**Roster.java** Write a class to simulate a class roster. A Roster consists of a list of Student objects and a class name, represented by a String. Complete the entire Roster class. The toString method is completed.

**public class Roster {**

**private List<Student> roster;**

**private String className;**

/\*\*

\* Creates a <code>Roster</code>

\* @param roster

\* @param className

\*/

**public Roster(List<Student> roster, String className) {**

/\*\*

\* Returns the name of the class.

\* @return the name of the class.

\*/

**public String getClassName() {**

/\*\*

\* Returns a list of Strings of all the names of students in the roster

\* @return a list of Strings of all the names of students in the roster

\*/

**public List<String> getNames() {**

/\*\*

\* Returns a list of unique majors as a <code>String</code>

\* @return a list of unique majors

\*/

**public List<String> getMajors() {**

/\*\*

\* Returns a list of all <code>Student</code> objects whose name has the

\* given sequence of characters of <code>str</code> somewhere in the

\* name of the student.

\* @param str the target string of characters

\* @return a list of all <code>Student</code> objects whose name contains the

\* given target string

\*/

**public List<Student> nameContains(String str) {**

/\*\*

\* Returns a new <code>Roster</code> of students occurring lexicographically before

\* the target <code>name</code> while ignoring case-sensitivity

\* @param name the target string

\* @return a new <code>Roster</code> of students occurring lexicographically before

\* the target <code>name</code> while ignoring case-sensitivity

\*/

**public Roster getNamesBefore (String name) {**

@Override

**public String toString() {**

**String string = className + ": \n";**

**for (Student student : roster) {**

**string += student + "\n";**

**}**

**return string;**

**}**

Copy your **Person** and **Student** programs into this **Project**.

**RosterApplication.java**  Create a file that will have data for a Student object. Each Student will be entered on one line, separated by a comma **“,”** for each data field. There will be no spaces before or after a comma.

Remember that a Student has a name (first and last), birth year, gender, and major. Use an integer for the gender (1 = Female, 2 = Male, 0 = Unknown).

**Sample File**

**Aaron Rogers,1983,2,Leadership**

**Jonathan Allen,1995,2,Personal Finance**

**Suzy Q,2004,1,Communications**

Read the data from a file into a String array. When reading the line from the file use the split method from the String class. Split each token at the comma. Create each String array into a Student object. Use Integer.parseInt to convert a string to an integer. Add that Student to the ArrayList.

Print the roster to ensure the entire file was read correctly. Print each student on a separate line.

Create a new Roster object. Name the class anything you choose.

Find the names of each student in the roster using the getNames method. Print each name in the list (Remember the Student class only prints the first name).

Print all the majors in the roster using the getMajors method on a separate line. Each major should only be printed once (no repeats).

Print all the names of students that contain a given string sequence anywhere in the name on a separate line using the nameContains method. The sequence is case-sensitive. Test multiple sequences (single letters and multiple letters).

Print a new class roster with all the students whose name comes alphabetically before the given string sequence on a separate line using the getNamesBefore method. The sequence is NOT case-sensitive.

Give a classmate you aren’t sitting next to a hi-five when you complete this program!