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 integer numbers can be illustrated with a concrete sequence:

```
7 2 6 8 4 3 9 1 5

2 7 6 8 4 3 9 1 5
2 6 7 8 4 3 9 1 5
2 6 7 4 8 3 9 1 5
2 6 7 4 3 8 9 1 5
2 6 7 4 3 8 1 9 5
2 6 7 4 3 8 1 5 9

2 6 4 7 3 8 1 5 9
2 6 4 3 7 8 1 5 9
2 6 4 3 7 1 8 5 9
2 6 4 3 7 1 5 8 9

2 4 6 3 7 1 5 8 9
2 4 3 6 7 1 5 8 9
2 4 3 6 1 7 5 8 9
2 4 3 6 1 5 7 8 9

2 3 4 6 1 5 7 8 9
2 3 4 1 6 5 7 8 9
2 3 4 1 5 6 7 8 9

2 3 1 4 5 6 7 8 9

2 1 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9
```

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

b) Let n denote the number of integers being sorted. Determine the worst case time complexity in terms of element exchanges. Categorize the corresponding complexity function: to which Θ–set does it belong?

Task 2 (3 points)

In order to categorize various complexity functions and their corresponding algorithms, one sometimes use the so called 'big O'-notation. A complexity function may belong to a certain O–set. One example of such a set is $O(n \log_2 n)$.

How can you determine if a complexity function belongs to the set $O(n \log_2 n)$? Define this set with mathematical precision.

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

The class `List` represents a list that can be adapted to different types of elements:

```
public class List<E>
// E – the element type
{
    private class Node
    {
        public E       element;
        public Node    nextNode;

        public Node (E element)
        {
            this.element = element;
            this.nextNode = null;
        }
    }

    // the first node in the sequence
    private Node    firstNode;

    // List creates an empty list
    public List ()
    {
        firstNode = null;
    }

    // toString returns a string that represents the list
    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("[");
        Node    node = firstNode;
        if (node != null)
        {
            while (node.nextNode != null)
            {
                sb.append (node.element + ", ");
                node = node.nextNode;
             }
            sb.append (node.element);
        }
        sb.append ("]");

         return sb.toString ();
    }

    // add appends a given element to the end of the list
    // code is missing here

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

A list is created and used like this:

```
List<String>    list = new List<String> ();
System.out.println (list);
```

```
list.add (new String ("A"));
list.add (new String ("B"));
list.add (new String ("C"));
list.add (new String ("D"));
// list.add (new StringBuilder ("E")); // (1)
System.out.println (list);

String    s = list.get (1);
System.out.println (s);
```

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

```
[]
[A, B, C, D]
B
```

If statement `(1)` is included there will be a compilation error; only elements of type `String` can be stored in the list `list`.

a) Implement the method `add`.

b) Implement the method `get`.

c) Draw the object referred to by reference `list`.