Länkade listor

Tema: Implementera klasser med hjälp av enkellänkade listor.

Enkellänkad lista

I några av uppgifterna används klasserna `SingleLinkedList` och `ListNode`:

```
public class SingleLinkedList<E>  {
    private ListNode<E> first;

    /** Creates an empty list. */
    public SingleLinkedList() {
        first = null;
    }

    /** Inserts the specified element at the beginning of this list. */
    public void addFirst(E e) {
        ListNode<E> n = new ListNode<E>(e);
        n.next = first;
        first = n;
    }
    ...

    /* Nested class. Represents a node which contains an element of type E. */
    private static class ListNode<E> {
        private E element;
        private ListNode<E> next;

        /* Creates a listnode which contains e. */
        private ListNode(E e) {
            element = e;
            next = null;
        }
    }
}
```

U 1. Skriv programrader som skapar ett lista av typen `SingleLinkedList` för att lagra heltal samt ser till att listan innehåller talen 1, 2, 3.

U 2. a) Lägg till följande metod i klassen `SingleLinkedList`:

```
/** Returns the first element in this list.
    Throws NoSuchElementException if this list is empty. */
E getFirst();
```

b) Lägg till följande metod i klassen `SingleLinkedList`:

```
/** Returns the last element from this list.
    Throws NoSuchElementException if this list is empty. */
E getLast();
```

U 3. Lägg till följande metod i klassen `SingleLinkedList`:

```
/** Returns true if this collection contains the specified element. */
public boolean contains(Object x);
```

U 4. Lägg till följande metod i klassen `SingleLinkedList`:

```
/** Removes the first occurrence of the specified element from this list,
    if it is present. If this list does not contain the element, it is unchanged.
    Returns true if this list contained the specified element (or equivalently,
    if this list changed as a result of the call). */
boolean remove(Object e);
```

Ledning:

- Utgå från din lösning på uppgift U 3.
- Tänk på att du för att kunna ta bort ett element måste ha en referens till föregående element i listan. Håll därför reda både på aktuellt element och dess föregångare när du går framåt i listan under sökningen.
- Tänk på specialfallen (tom lista, det är första elementet som ska tas bort ...).

Dubbeländad kö

U 5. En `dequeue` (eng: double-ended queue) är en följd av element där det är tillåtet att ta bort första och sista elementet och att lägga in nya element först och sist. Implementera en `dequeue` med en enkellänkad lista enligt följande:

```
public class Dequeue<E> {
    private Node<E> first;  // reference to the first element
    private Node<E> last;   // reference to the last element

    /** Creates an empty dequeue. */
    public Dequeue() {
        first = last = null;
    }

    /** Inserts the specified element at the beginning of this dequeue. */
    public void addFirst(E x) {...}

     /** Inserts the specified element at the end of this dequeue. */
    public void addLast(E x) {...}

    /** Removes and returns the first element in this dequeue.
        Returns null if this dequeue is empty. */
    public E removeFirst() {...}

    /** Removes and returns the last element in this dequeue.
        Returns null if this dequeue is empty. */
    public E removeLast() {...}

    private static class Node<E> {
        private E element;
        private Node<E> next;

        private Node(E element) {
            this.element = element;
            next = null;
        }
    }
}
```

U 6. I föregående uppgift representeras noderna i kön av en statisk klass `Node<E>` som är deklarerad i klassen `Dequeue`.

a) Om vi stryker `static` i deklarationen av nodklassen blir den en s.k. inre klass. (En inre klass är en icke-statisk klass deklarerad i en annan klass). Vad blir skillnaden?

b) Kan vi låta nodklassen vara en inre klass?