Ugeseddel 6

23. oktober - 30. oktober

- Læs kapitel 10 og 11

- Løs følgende opgaver

  Opgave 20: 8.2 (1 point)
  Opgave 21: 8.11 (2 point)
  Opgave 22: Se de næste sider (4 - 6 point, ikke-obligatorisk)

Afleveringsfrist: **tirsdag den 6. november**

På næste side er angivet et udkast til en klasse, `OneWayList`, til håndtering af *envejslister*. Hvert element i en envejsliste har en reference til sin efterfølger. Internt repræsenteres elementerne ved hjælp af den private, statiske klasse `Node`. Et objekt, `header`, af denne klasse udgør et hoved for listen.

Spørgsmål 1
Programmer metoderne `addFirst` og `isEmpty`.
`addFirst(obj)` skal tilføje objektet `obj` forrest i listen.
`isEmpty` skal returnere `true`, hvis listen er tom; ellers `false`.

Spørgsmål 2
Programmer klassen `OneWayListIterator`.

Spørgsmål 3
Programmer metoden `reverse`, der skal vende om på rækkefølgen af listens elementer.
Metodens lagerforbrug bør være konstant, og altså uafhængigt af listens længde.

Spørgsmål 4
Programmer de to udgaver af metoden `sort`. Kaldet `sort()` skal bevirke, at listens elementer sorteres i stigende orden, dog kun under forudsætning af, at de alle tilhører en klasse, der implementerer grænsefladen `Comparable`. Et kald af `sort(comp)` skal bevirke, at listens elementer sorterer ifølge `Comparator`-objektet `comp`.

Metodernes lagerforbrug må ikke være O(n) eller mere. Det er således ikke tilladt at bruge et hjælpearray til sorteringen.

Benyt f.eks. sortering ved udvælgelse (tidskompleksitet $O(n^2)$). En implementering, der benytter sig af sortering ved fletning (tidskompleksitet O(n log n)), belønnes med 2 ekstrapoint.

NB. Udkastet til klassen kan med fordel hentes fra kursets hjemmeside via henvisningen "Kode til opgaver".

```
import java.util.*;

public class OneWayList {
    public OneWayList() { header = new Node(null, null); }

    public void addFirst(Object obj) { /* se spørgsmål 1 */ }

    public boolean isEmpty() { /* se spørgsmål 1 */ }

    public Iterator iterator() { return new OneWayListIterator(

    public void reverse() { /* se spørgsmål 3 */ }

    public void sort() { /* se spørgsmål 4 */ }

    public void sort(Comparator comp) { /* se spørgsmål 4 */ }

    public String toString() {
        String result = "";
        Iterator it = iterator();
        while (it.hasNext())
            result += it.next() + " ";
        return result.trim();
    }

    private static class Node {
        Node(Object data, Node next) {
            this.data = data;
            this.next = next;
        }
        Object data;
        Node next;
    }

    private class OneWayListIterator implements Iterator {
        /* se spørgsmål 2 */
    }

    private final Node header;
}
```

fortsættes

Et simpelt testprogram er angivet nedenfor.

```
public static void main(String args[]) {
    OneWayList list = new OneWayList();
    Random rand = new Random(7913);
    for (int i = 0; i < 10; i++)
        list.addFirst(new Integer(rand.nextInt(100)));
    System.out.println("Original:\n" + list);
    list.sort();
    System.out.println("Sorted:  \n" + list);
    list.reverse();
    System.out.println("Reversed:\n" + list);
}
```