**LAB**

**Java Generics**

### **Problem 1**

Write a generic class Set with a type parameter T. The class uses an array of 100 elements of type T (initially just storing all NULL values). The class supports the following operations:

1. void insert(T element) - inserts an element into the set. An element is inserted into the first empty spot in the array. For now don't worry about the array being full - just do nothing if you can't insert an element. Also, don't worry about duplicate elements - just insert them.
2. delete(T element)- delete an element equal to element (use the equals method to determine equality).
3. T find(T element) returns the first element equal to element, as determined by equals method.

Test your code to create a Set of Strings (<String>Set) and a Set of Integers. Do not cast the elements returned by find - generic types don't require typecasting.  
Can you insert a String into a Set of Integers? Explain what happens when you try.

### **Problem 2**

Write an equals method for a Set class (two sets are equal if their arrays contain equals elements in the same order). Can you use *instanceof* T in your equals method? Why or why not? Use the equals methods of the elements to compare them.

### **Problem 3**

Change the Set implementation so that it stores elements in the increasing order. In this implementation equal elements should be inserted only once. The implementation doesn't have to be efficient - you may shift all elements when inserting a new one. Note that in this case the class only works for Comparable elements, so in the class declaration you need to say <T extends Comparable>. Check that the only objects you can use it with are those implementing Comparable interface.

Submit all you code, write answers to questions in comments.