CO320 Assignment 3 (Version 2018.12.04)

Introduction

This assignment is designed to help you explore the subject of *inheritance* in object-oriented languages. In particular, you will be exploring how shared characteristics of related classes can be represented in a *superclass*, with the specialised elements of the related classes represented in multiple descendent *subclasses*.

You will also find the starting project has some interesting examples of Java API and language features that are worth studying to deepen your knowledge of the course material.

Download the starting project from:

https://www.cs.kent.ac.uk/~djb/co320/LOCAL-ONLY/assign3.zip

Date set: Tue 4th December 2018
Deadline: 23:55 Mon 14th January 2019
Weighting: 20% of the module's coursework mark.

Please make an effort to read through the assignment as soon as possible so that you know what you need to understand to complete it, and can think about how best to organise your time.

Plagiarism and Duplication of Material

The work you submit must be your own. We will run checks on all submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism.
Some guidelines on avoiding plagiarism:

- One of the most common reasons for programming plagiarism is leaving work until the last minute. Avoid this by making sure that you know what you have to do (that is not necessarily the same as how to do it) as soon as an assessment is set. Then decide what you will need to do in order to complete the assignment. This will typically involve doing some background Taxi and programming practice. If in doubt about what is required, ask a member of the course team.

- Another common reason is working too closely with one or more other students on the course. *Do not* program together with someone else, by which I mean do not work together at a single PC, or side by side, typing in more or less the same code. By all means *discuss* parts of an assignment, but do not thereby end up submitting the same code.

- It is not acceptable to submit code that differs only in the comments and variable names, for instance. It is very easy for us to detect when this has been done and we will check for it.
- **Never** let someone else have a copy of your code, no matter how desperate they are. Always advise someone in this position to seek help from their class supervisor or lecturer. Otherwise they will never properly learn for themselves.
- It is not acceptable to post assignments on sites such as Freelancer and we treat such actions as evidence of attempted plagiarism, regardless of whether or not work is payed for.

Further advice on plagiarism and collaboration is also available.
You are reminded of the rules about plagiarism that can be found in the Stage I Handbook. These rules apply to programming assignments. We reserve the right to apply checks to programs submitted for assignment in order to guard against plagiarism and to use programs submitted to test and refine our plagiarism detection methods both during the course and in the future.

The task

The scenario is a surgery where patients are assigned to a patient list to be seen by a doctor. To work with the project, create a Doctor object, a PatientList object and several Patient objects. Add the patients to the list and assign the doctor to the list. You might like to create your own test classes and test fixtures to make this process easier over the course of the assessment. See Chapter 9 of the course text book for further details of how to set up test classes.

A further refactoring will involve the addition of a Nurse class as someone who can be assigned to manage a patient list.

Correcting the printList method (5 marks)

There is an error in the current version of the printList method of PatientList. To illustrate this, create a PatientList object and call printList straight away without adding any patients or a doctor to it.

Correct the error by checking for a null doctor in printList. If there is no doctor then print "unknown" in place of the doctor's name. Aim to code with in a way that avoids a lot of repetitious code.

Test your changes by printing a list of patients both with and without a doctor. Make sure that room details are still printed even if there is no doctor.

Introducing inheritance into the project (20 marks)

The Patient and Doctor classes share some common attributes – both have a name and a title. They also have some common methods – getName and getTitle. Capture

these common elements in a new class, Person, that becomes the *superclass* of both Patient and Doctor.

This change involves placing the common fields and methods into Person and removing them from Patient and Doctor. Rather than making the changes all in one go, you might find it safer to move one field at a time rather than all in one go.

Introducing a new method (5 marks)

It must be possible to change the titles of patients and doctors. Add a method, changeTitle, that makes this possible. Make appropriate tests to check that this method works as it should.

Replacing print in Patient (10 marks)

We wish to remove printing to the terminal from the Patient class and do it elsewhere.

Replace the print method in Patient with a toString method with the following header:

```
/**
 * Return a string in the form "Title Name (ID)"
 * if a title is present, or
 * "Name (ID)" if there is no title.
 * @return A string containing
 *           the patient's title (optional), name and ID number.
 */
public String toString()
```

The toString method must return a String that is almost identical with what would have been printed by the print method. For instance, if the patient "Claude Monet" has the id "Clau00001" and no title, then this method will return:

```
"Claude Monet (Clau00001)"
```

but if his title is `"Mr"` it will return:

```
"Mr Claude Monet (Clau00001)"
```

Note that there must be no leading space in the string if a patient has no title. Note, too, that there must be no trailing newline (\n) character in the string that is returned.

In removing the print method from Patient, you will have to change the printList method of PatientList. The printList method must now print the string returned by a patient's toString method.

Carry out tests to ensure that your changes have not broken anything.

Introducing polymorphism (20 marks)

We wish to add a new class, Nurse, to the project. You can download it from:

https://www.cs.kent.ac.uk/~djb/co320/LOCAL-ONLY/Nurse.java

Save a copy of Nurse.java in your project folder and ensure that it compiles with your existing classes.

Nurse is very similar in content to the original Doctor class. It was written before common elements of patients and doctors were moved to a Person class. As a first step, refactor Nurse so that it is also a subclass of Person.

A list of patients could be waiting to see a nurse rather than a doctor. Rather than having different patient list classes for nurses and doctors, the PatientList class must be refactored so that the field currently called `doctor` is renamed to `medic` and is able to refer to either a Doctor object or a Nurse object. However, it must not be able to refer to a Patient object.

Make all the changes necessary to the project to accommodate a medic field. *Hint:* this might include adding a further class. Think carefully about where common elements must be placed. Also, be sure to change all references from *doctor* to *medic* in PatientList.

Finally

Before you submit your assessment, thoroughly test the whole class to make sure that nothing you added later has broken anything added earlier. If you are unable to complete the class – even if you cannot get it to compile – still submit what you have done because it is likely that you will get at least some credit for it.

Aim to write your code neatly and follow the style guidelines that have been used throughout the course.

David Barnes
2018.12.04