Name: _____ Andrew Id: _____

15-121 Fall 2019 Quiz 7

Up to 25 minutes. No calculators, no notes, no books, no computers. Show your work!

This is one long, cohesive problem. You will need to read the code carefully in order to determine how it should work. Take time to read and think.

Please note that Page 2, in addition to giving you code to read, will also be the location where you write the answer to Question 3. (So don't write anything else on that page...)

Consider the following **three** Java classes provided for you:

```
public class QListTester {
    public static void main(String[] args) {
        // Create a new QList
        QList theList = new QList();

        // Add two students to it
        theList.addStudent(new Student("Ahmed", 1));
        theList.addStudent(new Student("Carol", 4));

        // Print the students. This outputs:
        // Ahmed[Freshman]
        // Carol[Senior]
        theList.printStudents();

        // Add a faculty member
        theList.addFaculty(new Faculty("Anis Charfi", "Associate Teaching Professor"));

        // Print out the faculty. This outputs:
        // Anis Charfi[Associate Teaching Professor]
        theList.printFaculty();
    }
}
```

```
public abstract class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public abstract String getInfo();
}
```

```java
/**
 * A data structure to store information about Students and Faculty. They are
 * stored in different ArrayLists for convenience, but QList as a whole should
 * be thought of as one data structure that contains both types of Persons.
 */
public class QList                                          {

    private ArrayList<Student> students = new ArrayList<Student>();
    private ArrayList<Faculty> faculty = new ArrayList<Faculty>();

    public void addStudent(Student s) {
        students.add(s);
    }

    public void addFaculty(Faculty f) {
        faculty.add(f);
    }

    public void printStudents() {
        for (Student s : students) {
            System.out.println(s.getInfo());
        }
    }

    public void printFaculty() {
        for (Faculty f : faculty) {
            System.out.println(f.getInfo());
        }
    }

}
// Note: Do not write anything on this page except your answer to Question 3.
```

1. (5 points) Write the `Faculty` class (which properly inherits from the `Person` class), so that the `QListTester` works properly.

2. (7.5 points) Write the `Student` class (which properly inherits from the `Person` class), so that the `QListTester` works properly.

3. (7.5 points) Consider the following interface as well as an alternative tester for `QList`:

```
public interface PersonDB {
    /**
     * Add a Person, p, to the data structure.
     *
     * @param p The person to be added.
     * @return true if the person was added and false otherwise
     */
    public boolean add(Person p);

    /**
     * Print all the people in the data structure.
     */
    public void printAll();
}
```

```
public class AltQListTester {
    public static void main(String[] args) {
        // Create a new QList
        QList theList = new QList();

        // Add some students and faculty to it in a variety of ways
        theList.add(new Student("Ahmed", 1));
        theList.addStudent(new Student("Carol", 4));
        theList.add(new Faculty("Anis Charfi", "Associate Teaching Professor"));
        theList.addFaculty(new Faculty("Ryan Riley", "Associate Teaching Professor"));

        // Print everyone in QList. This outputs:
        // Anis Charfi[Associate Teaching Professor]
        // Ryan Riley[Associate Teaching Professor]
        // Ahmed[Freshman]
        // Carol[Senior]
        theList.printAll();
    }
}
```

Modify the `QList` class on page 2 so that it implements the new interface and works properly with the new tester. **You should write your solution directly on to page 2**, modifying the class found there. You may not add any new instance variables to the class. (This means that your new `add()` method needs to determine which existing `ArrayList` to add the item to.)