Lab 2: Introduction to generics and exceptions

Objectives: The main objective of this laboratory class is to;

- use generics when developing classes, and
- use exception handling in code.

Task:
For this laboratory class you are required to use the provided skeleton code. Some code cannot be changed – such places are marked in the code with comments.

Your task is to implement a *set* data structure. A set is basically a collection of data items where there is only one copy of a element. That is, item can appear at most once. Items can be in any order – the order in which the items are stored internally and the order in which they are moved is irrelevant.

Your set is expected to provide the following functions:

Function	Description
Constructor	Where one can, if need be specify the number of elements to be expected or not. If the number is not specified a default value should be used.
add	The add function should get as argument the item to be added and should return true if the set was modified – that is if the item was added. If there is an item in the set equal to the item to add it should not modify the set and return false.
remove	The remove function should return *some* element from the set. The order in which items are removed is irrelevant.

Have a look at the given skeleton code. As marked by comments some of the functions/statements you are not allowed to change and your design/implementation will have to work around them.

Implement the *constructors, add* and *remove* functions. You may add other functions as required.

Testing: We would test your code by typing "*$ java Test `cat Test.java`*". The word list is expected as command-line argument which `cat Test.java` will provide by reading the *Test.java* file. If you have implemented the sets correctly you should see the words used for writing the *Test.java* file (each word would appear at most once).

What to turn in: Submit your *MySet.java* file and *answer.txt* which answers the questions given below based on your implementation as a single tar ball. We will test you implementation of sets with the original Test.java given to you. So do not change Test.java.

Questions:

1. Suppose the set currently have *N* number of elements and one wants to add another. How many comparison operations do you need?
2. To remove an element how many operations do you need to do? Does it change with the number of elements in the set?
3. Suppose the items are kept sorted. Can you improve the performance of add? Explain.
4. Consider the two implementation options given for remove in the skeleton code. Which option is better? Why?.
5. Can you use Java annotation to get rid of -Xlint flag?