Assignment 7
(Full Score: 80 points)

(**Due by 3/4/Friday-Midnight at Moodle**)

Your name:	Score:

Java Generics and *javadoc*

Create and test Java code that implements standard Java *java.lang.Comparable<T>* and *java.util.Comparator<T>* interfaces using Java generics.

Requirements:

1. Create a class *Student* that has the following 3 fields: *name, age, studentId*. The *Student* class must implement the *java.lang.Comparable<Student>* interface based on the *studentId* field and follow the standard OOP design rules:
   a. All data fields are *private.*
   b. The class has two constructors - a default one and a 3-argument one.
   c. All data fields have associated *public* accessor and mutator methods.
   d. The class has a *toString()* method that returns a human readable presentation of the class attributes (data fields).
   e. The *compareTo*(*Student o*) method of the *Student* class should delegate the comparison processing to the standard *String* class *compareTo(String o)* method based on the *studentId* field.

2. Code a separate class *StudentCompare* that implements the *java.util.Comparator<Student>* interface for the objects of type *Student* using the *age* field as the comparison criterion. The *compare*(*Student o1, Student o2*) method of the *StudentCompare* class performs comparisons using the *age* fields of the *o1* and *o2* instances of the *Student* class.

3. Provide a separate test class that:
   a. Creates and populates an **array** of type *Comparable<Student>* with **at least 5 different** *Student* instances.
   b. Performs sorting of this array using **both** of the comparison techniques implemented according to the items 1 and 2 above.
   c. Displays both sorted arrays using the *toString*() method of the *Student* class.

4. Use the test results from the item 3 above, to formulate in the assignment report your conclusions regarding the two comparison approaches described.

5. Your code should include *javadoc* comments for all classes, methods, and data fields.

6. The project submission should include the HTML documentation for your code generated by the *javadoc* utility.

Hints:

1. Use Java 6 API specification http://download.oracle.com/javase/6/docs/api/ for the information on Java 6.

2. The *javadoc* utility may be invoked from the command-line or inside NetBeans.

3. Use the appropriate sorting methods from the Java *java.util.Arrays* class to sort your array.

4. Use Java coding guidelines when naming your identifiers and creating class fields. The fields should be *private* or *protected* and appropriate accessor and mutator methods should be provided.

5. Save your work regularly, especially at the end of each class. Keep a detailed record of all steps performed.

Submission:
Submit your project ***including the HTML documents produced through javadoc*** in a compressed file to **Moodle by 3/4/Friday-Midnight**!

*************************Important************************
Your code must follow Java Coding Convention; otherwise, **20 points** will be deducted from your total scores.
****