CSC 330 Project I Name: KOK

In this project you will use linked lists to implement a system called Study-Friends. The system keeps track of students and their friends (other students) that they study with. Your program(s) will accept a sequence of commands which will add or delete students; add or delete friends; list the friends of a particular student; and check to see if two students are friends or not.

The commands will be:

Command	Parameter	Description
S	<Student>	Add a Student to course
D	<Student>	Delete Student from course
P		Lists all students' names in the course
B	<Student1> <Student2>	Add Student2 into the friend list of Student1
N	<Student1> <Student2>	Delete Student2 from the friend list of Student1
L	<Student>	List all the friends of Student
Q	<Student1> <Student2>	Check if Student2 is in the friend list of Student1
X		Exit program

A Sample execution will look like:

```
Input                         Output

S Joe
S Bill
S Jane
P                             Joe Bill Jane
S Lisa
B Joe Bill
B Joe Jane
B Lisa Jane
B Joe Lisa
L Joe                         Bill Jane Lisa
Q Lisa Jane                   Yes
D Jane
P                             Joe Bill Lisa
Q Joe Jane                    No
L Lisa                        Nobody
L Joe                         Bill Lisa
X
```

CSC 330 Project I Name:
KOK

PART I (DUE 3/14/2017)

In this part you will implement four commands (S, D, P, X) by constructing a linked list of students that are in the course. The list will contain nodes where each node will contain a Student object and a link to another student node. You must implement `Student`, `StudentNode` and `StudentList` classes.

The `Student` class should inherit `Person` class with additional attributes of `major` and `GPA`. A complement of accessors and mutators as well as default and overloaded constructors should be implemented for this class

The `StudentNode` class should have the full complement of its accessors and mutators as well as default and overloaded constructors.

The `StudentList` class should have the full complement of its accessors and mutators as well as default constructor. In addition this class should have the following methods:

- `public void insertStudentNode(StudentNode):` inserts a `StudentNode` into the tail of `StudentList`
- `public void deleteStudentNode(StudentNode):` deletes a `StudentNode` from the `StudentList`
- `public StudentNode findStudentByName (String):` Finds the `StudentNode` that references the `Student` object that has the value of the String parameter in `name` attribute. If the `Student` does not exist in the list, `null` is returned.
- `public void printStudentListByName() :` Displays the names of all the students' in the list

CSC 330 Project I Name: KOK

PART II (DUE 3/21/2016)

In this part you will implement three commands (B, N, L) by constructing a linked list of friends of a student. The list will contain nodes where each node will contain a reference to a `StudentNode` object and a link to another friend node. You must implement `FriendNode` and `FriendList` classes as well as add an attribute of type `FriendList` to the `StudentNode` class.

The `FriendNode` class should have the full complement of its accessors and mutators as well as default and overloaded constructors.

The `FriendList` class should have the full complement of its accessors and mutators as well as default constructor. In addition this class should have the following methods:

- `public void insertFriendNode(StudentNode):` inserts a `FriendNode` into the tail of `FriendList`
- `public void deleteFriendNode(StudentNode):` deletes a `FriendNode` from the `FriendList`
- `public FriendNode findFriendByName(String):` Finds the `FriendNode Node` that references the `FriendNode` object that has references the `Friend` object with the value of the String parameter in `name` attribute. If the `Student` does not exist in the list, `null` is returned.
- `public void printFriendListByName() :` Displays the names of all the friends' in a friends list.

You must also modify the `deleteStudentNode()` method in `StudentList` to **delete all references to from the `FriendNode` objects in ALL** `FriendLists` **to the deleted** `StudentNode` **object.**

CSC 330 Project I Name: KOK

PART III (DUE 3/23/2016)

In this part you will implement the remaining command (Q,) by implementing a method in <u>`StudentNode`</u> class:

- `public boolean isFriends(String) :` Returns `true` if the `FriendList` attribute has a `FriendNode` object that references a `FriendNode` object which references a `Student` object with `name` attribute equal to the String parameter.

CLASSES TO IMPLEMENT FOR THE PROJECT

```
public class Person {
        private String name;
        private String gender;
        public Person()
        public Person(String n, String g)
        public void setName(String n)
        public String getName()
        public void setGender(String g)
        public String getGender()
}
public class Student extends Person {
        private String major;
        private double gpa;
        public Student()
        public Student(String nm)
        public Student(String nm, String gn, String m, double gp)
        public void setMajor(String m)
        public String getMajor()
        public void setGPA(double g)
        public double getGPA()
}
public class StudentNode {
        private Student student;
        private StudentNode Sptr;
        private FriendList Fptr;
        public StudentNode()
        public StudentNode(Student s)
        public void setStudent(Student s)
        public Student getStudent()
        public void setSptr( StudentNode s)
        public StudentNode getSptr()
        public void setFptr( FriendList f)
        public FriendList getFptr()
        public boolean isFriends(String fn)
        }
```

CSC 330 Project I Name:
KOK

```
public class StudentList {
        private StudentNode shead;
        public StudentList()
        public void setShead(StudentNode sh)
        public StudentNode getShead()
        public void insertStudentNode(StudentNode)
        public void deleteStudentNode(StudentNode)
        public StudentNode findStudentByName(String )
        public void printStudentListByName()
}

public class FriendNode {
        private StudentNode Sptr;
        private FriendNode Fptr;
        public FriendNode()
        public FriendNode(StudentNode s)
        public void setSptr( StudentNode s)
        public StudentNode getSptr()
        public void setFptr( FriendNode f)
        public FriendNode getFptr()
}
public class FriendList {
        private FriendNode fhead;
        public FriendList()
        public void setFhead(FriendNode f)
        public FriendNode getFhead()
        public void insertFriendNode(FriendNode fn)
        public void deleteFriendNode(FriendNode fn)
        public FriendNode findFriendByName(String fn)
        public void printFriendListByName()
}
```