

## Übungsblatt 9.2

Die Aufgaben auf diesem Übungsblatt befassen sich mit der Nutzung und der Deklaration von Iteratoren bei generischen Klassen. Die Aufgaben 2 und 3 stellen die generischen Varianten der entsprechenden Aufgaben des Übungsblatts 9-1 dar.

### Aufgabe 1 - Vergleich von Listen mit Iteratoren

Die Methode `boolean id(DoublyLinkedList<T> other)` soll `true` zurückgeben, wenn die ausführende und die dem Parameter `other` übergebene Liste an allen Positionen die gleichen Inhalte haben.

```
public boolean id( DoublyLinkedList<T> other )
{
    if ( size() != other.size() )
    {
        return false;
    }
    else
    {
        Iterator<T> it1 =  ;
        Iterator<T> it2 =  ;
        while (  )
        {
            
        }
    }
    return true;
}
```

### Aufgabe 2 - Iterator über viele Listen

Die Klasse `NLists` verwaltetet `n` Listen in einer Liste. Die Klasse `NLists` soll eine Iterator bereitstellen, der die Inhalte aller `n` Listen nacheinander liefer. Ergänze den vorgegebenen Code passend. In der bekannten Klasse `DoublyLinkedList` stehen u.a. die Methoden `size`, `get` und `iterator` zur Verfügung.

```
public class NLists<T>
{
    DoublyLinkedList<DoublyLinkedList<T>> lists;

    public NLists( int n )
    {
        lists = new DoublyLinkedList<DoublyLinkedList<T>>();
        for ( int i = 0; i < n; i++ ) { lists.add( new DoublyLinkedList<T>() ); }
    }

    public void add( int listNo, T content )
    {
        lists.get( listNo ).add( content );
    }

    public Iterator<T> iterator()
    {
        
    }
}
```

```
private class NListsIterator extends Iterator<T>
{
    Iterator<T> currentIterator; // kann den Iterator einer Liste referenzieren
    int currentListNo;          // kann eine der Listen identifizieren

    public NListsIterator()
    {
        currentListNo = 0;
        currentIterator = null;
        if ( lists.size() > 0 )
        {
            
        }
    }

    public boolean hasNext()
    {
        while ( currentListNo < lists.size() ) {
            if ( currentIterator.hasNext() ) {
                return true;
            }
            else
            {
                
            }
        }
        return false;
    }

    public T next()
    {
        
    }
}
```

### Aufgabe 3 - Entwurfsmuster Iterator

Die Klassen `Data` und `DataIterator` besitzen Konstruktoren und weitere Methoden, die hier aber nicht verwendet werden sollen.

```
public class Data<T>
{
    // ... (Konstruktor und weitere Methoden sind nicht von Interesse)

    public Iterator<T> iterator()
    {
        return new DataIterator();
    }

    private class DataIterator extends Iterator<T> {
        // ... (Konstruktor und weitere Methoden sind nicht von Interesse)
    }
}
```

Die Methode `int countFirst( Data<E> structure )` soll die Häufigkeit zurückgeben, mit der das erste durch den Iterator gelieferte Objekt in `structure` vorkommt. Gibt es kein Objekt, das durch den Iterator geliefert wird, so soll der Wert `0` zurückgegeben werden. Die Gleichheit soll mit der Methode `equals` bestimmt werden.

```
public class Use<E>
{
    public static int countFirst( Data<E> structure )
    {
        int count = 0;
        Iterator<E> it =  ;
        if (  )
        {
            E ref =  ;
            
            while (  )
            {
                
            }
        }
        return count;
    }
}
```