

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 8 6 1 4 3 2 5

1 8 6 7 4 3 2 5

1 2 6 7 4 3 8 5

1 2 3 7 4 6 8 5

1 2 3 4 7 6 8 5

1 2 3 4 5 6 8 7

1 2 3 4 5 6 8 7

1 2 3 4 5 6 7 8

- Create a method `sort` that accepts an array of integers and sorts it according to the given algorithm.
- Let n denote the number of integers sorted. Determine the worst case time complexity of the algorithm in terms of the number of element exchanges. Categorize the corresponding complexity function: to which θ -set does it belong?

Task 2 (3 points)

Complexity functions for several useful algorithms belong to the set $\theta(n^2)$. Some sorting algorithms, for example, are $\theta(n^2)$ -algorithms in terms of time.

How can you determine if a complexity function belongs to this set: define the set $\theta(n^2)$ with mathematical precision.

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

The class `Vector` represents an array that can be adapted to different types of elements:

```
public class Vector<E>
// E - the element type
{
    // the array elements
    private Object[] elements;

    // the number of elements in the array
    private int countElements;
```

```
// Vector creates an array with a given initial capacity
public Vector (int initialCapacity)
{
    elements = new Object[initialCapacity];
    countElements = 0;
}

// toString returns the string representation of the array
public String toString ()
{
    StringBuilder s = new StringBuilder ("[");
    for (int pos = 0; pos < countElements - 1; pos++)
        s.append (elements[pos] + ", ");
    if (countElements > 0)
        s.append (elements[countElements - 1]);
    s.append ("]");

    return s.toString ();
}

// add adds a given element to the array
// code is missing here

// get returns the element in a given position
// code is missing here
}
```

An array is created and used like this:

```
Vector<Integer> v = new Vector<Integer> (2);
v.add (new Integer (10));
v.add (new Integer (20));
v.add (new Integer (30));
v.add (new Integer (40));
// v.add (new Double (50)); // (1)
System.out.println (v);

Integer n = v.get (3);
System.out.println (n);
```

When this code fragment is executed, the following printout is generated:

```
[10, 20, 30, 40]
40
```

If statement (1) is included a compilation error occurs: only elements of type `Integer` can be stored in the array `v`.

- a) Implement method `add`.
- b) Implement method `get`.
- c) Draw the object referred to by reference `v`.