AP Computer Science — Partner Project - GiftList

ASSIGNMENT OVERVIEW

In this assignment you'll be creating a small package of files which will allow a user to manage a simple GiftList. The package will include at least three files: `GiftEntry.java`, `GiftList.java`, and `GiftListTester.java`, all written by you and a partner.

This assignment is worth 50 points and is due on the *crashwhite.polytechnic.org* server at 23:59:59 on the date given in class.

BACKGROUND

Around the holiday season, keeping track of gifts you wish to give to people, and to make or buy for people, is part of the fun. This project provides you with the opportunity to practice your `Array` or `ArrayList` manipulation skills.

PROGRAM SPECIFICATION

Create a package of Java classes, that:

a. allows a user to identify a person, possibly a gift for that person, and whether or not the gift has been procured (`GiftEntry.java`).
b. allows a user to manipulate a list of these `GiftEntry` objects, including adding recipients to the list, deleting them from the list, editing the gift for a person, crossing a person off the shopping list, etc. (`GiftList.java`).
c. test the capabilities of the `GiftList` class by creating a `GiftList` object and manipulating it (`GiftListTester.java`).

DELIVERABLES

`GiftList.zip`

This single file will be a zipped directory (folder) of your BlueJ project. It will include as a minimum the three files listed above along with any other classes you create during the development of your program, and a `package.BlueJ` file.

To submit your assignment for grading, copy your file to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- This project is partly a design challenge, and partly an implementation challenge. Three classes have been identified for you in the statement of this problem, but you may choose to consider the GiftList project from a different design perspective.

- That being said, it's important that you and your partner produce some working code relatively quickly.

- Your `GiftList` and `GiftEntry` classes need to be able to handle a number of different common

"gift list scenarios," and your Tester will need to demonstrate that your classes can handle these situations:

- adding a person to my gift list with no gift yet specified
- adding a person to my gift list with a gift specified
- editing the gift that I've specified for a person
- checking a person off once I've gotten a gift for them (but keeping that entry on the list)
- deleting a person from my gift list if no longer needed (removing the entry entirely from the list)
- displaying my complete list of recipients/gifts and checkoff status
- displaying a "shopping list" of only the recipients for which I haven't yet gotten their gift

- You are encouraged to get a basic working version of this package up-and-running *without* the addition of any "feature creep." Attempting to add additional features before you've gotten the main package working will overly complicate the assignment.

- What additional features might you consider, after completing the basic project? Two of the most obvious design considerations might be the fact that
  - a `GiftEntry` actually consists of two items that could be identified as subclasses in their own right: a `Recipient` and a `Gift`.
  - a `GiftList` only lasts for the life of the program. There isn't yet a convenient way of storing the list (in a text file? in a database?) so that we have long-term access to our Gift List.

  Neither of these design considerations is intended to be addressed in this assignment. If you choose to explore them, don't submit them as part of your official project package for this assignment.

- Your `GiftListTester` will include a main program that creates a `GiftList` and manipulates it, with multiple print statements indicating what is being done to the `GiftList`, what the results are of those manipulations, and what the expected results are. See the Sample Interactions at the end of this sheet for an example of what that might look like. (Your list doesn't need to look exactly like this—this is just an example of one format.)

GETTING STARTED

1. With paper and pencil, and in collaboration with a partner, identify what the main components are that you'll need to include in your program.

2. Give some thought to "real world" process of creating and manipulating a Gift List. Consider what classes you'll want to include for your project, and what features those classes might need to include.

3. Sketch out the basic features of each class, including instance variables, constructors, accessor methods, and mutator methods.

4. Consider writing some pseudocode that you can use to begin implementing those classes.

5. Create a new project in BlueJ that will allow you to manage this assignment.

6. Begin writing either at the lowest level (the `GiftEntry` class) or the highest level (the

`GiftListTester`), and test each bit of code as you go, making sure that one piece works before you proceed on to the next section. You'll repeatedly run through this edit-compile-test, edit-compile-test process to progressively find bugs and fix them *while* you're writing your program, not afterwards.

7. Save your program from time to time, and make backups.

8. Once a day or so, archive/zip your BlueJ GiftList folder and save a backup copy of it on another device or machine: a flash drive, your home folder on the *crashwhite.polytechnic.org* server, etc.

9. When your program is completed (but before the deadline), copy a final archived package (`GiftList.zip`) to the server as indicated above.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. Which part of this assignment is the more challenging for you: the design or the coding?

2. How much were you able to anticipate the methods needed for this project ("You know, we need a way to tell the GiftList that an entry has been completed"), and how much did you discover as you were going through writing the code?

3. How useful or productive did you find it, being able to work on this assignment with someone else? Did you "divide and conquer" the tasks that needed to be done, or did you work together the entire time, watching over each other's shoulders? Take a look at the Wikipedia entry on Pair Programming at *https://en.wikipedia.org/wiki/Pair_programming* . Were any of the strengths of that system evident in your work on this project?

SAMPLE INTERACTIONS

```
Initializing Holiday List
Adding Jill to list, no gift
Adding Craig to list, frisbee for gift
Adding Robin to list, no gift

Printing out full list:
Jill--not done
Craig-frisbee-not done
Robin--not done

Expected:
Jill--not done
Craig-frisbee-not done
Robin--not done

Test 1 passed

Setting gift for Robin, then printing out full list

Jill--not done
Craig-frisbee-not done
Robin-yoga mat-not done
```

```
Expected:
Jill--not done
Craig-frisbee-not done
Robin-yoga mat-not done

Test 2 passed

Buying Robin's gift and checking it off list
Jill--not done
Craig-frisbee-not done
Robin-yoga mat-done

Expected:
Jill--not done
Craig-frisbee-not done
Robin-yoga mat-done

Test 3 passed

Craig has been naughty--removing him from list
Jill--not done
Robin-yoga mat-done

Expected:
Jill--not done
Robin-yoga mat-done

Test 4 passed

Printing out just the shopping list
Jill-

Expected:
Jill-

Test 5 passed
Tests passed: 5/5
```