Challenge Project 7

(**Due by 2/28/Monday-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. Using inheritance, create a subclass of the *Student* class that has an additional *GPA* (grade point average) field.

4. Create a subclass of the *StudentCompare* class using the *GPA* field as a secondary comparison criterion (the *age* field should remain the primary criterion and the *GPA* field comparison should be performed only when the values in the *age* field are equal).

5. 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 **all** of the comparison techniques implemented above.

    c. Displays both sorted arrays using the *toString*() method of the *Student* class.

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

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

Scores: in addition to 80 points for A#7, you will receive 1 point for the Challenge Project.

Submission:
Submit your project ***including the HTML documents produced through javadoc*** in a compressed file to **Moodle by 2/28/Monday-Midnight**!

**********Important**********
Your code must follow Java Coding Convention; otherwise, up to **100% points** will be deducted from your total scores.
****