TDDC91,TDDE22,725G97 Lektion 3
Grafer och Lab4

Magnus Nielsen
magnus.nielsen@liu.se

2 oktober 2018

- Uppgift 1 - Optimera följande klass.

```
public class SortedList {
    private ArrayList<Integer> contents;

    public SortedList() {
        contents = new ArrayList<Integer>();
    }

    public void insert(Integer i) {
        contents.add(i);
        sort();
    }

    public int getAndRemoveFirst() {
        Integer result = contents.get(0);
        contents.remove(result);
        return result;
    }

    public int getAndRemoveLast() {
        Integer result = contents.get(contents.size()-1);
        contents.remove(result);
        return result;
    }

    private void sort() {
        for (int i = 0; i < contents.size(); i++) {
            bool swapped = false;
            for (int j = contents.size()-1; j > i; j--) {
                if (contents[j]<contents[j-1]) {
                    swap(contents[j], contents[j-1]);
                    swapped = true;
                }
            }
            if (!swapped) {
                break;
            }
        }
    }
```

```
    private void swap(Integer a, Integer b) {
        Integer c = a;
        a = b;
        b = c;
    }
}
```

Bonus: Försök finna vad som skulle kunna gå fel i nuvarande implementation!
Integer används för att göra det lättare att tänka igenom koden, dock skulle vilken jämförbar datatyp som helst kunna användas (eller göra klassen generisk!). Håll detta i åtanke när du funderar över lösningar.

- Uppgift 2 - Optimera följande klass(er).

```
public class Tree {
    private Node root;

    public Tree(){
        root = null;
    }

    public void insert(int i) {
        if (root == null) {
            root = new Node(i);
        } else {
            root.insert(i);
        }
    }

    public void remove(int i) {
        if (root != null) {
            root = root.remove(i);
        }
    }

    public void size() {
        if (root == null) {
            return 0;
        }
        return root.count();
    }

    // Internal node class
    private class Node {
        private Node right;
        private Node left;
        public int key;

        public Node(int i) {
            key = i;
            left = null;
            right = null;
        }
```

```
        public int count() {
            if (left != null && right != null) {
                return 1 + left.count() + right.count();
            } else if (left != null) {
                return 1 + left.count();
            } else if (right != null) {
                return 1 + right.count();
            } else {
                return 1;
            }
        }

        public void insert(Node n) {
            if (n.key < key) {
                if (left == null) {
                    left = n;
                } else {
                    left.insert(n);
                }
            }
            else if (n.key > key) {
                if (right == null) {
                    right = n;
                } else {
                    right.insert(n);
                }
            }
        }

        public Node remove(int i) {
            if (key == i) {
                if (left != null) {
                    if (right != null) {
                        left.insert(right);
                    }
                    return left;
                } else if (right != null) {
                    return right;
                } else {
                    return null;
                }
            } else if (key > i) {
                left = left.remove(i);
                return this;
            } else if (key < i) {
                right = right.remove(i);
                return this;
            }
            return null;
        }
        // End of Node class
    }
}
```

OBS!!! Delar av koden är direkt dålig, avsiktligt.