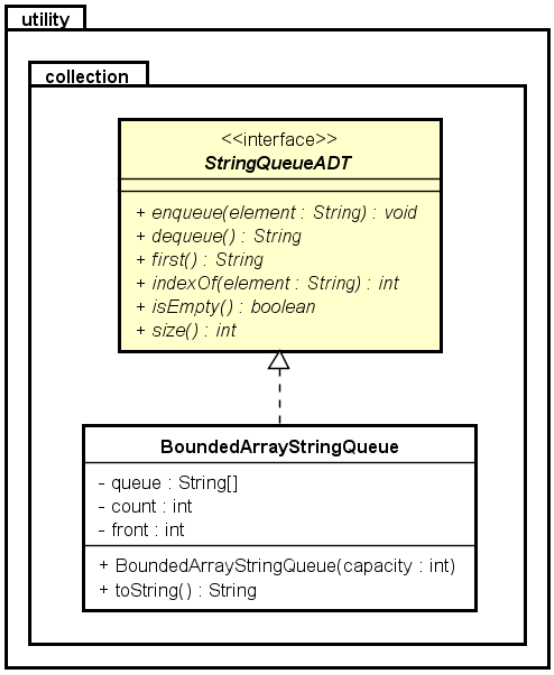
Exercises Session 1 SDJ2

Exercise 01.01

Implement in Java what you can see in the class diagram below. The diagram represents the abstract data type StringQueue implemented with an array as data structure. The class is called `BoundedArrayStringQueue`, is in package `utility.collection` and the array cannot resize.



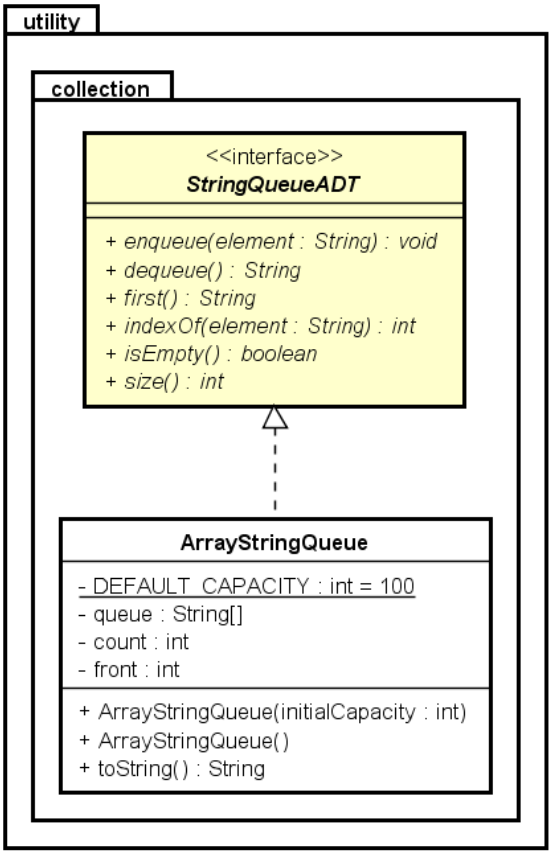
There are a few things to mention:

1. Both the class and the interface are in a package called `utility.collection`
2. You have to use an array as one of the instance variable (`queue`) but you are allowed to use other instance variable instead of the two integer variables – but do not use dependent variables.
3. Methods `enqueue`, `dequeue` and `first` could throw exceptions (see javadoc documentation)
4. Make the solution as efficient as possible such that `indexOf` is the only method having loops.
5. Method `toString` return a string with all elements separated with semicolons and encapsulated in a set of curly braces, example: `"{A, B, C}"`

Test your solution such that you are convinced that your implementation of `BoundedArrayQueue` is correct.

Exercise 01.02

Implement in Java what you can see in the class diagram below. The diagram represents the abstract data type StringQueue implemented with an array as data structure. This time with an array that can "resize" when full.



Most of it is identical to your implementation of class `BoundedArrayStringQueue` from the first exercise. There are a few things to mention:

1. In class `StringArrayQueue` there is a class variable (static field) with the initial capacity for the queue to be used in the no-args constructor.
2. Methods `enqueue`, `dequeue` and `first` could throw exceptions (see javadoc documentation). However, `enqueue` cannot throw an `IllegalStateException` for a full queue. Instead, the queue is resized calling `expandCapacity()`.
3. Method `expandCapacity()` double the size of the array

Test your solution for `ArrayStringQueue`