Person Catalog

1 Introduction

This assignment will cover inheritance, simple polymorphism, and simple lists.

2 Problem Description

Students and Professors have completely different agendas; however, they both deal with a lot of the same day to day issues. In this homework you will implement multiple classes that inherit from their parent classes, so that the methods properly work with the driver that has been provided.

3 Solution Description

Driver.java (DO NOT CHANGE THIS FILE) will be provided in this homework assignment. The driver will handle all the command line input, as well as displaying the methods that you will actually implement. You will personally be creating the following items:

- PersonList.java
- Person.java
- Professor.java
- Student.java
- GraduateStudent.java
- UndergraduateStudent.java

Below are the specific requirements that you will have to meet for each class.

1. *PersonList*

- Instance variables: `Person[] people` and `int count`
- `PersonList(int maxSize)`: sets `people` to a max size
- `listPeople()`: lists the people inside of the `people`. Should NOT print `null` when indexing into null elements of the array.
- `add(Person p)`: If `people` has empty space, adds a `Person` to `people`. Do not worry about edge cases for this method.

2. *Person*

   - Instance variables: `String firstName` and `String lastName`.
   - `Person(String firstName, String lastName)`: sets `firstName` and `lastName`.
   - `String toString()`: returns `firstName` + `lastName`

3. *Professor*

   - This class extends `Person`
   - Instance variables: `int rateMyProfessorRating` and `double averageGPA`
   - `Professor(String firstName, String lastName, int rateMyProfessorRati double averageGPA)`: calls the constructor of its immediate superclass, and sets `rateMyProfessorRating` and `averageGPA`
   - `toString()`: returns `firstName` + `lastName` + `rateMyProfessorRating` + `averageGPA` (see examples for formatting)

4. *Student*

   - This class extends `Person`
   - Instance variables: `int intelligence` and `int motivation`
   - `Student(String firstName, String lastName, int intelligence, int motivation)`: calls the constructor of its immediate superclass, and sets `intelligence` and `motivation`
   - `toString()`: returns `firstName` + `lastName` + `intelligence` + `motivation` (see examples for formatting)

5. *GraduateStudent*

   - This class extends `Student`
   - `GraduateStudent(String firstName, String lastName, int intelligence, int motivation)`: calls the constructor of its immediate superclass
   - `toString()`: returns `firstName` + `lastName` + `intelligence` + `motivation` (see examples for formatting)

6. *UndergraduateStudent*

- This class extends `Student`
- `UndergraduateStudent(String firstName, String lastName, int intelligence, double motivation)`: calls the constructor of its immediate superclass
- `toString()`: returns `firstName` + `lastName` + `intelligence` + `motivation` (see examples for formatting)

4 Example Output

This is example code of adding a Person, a Student, a Professor, a UnderGraduateStudent, and a GraduateStudent.

```
$ java Driver
Welcome to the Person Catalog!

Which option would you like?
0. List the people
1. Add a person
2. Exit

0

Hi, my name is Bob Jones. My Rate My Professor rating is 4/5 and my average GPA is 5.0/4.00. I
    really wish students would stop emailing me so much.
Hi, my name is Bill Billson. My intelligence is 10/10 and my motivation is 10/10. I'm stressed
    out.
Hi, my name is Walter White. My intelligence is 10/10 and my motivation is 10/10. I'm going home
    this weekend to get laundry done; talk about clutch.
Hi, my name is John Appleseed. My intelligence is 5/10 and my motivation is 5/10. I'm stressed
    out AND broke.

Welcome to the Person Catalog!

Which option would you like?
0. List the people
1. Add a person
2. Exit

1

Which type of person would you like to add?
0. Professor
1. Student
2. Graduate Student
3. Undergraduate Student

1

Please enter their first name.
Joe

Please enter their last name.
Bob

Please enter their intelligence.
1

Please enter their motivation.
10
```

```
Welcome to the Person Catalog!

Which option would you like?
0. List the people
1. Add a person
2. Exit

0

Hi, my name is Bob Jones. My Rate My Professor rating is 4/5 and my average GPA is 5.0/4.00. I
    really wish students would stop emailing me so much.
Hi, my name is Bill Billson. My intelligence is 10/10 and my motivation is 10/10. I'm stressed
    out.
Hi, my name is Walter White. My intelligence is 10/10 and my motivation is 10/10. I'm going home
    this weekend to get laundry done; talk about clutch.
Hi, my name is John Appleseed. My intelligence is 5/10 and my motivation is 5/10. I'm stressed
    out AND broke.
Hi, my name is Joe Bob. My intelligence is 1/10 and my motivation is 10/10. I'm stressed out.

Welcome to the Person Catalog!

Which option would you like?
0. List the people
1. Add a person
2. Exit

2

$
```

5 Checkstyle

Review the Style Guide and download the Checkstyle jar and associated XML file. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
      2
```

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting
    audit..." | findstr /v "Audit done" | find /c /v "hashcode()"
0
```

> Food for thought: is there a one-liner like above that shows you only the number of errors? Hint: `man grep`.

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **20** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Tips

- Look at the driver provided (but don't edit it!) and see what methods are being called when the user selects certain options.
- Write out a diagram of the inheritance structure before you start the homework! Knowing how all the classes interact with each other will make the implementation much simpler!
- Think about the access modifiers that you know. "Protected" may come in handy.

7 Turn-in Procedure

You will need to turn in these items.

- PersonList.java
- Person.java
- Professor.java
- Student.java
- GraduateStudent.java
- UndergraduateStudent.java

Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, the `cs1331-checkstyle.xml` file, or the provided driver file. When you're ready, double-check that you have submitted and not just saved a draft.

8 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.

(b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

(c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!