Prof. Dr.-Ing. Ralf Schenkel
Martin Blum
Trier University

Wintersemester 2024/2025

Exercises for the Class

Elements of Computer Science: Programming

Assignment 10

Submission of solutions until 23.01.2025 at 10:00
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way in order to get a point!
- Please comment your solutions, so that we can easily understand your ideas!
- If you have questions about programming or the assignments, just ask you teachers!
- **Submission that can't be compiled are graded with 0 points!**

Exercise 1 (Evaluation: predefined `main` method)

See the class `Queue` from example `K5B05E_Queue_Linked` of the lecture. Extend `Queue` with the following functionalities:

- **`public void`** `append(`**`int`**`[] newData)`
  This function appends all values from the `newData` field to the queue.
- **`public int`** `size()`
  This function outputs how many elements are currently stored in the queue.
- **`public int`** `elementAt(`**`int`** `position)`
  This function outputs the value at the position `position` of the queue (index 0 represents the first element). If `position` is negative or the queue contains too few elements, -1 is returned.
- **`public int`** `contains(`**`int`** `m)`
  This function is to determine whether the value `m` is contained in the queue. If it is included, its position in the queue is returned, if not -1.
- **`public static`** `Queue concat(Queue q, Queue p)`
  This static function should concatenate the two queues, so that first the elements from `q` and then the elements from `p` are contained in the resulting queue. Both `q` and `p` must remain unchanged.
- **`public int`**`[] toArray()`
  This function is intended to create an array containing all elements of the queue. The queue itself should be empty afterwards.

`QueueTest` may be changed for your own tests, but for the grading a new version of `QueueTest` will be used, which will include more extensive tests. Therefore your program must work with the original version of `QueueTest` as well!

Exercise 2 (Evaluation: predefined `main` method)

Take another look at the class `Queue`, but now from the example `K5B05E_Queue_Array` of the lecture. Extend `Queue` with the same functionalities as in the previous task:

```
public void append(int[] newData)
public int size()
public int elementAt(int position)
public int contains(int m)
public static Queue concat(Queue q, Queue p)
public int[] toArray()
```

However, you should also change the implementation of the queue so that queues can never become full when elements are added:

- In the constructor, the value 4 is to be used as the initial size of the array used in the queue.
- A dynamically growing array should be used for storage (similar to `StringBuffer`): If the array of a queue is full (usually by `enQueue()` invocations), a new array of double size is created, into which all data is transferred.
- Similarly, if the queue is shrinking, the array is to be replaced by a new array with half the capacity: If an element is removed and the number of elements in the queue is then less than or equal to a quarter of the current capacity, the size of the array in the queue is to be halved. However, the capacity should not be less than value 4.
- In order to trigger these capacity changes manually, two methods

  ```
  public void setCapacity(int n)
  public int getCapacity()
  ```

  are to be implemented which can be used to manually adjust or query the current capacity. A call `setCapacity(n)` should be ignored if $n < 4$ or $n \leq$ `size()` is true.

The note from the previous task for the files `Queue.java` and `QueueTest.java` also applies here.