Hand in assignment 3 for Programming 2IS45 (fall 2012)

For this assignment you should hand in one java-file IntList.java via the PEACH-system (http://peach.win.tue.nl/) before Friday 26th October 23.00h. The file should contain your name and student id in the form of a comment at the beginning of the file. It should NOT contain a package line. After submission your implementation will be tested automatically and the results of these tests are provided as feedback in PEACH. Make sure your implementation passes all tests.

Integer lists

Integer lists are finite sequences of integers like \([3, 5, -7, 0]\), \([-2, 0, 2]\) and \([\ ]\) (the empty list). An integer list is either the empty list \([\ ]\) or a non-empty list \(x;S\) consisting of an integer \(x\) prepended to an integer list \(S\). The length of integer list \(S\) (the number of elements of \(S\)) is denoted by \(|S|\). The positions of the elements of \(S\) are numbered from 0 upward and the element at position \(i\) (\(0 <= i < |S|\)) is given by \(S[i]\). List \(S\) can be concatenated with list \(T\) yielding list \(S++T\).

1 Informal description

In this assignment you are asked to implement a class \(\text{IntList}\) (according to a given design) to represent and maintain a list of integers (as described above).

Informally, the class \(\text{IntList}\) should allow for the creation of an empty list, verifying whether a list is empty, and retrieving the number of elements of a list. It should allow for the inspection of a list element at a given position, the removal of such an element, and the insertion of an integer at such a position. Verifying whether a given integer occurs in the list and obtaining the first position where it occurs should be possible as well as removing one or all occurrences of a given integer.

2 Interface of class \(\text{IntList}\)

The following interface listing is a formalisation of the informal description above.

```java
// Class IntList implements a sequence (list) of integers.
// The positions of the list elements are numbered from 0 upward.
// It provides the following public operations:
// - IntList constructor: create list, initially empty
// - count number of elements in the list
// - isEmpty return whether the list is empty
// - occurs(x) return whether x occurs in the list
// - indexOfe(x) return the index in the list at which position x occurs
// - elemAt(index) return element at position index, provided it exists
// - insertAt(x, index) insert x at position index in the list shifting
//   all list elements after it to a position one higher
// - remAt(index) remove the element at position index from the list, provided it exists
// - remOne(x) remove the first occurrence of x from the list, provided it exists
// - remAll(x) remove all occurrences of x from the list

public class IntList {
    // Constructors
    public IntList() {...}
    ...
    // Queries

    // Utility methods
    public int[] toArray() {...}
    public String toString() {...}
}
```
3 Contract of class IntList

Taking a sequence of integers L as the model variable the following listing provides the contracts for the class IntList.

```java
public class IntList {

    // Private part
    // private instance variables of
    // the internal representation (provided in the next section)

    // Private invariants
    // private invariants (representation invariants) that the variables
    // of the internal representation satisfy (provided in the next section)

    // Representation
    // the abstraction function mapping the internal representation
    // onto the model (provided in the next section)

    // Model variables
    // L: sequence of Int

    // Public invariants
    // none

    // Constructors
    // pre: true
    // post: L == []
    public IntList() {
        // TODO implement actual body
    }

    // Additional constructors ...

    // Queries
}
```
// pre: true
// return: L == [] (Is the list empty?)
public boolean isEmpty() {
    // TODO implement actual body
}

// pre: true
// return: |L|, the length of L (= number of elements in the list)
public int count() {
    // TODO implement actual body
}

// pre: true
// return: Does there exist an i with 0 <= i < |L| with L[i] == x?
public boolean occurs(int x) {
    // TODO implement actual body
}

// pre: true
// return: i where 0 <= i < |L|, L[0..i-1] does not contain x, L[i] == x
// or -1 if x does not occur in L
public int indexOf(int x) {
    // TODO implement actual body
}

// Additional queries ...

// Commands
// pre: 0 <= index <= |L|, L == S ++ T where |S| == index
// post: L == S ++ [x] ++ T
public void insertAt(int x, int index) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

// pre: 0 <= index <= |L|, L == S ++ [x] ++ T where |S| == index
// post: L = S ++ T
public void remAt(int index) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

// pre: true
// post: L' = 'L with the first x removed (if any)
public void remOne(int x) {
    // TODO implement actual body
}

// pre: true
// post: L' = 'L with all values x removed
public void remAll(int x) {
    // TODO implement actual body
}

// Additional commands ...
4 Internal representation using a linked list

Consider the following internal representation/implementation for class IntList consisting of a reference variable fHead to a chain of linked objects of class IntNode (representing the integer list) and an integer variable fCount containing the number of elements of the list:

```java
class IntNode {
    int item;  // integer contained in this node
    IntNode next;  // reference to next node in the chain (if any)

    IntNode(int x, IntNode n) {
        item = x;
        next = n;
    }
}

public class IntList {

    // Private part
    private IntNode fHead;  // reference to first IntNode in chain, if any
    private int fCount;  // number of IntNodes in chain referenced by fHead

    // Private invariants
    // Pri0: 0 <= fCount
    // Pri1: fHead(.next)ˆfCount == null, where (A)ˆB stands for B repetitions of A
    // N.B. in particular if fCount == 0, then fHead == null

    // Representation
    // L == [ fHead.item, fHead.next.item, fHead.next.next.item, ...
    // (elements are numbered from 0 onwards)
    // N.B. in particular if fCount == 0, then L == [] is represented
    // count == fCount

    // Model variables
    // L: sequence of int

    // Public invariants
    // none
}
```

The following object structure for instance represents the integer list [5, 7, 8] (if a reference variable contains a reference to an object it is indicated with an arrow to that object, otherwise it just contains null)

![Object structure](image)

The IntNode objects in the chain (linked list) can be referenced by fHead ( = fHead(.next)

0), fHead.next ( = fHead(.next)

1), and fHead.next.next ( = fHead(.next)

2). The values of the item instance variables within these nodes are, in order, the integers in the list represented.
5 Assignment

On the course website a complete NetBeans project **IntListTester_practicum** is available. Download the zip-file containing the project, unzip it to a suitable location, and open the project in NetBeans. The project contains a file **IntList.java** with the class contract as given in section 3, with the instance variables for the internal representation, representation invariants, abstraction function, but without implementations of its methods (functions have a default return statement in order for the project to be runnable). All locations in the file where something needs to be filled in are marked with // **TODO ...** comments. It also contains a file **IntNode.java** with the class declaration for integer nodes in linked lists.

The project provides a GUI-application that provides a manual testing interface for class **IntList**. Its user interface looks like this:

![Image of GUI interface](image)

Implement all methods according to their contracts (thereby replacing the default return statements given for the functions). Make sure that preconditions are checked with an **assert** statement with an informative message. Make sure your name and student id. are in a comment in the beginning of the file.