Prof. Dr.-Ing. Ralf Schenkel
Tobias Zeimetz
Trier University

Wintersemester 21/22

Exercises for the Class

Elements of Computer Science: Programming

Assignment 09

Submission of solutions until 3:00 p.m. at 12.01.2022
at `moodle.uni-trier.de`

- Every task needs to be edited in a meaningful way!
- Please comment your solutions, so that we can easy understand your ideas!
- If you have questions about programming or the homeworks, just ask you teachers!
- **Submission that can't be compiled are rated 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 is to output how many elements are currently in the queue.
- `public int elementAt(int position)`
  This function is to output the value at the position `position` of the queue (it is to count as in a field starting from index 0). 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` should remain unchanged.

- **public int**[] toArray()
  This function is intended to create an array containing all entries of the queue. The queue should then be empty.

QueueTest may be changed for your own tests, but for the correction a new version of QueueTest will be used, which will make 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 queues 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 field used in the queue.
- A dynamically growing field should be used for storage (similar to StringBuffer): If the field of a queue is full (usually by enQueue), a new field of double size is created, into which all data is transferred.
- Similarly, if the queue is shrinking, the field is to be replaced by a new field 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 field in the queue is to be halved. However, the size 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 valid.

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