SYSC 2100, Winter 2013
Assignment 2: Linked Set
Due: February 13, 2013
(Due at noon that day)

In this assignment, you are to implement a set, and write a test program that adequately tests your set implementation. The set implementation has to implement the following interface (which you can download from the course website as well):

```
public interface SetInterface<T>
{
      /** Adds a new entry to this set, avoiding duplicates.
          Increases the number of entries by 1.
          @param newEntry  the object to be added as a new entry
          @return true if the addition is successful, or false if
                  the item already is in the set */
      public boolean add(T newEntry);

      /** Removes a specific entry from this set.
          Decreases the number of entries by 1.
          @param anEntry  the object to be removed
       @return true if the removal was successful, or false if not */
      public boolean remove(T anEntry);

      /** Removes an unspecified entry from this set.
          Decreases the number of entries by 1.
          @return either the entry if the removal
                  was successful, or null */
      public T remove();

      /** Removes all entries from this set. */
      public void clear();

      /** Tests whether this set contains a given entry.
          @param anEntry  the object that is the desired entry
          @return true if the set contains anEntry, or false if not */
      public boolean contains(T anEntry);

      /** Gets the size of this set.
          @return the integer number of entries currently in the set */
      public int getCurrentSize();

      /** Sees whether this set is empty.
          @return true if the set is empty, or false if not */
      public boolean isEmpty();

      /** Sees whether this set is full.
          @return true if the set is full, or false if not */
      public boolean isFull();

      /** Retrieves all entries that are in this set.
          @return newly allocated array of all entries in set */
      public T[] toArray();
} // end SetInterface
```

Your implementation should use, internally, a linked list to store the elements in the set. Therefore, your class that implements this set interface should be called `LinkedSet`.

Note that the definition uses Java Generics, as discussed in the course textbook in Section 5.5. A somewhat more detailed tutorial is available at http://docs.oracle.com/javase/tutorial/java/generics/. In essence, the use of generics allows a

program to instantiate a set with different elementary data types. For example, if you implemented the above interface with a class `LinkedSet`, you can create instances of sets with different elementary data types as follows:

```
SetInterface<String> set1 = new LinkedSet<String>();
SetInterface<Integer> set2 = new LinkedSet<Integer>();
```

In the first instance, you create a set that is composed of `String` objects, in the second instance the set contains `Integer` objects. You can operate on both types of sets with the exact same operations.

Once you implemented the set, using a linked list representation, write a program that instantiates one or multiple sets, and invokes a number of operations on them to test your application. Note that the TA may or may not use your specific test program to test your implementation though. It should be possible for the TA to run his test program with your implementation of `LinkedSet`. Also, the TA may choose to instantiate sets of different elementary data types.

Bonus part (3 extra marks): in addition to the operations specified above, sets typically provide operations such us union, intersection, and set difference. Implement these additional methods, based on the following extension to the interface:

```
/** merge the current set with a second set.
    @param anotherSet  the set to be merged with the current set
    @return union of the two sets */
public SetInterface<T> union(SetInterface<T> anotherSet);

/** form the intersection of the current set with a second set.
    @param anotherSet  the set to intersect with the current set
    @return intersection of the two sets */
public SetInterface<T> intersection(SetInterface<T> anotherSet);

/** Perform a set difference.
    @param anotherSet  the set to be subtracted from the current set
    @return current set minus anotherSet */
public SetInterface<T> difference(SetInterface<T> anotherSet);
```

Submission Requirements: Submit your assignment (the source files) using WebCT. Your program should compile and run as is in the default lab environment, and the code should be well documented. Submit all Java class files without using any archive or compression as separate files. The main program should be called `TestSet.java`, the implementation of your `LinkedSet` should be in file `LinkedSet.java`. Marks will be based on:

- Completeness of your submission
- Correct solution to the problem
- Following good coding style
- Sufficient and high-quality in-line comments
- Adhering to the submission requirements

The due date is based on the time of the WebCT server and will be strictly enforced. If you are concerned about missing the deadline, here is a tip: multiple submissions are allowed. So you can always submit a (partial) solution early, and resubmit an improved solution later. This way, you will reduce the risk of running late, for whatever reason (slow computers/networks, unsynchronized clocks, failure of the Internet connection at home, etc.).
In WebCT6, you can manage the submission until the deadline, taking it back, deleting/adding files, etc, and resubmitting it. The system also provides online feedback whether you submitted something for an assignment. It may take a while to learn the submission process, so I would encourage you to experiment with it early and contact the TA(s) in case you have problems, as only assignments properly and timely submitted using WebCT will be marked and will earn you assignment credits.