PPA Assignment 11

Overview

Do you think there is a problem with any of the content below? Let us know immediately at programming@kcl.ac.uk.

Read through this brief carefully before starting your attempted solution. Also ensure that you comment your code.

You are not advised to print this assignment, but instead to always access the latest version of this brief through KEATS, which will evolve with minor clarifications and corrections throughout the assessment period. Students will be notified of any major modifications to the brief by email.

A partner from your lab session should be selected for this piece of work at the next available opportunity, typically your next lab session. You must not complete any of the assignment below without your chosen partner present, as doing so is likely to jeopardise your grade.

Aims

The aims of this piece of coursework are as follows:

- To understand the relationship between the Law of Demeter, coupling and encapsulation;

Once completed, both you and your partner must submit your assignment using the link marked `Assignment 11: Nexus Submission Link' on KEATS. However, this is not enough to receive a mark for this piece of work. You must also attend the lab session following your submission, so that one of the teaching assistants can mark your work with you present, and ask you detailed questions about it. Revisit the `Lab Assessment and Pair Programming Q&A' guide on KEATS for more information.

Any submitted code or documentation that is found to be unduly similar to the code or documentation submitted by any other pair(s) of students, will result in a penalty for those involved.

Provisional marks for your code will be released on KEATS within one week of the final lab assessment. Final assignment grades will be submitted to the exam board at the end of the semester.

For all other queries, see the Support section on KEATS, specifically the `Lab

- To use packages and your chosen IDE for the first time, if you have not already done so;
- To allow you to experience pair programming for the first time, if you have not previously undertaken the practice;
- To revisit object-oriented concepts from the first semester.

Assessment and Pair Programming Q&A' guide.

Assessed Preliminaries

These following tasks are mostly explorative, and are designed to prepare you for the assessed tasks that follow. These preliminary tasks should be completed with your partner prior to the assessed tasks and will be checked as a part of your completeness mark by your examiner during your assessed lab:

1. Familiarise yourself with an Integrated Development Environment (IDE), preferably Eclipse, by locating the appropriate software on the lab machine you and your partner are using, or by downloading the appropriate software to one of your own machines. Going forward, both you and your partner should have access to an IDE individually. Then, ensure you know how to do *at least* the following with your chosen IDE:

    1. Create new, self-contained Java projects, that can be executed from within the IDE by running the class containing the main method.
    2. Create new packages.

2. Create a new project in your chosen IDE, called `assignment11', and within this project create a package called `preliminaries'.

3. We are now going to briefly revisit one of the assignments from the previous term, in order to improve its structure based upon the knowledge accrued towards the end of last term, and the start of this one. So, copy all of the code

from CW4 (related to exams and marking) into the preliminaries package of your assignment11 project. You are advised to use the code associated with the recommended solution to CW4, which you will find in the revision bundle on KEATS.

1. Firstly, you should introduce an appropriate inheritance structure into this code in order to avoid code redundancy.
    - Most of this process was shown to you as a live demo when covering Topic 9 last term, so this should be a straightforward modification.
    - Your examiner may ask you how you have reduced redundancy in the original solution to the assignment using inheritance.
2. Next, consider the Law of Demeter, discussed on Slide 7 of `Reviewing Object Orientation', and then conduct the following explorative exercise:
    - From the latest version of a solution to CW4 that you now have, identify the first place in which the Law of Demeter is broken.
    - At the time CW4 was completed it made sense to break this law, as it allowed us to explore the capabilities offered by a series of method calls on different objects. However, as we are now thinking more carefully about the structure of our programs, we ought to consider what the effect of these linked method calls are (i.e. the effects of violating the Law of Demeter) in respect of maintaining our program.
    - For example, consider the method call that constitutes the first violation of Demeter's law, as already identified. Try changing the name of this method. You should find that doing so produces a number of errors in your program, requiring the programmer to rename calls to this method in a number of different places.
    - Change the structure of the original CW4 solution so that Demeter's Law is no longer violated in the single example you have identified, and these errors are thus reduced to a single error. There is no requirement to remove all violations of Demeter's Law in your program as this is just an explorative exercise.
    - Investigate the terms *tightly coupled* and *loosely coupled*, and how these describe the initial solution to CW4, and would describe a solution with no violations of

Demeter's Law, respectively. You should be able to discuss these terms with your examiner, if asked.

- Consider the relationship between *encapsulation* and loose coupling. You should be able to discuss this relationship with your examiner, if asked.
- Consider the *drawbacks* of adhering to Demeter's Law (i.e. not breaking this law). You should be able to discuss these with your examiner, if asked.

3. You are advised to make a test submission to Nexus as early as possible, to ensure you are able to efficiently extract the source files generated by your chosen IDE. You should retain your package structure when submitting, so typically this will involve submitting everything from the *source* folder of the assignment11 project in the file system directory structure generated by your chosen IDE (not your entire project folder).

Assessed Task

To practise writing code with an understanding of the Law of Demeter (and coupling), and to review several object-oriented concepts, consider the following exercise, the completion of which contributes towards both your completion and quality marks:

Consider the fictional children's tale of the Baudelaires, three children who need to be assigned to a new guardian, who is a family relative, after A Series of Unfortunate Events. The task of assigning the children to this guardian falls to a banker, Mr Poe, who is entrusted with the well-being of the children. Write a program to help Mr Poe make this selection, based on a number of different factors relating to the relatives, in this fictional scenario.

Below are further details of the problem:

1. Each relative of the Baudelaires is a person, who has a name and a level of friendliness.

2. Mr Poe is a banker; a type of person, with a friendliness of 5.

3. In his possession, Mr Poe, as a banker, has a list of relatives, which he insists on keeping in alphabetical order (for all current and any future relatives added to the list), and their geographical distance from Mr Poe's home, where the Baudelaires are staying temporarily.

4. The list of relatives, and the distance of their homes from Mr Poe's, is as follows: Josephine, 100 miles; Olaf, 10 miles; Sir, 20 miles; Uncle Monty, 20 miles.

5. Mr Poe wishes to select the relative that lives closest to his home (incorrectly believing that this is the definition of `closest relative') as the new guardian for the Baudelaires. When he has done so, the children are moved to live with this relative, as indicated by a record Mr Poe keeps of the person with whom the Baudelaires are currently staying.

6. However, not all guardians are friendly, and this only becomes apparent once the Baudelaires have moved to live with a relative.

7. The friendliness of the Baudelaire's relatives is as follows: Josephine, friendliness 5; Olaf, friendliness -10; Sir, friendliness 0; Uncle Monty, friendliness 10; .

8. Mr Poe judges a relative to be unfriendly if their level of friendliness is strictly less than his own. If Mr Poe identifies a guardian as unfriendly, he removes them from the list of relatives, and moves the children to their next geographically closest relative.

9. The process continues until the closest, friendly relative is found for the children to live with.

You should also consider the following:

1. Key events in the program: the children being moved to live with a guardian, a guardian being unfriendly, the children subsequently being moved to a new guardian, and finally reaching the closest friendly guardian, should be indicated using print statements.

2. It should be possible to derive suitable string representations of each object, which displays all of the information in the fields of that object.

3. Be sure not to break Demeter's Law in your solution (marks will be lost if you do). But think carefully about the impact this has on your solution. Your examiner may ask you about this.

The classes that contain your solution for this assessed task should be placed within a new package called `assessment', within your assignment11 project.

Optional Tasks

The following tasks will not be directly assessed in order to determine your quality and completeness marks, but naturally your examiner will be impressed to see that they have been completed.

1. JavaDoc tags

- You should already be familiar with how to add JavaDoc (method) comments into your code. If you wish to refresh your memory, review the slides from last semester (particularly Topic 5) or this link may be of use.
- However, you are perhaps less familiar with the use of JavaDoc *tags*, and the effect that adding these into your comments has on the HTML pages that are

generated from JavaDoc comments. Read about JavaDoc tags here, and then try adding some into your code at appropriate places.

- To test the tags you have added, you should consider how your chosen IDE can generate HTML pages, representing your JavaDoc, for you, as was done from the command line in Topic 5 in the first semester. Several basic guides to do this in Eclipse are available here and here.

- Finally, you should consider the capabilities that your chosen IDE has to generate JavaDoc comments (including tags) for you.

2. A command line interface

- Currently details of the relatives (their names, geographical distance from Mr Poe's house and their friendliness) should be hardcoded into your solution.

- You might like to make a small command line interface that allows a user to add *additional* relatives to this information dynamically, using standard input. The user should then be able to issue a command to calculate the closest, friendliest relative, once they have finished adding new relatives to the list.

- It is essential that any modifications you make for this command line interface does not interfere with the existing functionality of your classes.

3. Removing the main class

- In your current solution you are likely to have a class that houses just the main method.

- What would happen if you were to delete this class? Where would the main method, and the commands it contains, be placed? Try out several different locations for the main method, and consider how these locations make sense conceptually.