

# Exam – extra part

## Explanations

This part of the exam enables a grade higher than E

In addition to the required part, the student can also do this part of the exam. This part is only considered when the required part has been achieved. At that point the student can collect additional points on this part and achieve a grade higher than E.

## The number of points and grades

In total: 20 points

Sufficient for grade D: 4 points

Sufficient for grade C: 8 points

Sufficient for grade B: 14 points

Sufficient for grade A: 17 points

## Tasks

### Task 1 (4 points + 3 points)

An algorithm that sorts a sequence of integers can be illustrated with a sample sequence:

7 2 6 4 3 1 5

7 2 6 4 1 3 5

7 2 6 1 4 3 5

7 2 1 6 4 3 5

7 1 2 6 4 3 5

1 7 2 6 4 3 5

1 7 2 6 3 4 5

1 7 2 3 6 4 5

1 2 7 3 6 4 5

1 2 7 3 4 6 5

1 2 3 7 4 6 5

1 2 3 7 4 5 6

1 2 3 4 7 5 6

1 2 3 4 5 7 6

1 2 3 4 5 6 7

a) Create a method `sort` that accepts an array of integers and sorts it according to the given algorithm.

b) Let  $n$  denote the number of integers sorted. Determine in that case the time complexity of the algorithm in terms of the number of comparisons.

### Task 2 (1 point + 1 point + 1 point)

Complexity functions for various algorithms can be categorized in different  $\Theta$ -sets. Examples of such sets are:  $\Theta(1)$ ,  $\Theta(\log_2 n)$ ,  $\Theta(n)$ ,  $\Theta(n \log_2 n)$ ,  $\Theta(n^2)$ , and so on.

To which  $\Theta$ -sets do the following complexity functions belong?

- a)  $u(n) = 100 \log_2 n + 0.5 n + 2$   
 b)  $v(n) = 0.1 n \log_2 n + 100 (n + 1)/2$   
 c)  $w(n) = 100 n \log_2 n + n (n + 1)/4$

### Task 3 (4 points + 3 points + 3 points)

The interface `Iterator` defines an iterator over a list:

```
interface Iterator
{
    // hasElement returns true if the iterator points to
    // an element in the list, otherwise false is returned.
    boolean hasElement ();

    // element returns the element in the list referred to
    // by the iterator. If no such element exists, an
    // exception is thrown of type
    // java.util.NoSuchElementException.
    int element () throws java.util.NoSuchElementException;

    // move advances the iterator to the next element in
    // the list. If no such element exists, the iterator
    // is set to point to nowhere.
    public void move ();
}
```

Class `List` represents a list of integers:

```
class List
{
    // Class Node represents a node in a list.
    private static class Node
    {
        public int    value;
        public Node   next;

        public Node (int value)
        {
            this.value = value;
            this.next = null;
        }
    }

    // references to the first and last nodes in the list.
    private Node    first = null;
    private Node    last  = null;

    public List ()
    {
        first = null;
        last  = null;
    }

    // add adds a given integer to the end of the list.
    public void add (int value)
    // code is missing here
}
```

```
private class ListIterator implements Iterator
{
    private Node    current = null;

    public ListIterator ()
    {
        current = first;
    }

    // code is missing here

}

public Iterator iterator ()
{
    return this.new ListIterator ();
}
}
```

- a) Create the class `ListIterator`: use that which is given, and add what is missing.
- b) Create the method `add`.
- c) Create a list of type `List` that contains the integers 1, 2, 3, 4, and 5. Then create an iterator for the list, and use it to visit and display all elements in the list.