter (75095-1) can be built from the available stock of parts?
    e) How many Sith Infiltrator (75096-1) can be built after building the 2 x Slave 1 (75060-1)?
    f) Is it possible to produce 11 x TIE Advanced Prototype (75085-1) and 2 x Battel Droid Troop Carrier (75086-1)? If not, why?

For the following task, create a new project `Exercise4` in your Eclipse IDE

Task 2

Implement the method `public String missingChar(String str, int n)` Use existing methods of string to solve the problem.

Given a non-empty string str and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0 to str.length()-1 inclusive).

Test set

```
missingChar("kitten", 1) → "ktten"
missingChar("kitten", 0) → "itten"
missingChar("kitten", 4) → "kittn"
```

Task 3

Implement the method `public boolean array123(int[] nums)`

Given an array of ints, return true if the sequence 1, 2, 3 appears in the array somewhere.

Test set

```
array123({1, 1, 2, 3, 1}) → true
array123({1, 1, 2, 4, 1, 3}) → false
array123({1, 1, 2, 1, 2, 3}) → true
```

Task 4

Implement the method `public boolean arraywave(int[] nums)`.
Given an array of ints, return true if each number is either bigger or smaller than its neighbors. Only arrays that form a wave pattern should qualify. For the numbers at the start and end, only consider a single neighbor.

Test Set:

```
arraywave ({1, 3, 2, 5, 1}) → true
arraywave ({1, 7, 2, 4, 5}) → false
arraywave ({1, 4, 2, 1, 9, 8}) → false
```

Task 5

Implement the method `public boolean noDuplicates(int[] nums)`.
Given an array of ints, return true if all the numbers are unique. Return false otherwise.

Test set

```
noDuplicates({1, 9, 2, 3, 8}) → true
noDuplicates({1, 1, 2, 2, 2, 1}) → false
noDuplicates({1, 8, 6, 1, 3}) → false
```