**Purpose:**

To demonstrate your ability to ***design***, ***code***, and ***implement*** a fully ***functional*** ***class*** with multiple ***constructors***, and to ***instantiate*** and ***store*** ***objects*** of that class in a ***binary*** search ***tree*** (***BST***) ***structure***. To show your ***proficiency*** with ***validating*** data ***input*** to your class ***using*** ***exception*** ***handling*** for errors. To ***design*** and ***implement*** a simple ***menu*** process for ***user*** ***interaction***. This is a ***course*** ***capstone*** ***project*** in which you are expected to use most of what you learned in this course.

Make sure you devote plenty of time designing your solution before you begin coding. While it is not a particularly difficult project, neither is it simple.

**Specifications:**

Create a new Java project called ***Student Search*** and save it in a Final Project folder.

**Step 1**: Begin by creating a ***Student*** ***class*** that contains: a ***student Id*** (***integer***), a ***student*** ***name*** (***String***), a ***student*** ***major*** (***String***), three ***constructors*** (see below), ***gets*** and ***sets***, a ***compareTo*** method, and a ***toString*** method. Note, the class must ***implement*** the ***Comparable*** ***interface***.

1. Student Id must be greater than zero (0).
2. Ensure that there are no duplicate student Ids.
3. A student Id cannot be changed if it is not zero (0) [created by the default constructor].
4. Student name cannot be blank or null.
5. Student major must be ***three*** (***3***) chars – ***accept lower case and convert to upper case***.
6. In addition to a default and normal constructors, a ***special*** ***constructor*** with a student id and blanks for name and major will be needed for use in the ***BST*** ***find*** method.
7. All Student ***errors*** must be handled as ***Exceptions***. No exception should terminate the program.

**Step 2:** Design and begin coding the main method of your project. Using the ***BinarySearchTree***(***BST***) file provided, create a tree of a minimum of ***five*** (5) ***computer*** ***generated*** students. Display the tree of the generated students.

**Step 3**: Design your ***menu*** process. You may build it as a ***separate*** ***class*** in its ***own*** ***file*** or use it as a ***method*** in your ***program***.

**Step 4:** using the menu**,** provide the user with the ***ability*** to ***search*** the ***list*** for a student ID. If ***found***, ***display*** the student information; if ***not*** ***found*** display an appropriate ***message***. Provide for the ***addition*** of a new Student, if not found, and ***removal*** of the existing ***student,*** if found. Display the tree of all the students in the list after any change.

**Step 5:** Provide the user with the ability to enter a “***major***” and use it to ***list*** ***all*** the ***students*** in that ***major***.

**Step 6:** make the entire ***menu*** ***process*** ***repetitive***. Repeat the process ***until*** the ***user*** selects the ***menu*** ***entry*** to ***quit*** / ***exit*** the program. Clear the screen, then ***display the full tree before terminating the program***.

**Step 7:** Using the ***JAVADOC*** format, ***document*** the ***program*** and ***every*** ***method*** (main is a method, not a constructor) as ***displayed*** in the ***sample*** ***programs*** found in the ***downloaded*** ***files*** for classroom activity. ***Completeness***, ***accuracy***, and ***spelling*** count.

Notes:

1. The ***Binary Search Tree*** class is ***generic*** in form so that any “***Student***” ***returns*** will need to be ***cast*** as ***Student*** before use.
2. Be certain to ***examine*** the ***Binary Search Tree*** class carefully ***before*** attempting to ***use*** it so that you thoroughly ***understand*** what is ***passed*** to and ***returned*** from the class.

Submit the entire project folder as a .zip file.