JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2019
Stefan Grünberger, Dari Trendafilov

Assignment 2

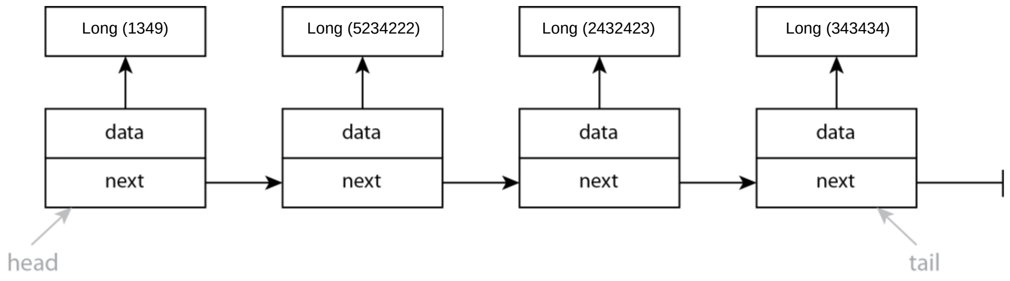
Deadline: **Mon. 4.11.2019, 23:59**
Submission via: **www.pervasive.jku.at/Teaching/**

Abstract Data Types

1. Priority Queue with lists — 7 + 3 points

Implement the abstract data type **priority queue** (see lecture) using a **single-linked list**.

a) First implement the single-linked list with a head and tail node at the beginning and end of the list as shown in the following figure (the first and last node of the list are marked as head and tail respectively):



The list should store elements of type **Long**. Use the **compareTo** method to compare the elements with each other when inserting sorted elements.

For implementation, use **exactly** the specified interface (a separate file for each class):

```
public class MyLinkedList<T> implements MyList<Long> {... }
```

```
public interface MyList <T>{
        int size();              //Returns number of elements in the List (in O(1))
        void addFirst(Long elem) //Adds element at the beginning of the list
                throws IllegalArgumentException;
        void addLast(Long elem)  //Adds element as end of list
                throws IllegalArgumentException;
        void addSorted(Long val) //Adds element to the list (sorted in asc order)
                throws IllegalArgumentException;
        void sortASC();  //Sorts the list in ascending order
        void sortDES();  //Sorts the list in descending order
        void clear();            //Clears the list by removing all elements (in O(1))
        Long removeFirst();      //Returns and removes the first element from the list
        Long removeLast();       //Returns and removes the last element from the list
        Long getFirst(); //Returns the first element from the list (no removal)
        Long getLast();  //Returns the last element from the list (no removal)
        boolean contains(Long val)//Returns true if T is in list; false otherwise
                throws IllegalArgumentException;
        Long get(int index);     //Returns element at index position (no removal)
        Long remove(int index); //Returns and removes element at index position
        String toString();       //Returns a string representation of the list
        Object[] toArray();      //Returns the list as array
}
```

Assignment 2

Use the following class **MyListNode** for the nodes in the list:

```
public class MyListNode<T> {

        private Long element;
        private MyListNode< Long > next;

        public MyListNode() {
                this.element = null;
                this.next = null;
        }

        public MyListNode(Long element) {
                this.element = element;
                this.next = null;
        }

        public Long getElement()                  { return this.element; }
        public void setElement(Long element)      { this.element = element; }
        public MyListNode< Long > getNext()               { return this.next; }
        public void setNext(MyListNode< Long > next)      { this.next = next; }
}
```

b) Implement the interface **MyPriorityQueue** in the form of the abstract data type **MyListPriorityQueue**, which uses the class **MyLinkedList** (from example 1a) to implement the methods.

For implementation, use exactly the specified interface (a separate file for each class):

```
public class MyListPriorityQueue <T> implements MyPriorityQueue< Long > {…}
```

```
public interface MyPriorityQueue <T> {
        public boolean isEmpty();        //Returns true if the PQ is empty; false otherwise
        public int size();               //Returns the current size of the PQ
        public void insert(Long elem)    //Inserts a new element into the PQ
                        throws IllegalArgumentException;
        public Long removeMin()          //Removes the min element from the PQ and returns it
                        throws NoSuchElementException;
        public Long min()                //Returns the min element from the PQ
                        throws NoSuchElementException;
        public Object[] toArray();       //Returns an array representation of the PQ
}
```

The implementation **MyListPriorityQueue** should internally use the class MyLinkedList from example 1a (do not derive from it) to provide the above methods.

Always check the passed arguments (*null*-objects must not be inserted into the list) and make sure that your *programming style is clean*. *Comment* those parts of your program, that are difficult to understand.

The Java sources for the implementation of *MyLinkedList*, *MyListNode, MyListPriorityQueue* must be submitted. You don't need to submit the given interfaces, test drivers or outputs of your tests – your implementations will be tested by the tutors! ***Therefore, strictly stick to the given interface!***

This also applies for all further assignments.

JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2019
Stefan Grünberger, Dari Trendafilov

Assignment 2

Deadline: **Mon. 4.11.2019, 23:59**
Submission via: **www.pervasive.jku.at/Teaching/**

2. Priority Queue with MinHeap 14 points

Implement the interface **MyPriorityQueue** in form of the abstract data type **MinHeap** (=Heap, where the smallest key is placed in the root). For your implementation, use exactly the specified interface (a separate file for each class):

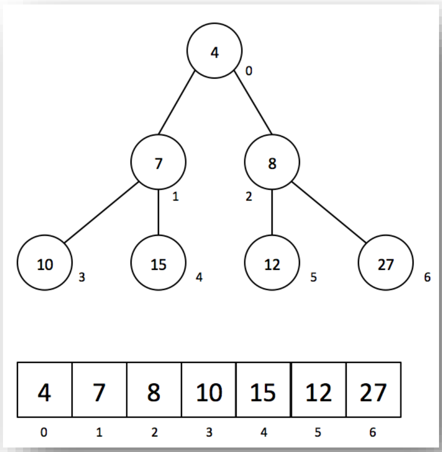
```
public interface MyPriorityQueue<T> {
        public boolean isEmpty();// return true if Priority Queue is empty
        public int size();       // return number of elements (in O(1))
        public void insert(Long val) // insert new element into Priority Queue
                throws IllegalArgumentException;
        public Long removeMin()  // remove and return the minimum element
                throws NoSuchElementException;
        public Long min()        // return the minimum element
                throws NoSuchElementException;
        public Object[] toArray(); // return the Priority Queue in array-representation
        public String toString();// return Priority Queue in string format
}
```

```
public class MinHeap<T> implements MyPriorityQueue<Long> { … }
```

Use an array, which is created with a certain size *initCapacity* in the constructor, for storing the nodes. Every time the array size is exceeded, i.e. if the n+1th element is inserted into a MinHeap of size n, the array size has to be **doubled**.

```
public MinHeap(int initCapacity) { … }
```

Nodes shall be stored „line by line” in the array, as the following example shows:



To make your code easier to read, consider which (private) help methods might be useful. Here's a suggestion:

```
private void upHeap(int index);
private void downHeap(int index);
private int parent(int index);
private int leftChild(int index);
private int rightChild(int index);
private void swap(int index1, int index2);
```

Always check the passed arguments (*null*-objects must not be inserted into the list) and make sure that your *programming style is clean*. *Comment* those parts of your program, that are difficult to understand.

The Java sources for the implementation of *MinHeap* must be submitted. You don't need to submit the given interfaces, test drivers or outputs of your tests – your implementations will be tested by the tutors! ***Therefore, strictly stick to the given interface!***

This also applies for all further assignments.