

Übungsblatt 10

Abgabe: 23.1. – 25.1.06 in den Übungen

Die Lösung dieses Übungsblatts soll in Gruppen von je 2 Personen erfolgen.
Die Lösungen sind inklusive Lösungsweg auf Papier abzugeben. Auf jedem Lösungsblatt sind die Matrikelnummern beider Gruppenmitglieder anzugeben!

Aufgabe 1 (7 Punkte)

Im Folgenden sollen Sie eine Klasse **List** schreiben, welche Objekte eines bestimmten Typs innerhalb einer einfach verketteten Liste verwalten soll. Die Operationen zum Einfügen (**insert()**) in die Liste sollen dabei so definiert werden, dass die Liste stets lexikographisch sortiert ist. Die Operation zum Umkehren der einfach verketteten Liste (**reverse()**) sollte so definiert sein, dass der Kopf das neue Ende und das Ende der neue Anfang wird. Alle Verweise werden dabei gedreht.

Hinweise:

Schreiben Sie die Klassen **Node** und **List**, welche die Interface **NodeI** bzw. **ListI** implementieren. Die Testklasse **ListDemo** muss funktionsfähig sein.

```
interface NodeI<T extends Comparable<T>> {
    T getItem();
    NodeI<T> getNext();
    void setNext(NodeI<T> next);
}

interface ListI<T> {
    void insert(T value);
    void showList();
    void reverse();
}

class ListDemo {
    public static void main(String args[]) {

        int[] int_values = {7, 4, 11, 5, 99, 2};

        ListI<Integer> intList = new List<Integer>();

        for (int i = 0; i < int_values.length; i++) {
            intList.insert(int_values[i]);
        }
    }
}
```

```

intList.showList();
System.out.println("=====");
intList.reverse();
intList.showList();

System.out.println("+++++");

String[] str_values
    = {"oo", "xyz", "abc", "java", "generics"};

ListI<String> strList = new List<String>();

for (int i = 0; i < str_values.length; i++) {
    strList.insert(str_values[i]);
}

strList.showList();
System.out.println("=====");
strList.reverse();
strList.showList();
}
}

```

Aufgabe 2 (7 Punkte)

Vollziehen Sie nach, was in folgendem Java-Code passiert und erläutern Sie die Ausgabe.

```

import static java.lang.System.*;

class X {
    int a = 4;
    int get() { return a; }
}

class Y extends X {
    static int a = 7;
    int get() { return a;}
    static void set(int x) { a = x; }
    static void set(char c) { a = 2*c; }
}

public class Z extends Y {

    static int b = 3;
    int get() { return b+a;}
    static int get(X x) { return x.a;}
    static void set(int i) { a = 3*i; }
    static void set(X x, int i) { a = i; }
}

```

```

static void test() {
    Z z = new Z();
    out.println( z.a );
    out.println( get(z) );
    out.println( ((X)z).get() );
    z.set('c' - 'a' - 1);
    out.println( get(z) );
    out.println( z.get() );
    Y y = z;
    y.set(2);
    out.println( z.get() );
    z.set(y, 0);
    out.println( y.get() );
}

public static void main(String[] a) {
    test();
}
}

```

Aufgabe 3 (6 Punkte)

Definieren Sie eine Klasse **InverseComparator**, mit deren Hilfe im folgenden Programm die Argumente in umgekehrter lexikographischer Reihenfolge ausgegeben werden:

```

class InterfaceDemo {

    public static void main(String args[]){
        java.util.Arrays.sort(args, new InverseComparator());
        for (int i = 0; i < args.length; i++)

            System.out.println(args[i]);
        }
    }
}

```