COMP1161

LAB5

PREAMBLE

This lab aims to provide experience in the use of exceptions and input/output, and also demonstrates a few concepts related to polymorphism. It specifically focuses on the following skills:

1. Writing and using try and catch blocks for specific exception classes
2. Reading input from the keyboard and from files.
3. Implementing polymorphism via overriding

THE PROBLEM

Vaccination of the population is a vital activity that needs to be managed carefully. As a result, the Ministry of Health in the island of Jamrock plans to release Vaccinator Prime within a week. The system operates on three main inputs, namely,

1. A list of persons that request the vaccination service

2. A list of approvals, each the id of the person who has been approved, and which may include background medical information includes comorbidities (A comorbidity describes a known additional illness of a person)

3. A list of vaccine batches, each with the name of the batch, the number of individuals that can be served by the batch, a preference rating for the batch, and a list of contraindications. A contraindication describes a reason not to take a medication. Generally , medical professionals try to avoid giving medications with contraindications, or prepare for advanced management if options are limited.

Vaccination is then applied to the population in the following order:

1. vaccines are given in the order of the preference rating of vaccines.

2. While processing a batch of vaccines, the approved persons are processed in reverse order of age (ie. oldest first). Anyone with a comorbidity that matches a contraindication will not be given a vaccine.

(Disclaimer-any medical information gleaned from this project, and represented in the test cases should be regarded as pure fiction. The model presented here also ignores certain details that may be relevant in a real deployment, such as the need for multiple doses of certain vaccines)

Please note that a total of 10 marks are allocated to the lab. Lab-techs will use the 5 marks earned from the autograder as a guide to functionality, and the other 5 marks are earned from an assessment of understanding.

MARKING CRITERIA (PLEASE REVIEW BEFORE STARTING)

Criterion	Mark(s)
Explanation of the use of the exception hierarchy in software	1
Explanation of techniques for file/screen I/O	1
Explanation of why method findPerson needed a wildcard (ie. ?)	1
Explanation of why sort orderings affect logic	1
Coding style and readability.	1

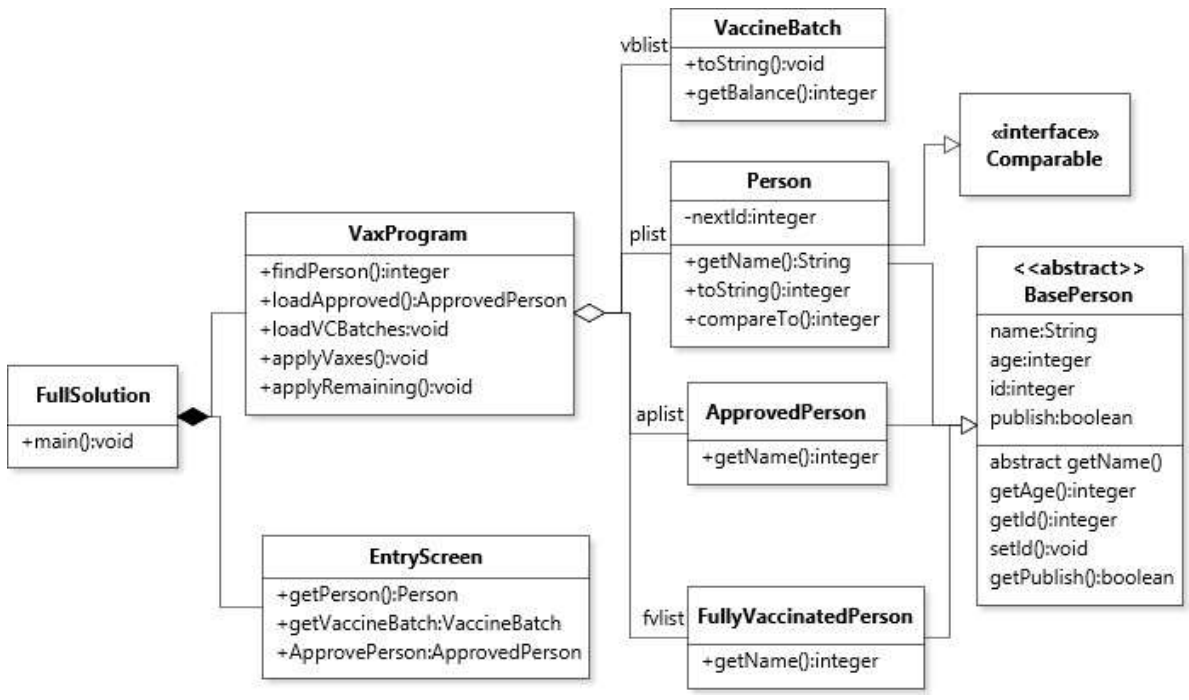

Figure 1: Simplified UML Class Diagram for Vaccinator Prime

Figure 1 depicts a simplified model of the classes that are needed to solve this problem. The starting code includes classes that will be run locally on your system, or in the REPL. The starting code implements the following functionality:

1. Allow the user to indicate whether interactive or test case execution is to be used.
    a. If running test case execution, the system will execute test cases, if a local id number has been saved.
    b. If interactive mode is selected, the program allows you to either prime the system by specifying a test case to load(six test cases have been provided), or go directly into data entry mode. Once in data entry mode, the user has to option to enter data, show the data, enter an ID number and print a report that can be read as a web page (which includes the list of vaccine batches and the list of persons who have agreed to publish)- Note the report location (which is printed by the application) allows you to see user directory for the application. .
2. Note the system includes a class named TestCase, however there is no need for you to interact with it. You should not modify it.

Your specific tasks to complete the system include :

1. Completing the menu in FullSolution to allow tests to be run
2. Writing the logic for the findPerson method in VaxProgram
3. Completing the loadApproved method in VaxProgram
4. Write logic for the loadVCBatches method in VaxProgram
5. Write the getVaccineBatch method in EntryScreen
6. Write the toString methods in ApprovedPerson and FullyVaccinatedPerson
7. Write the publish method in FullyVaccinatedPerson
8. Ensure appropriate sort orders are in place for VaccineBatch, ApprovedPerson and FullyVaccinatedPerson.

LAB EXERCISES

1 Allow the system to run test cases
   a. Adjust the system so that if the user selects 'n' , the system will execute the method TestCase.runTestCases().
   b. Modify the main method of full solution so that the submenu also includes a submenu **Run [T]est Cases**. On entering either T or t, the system should execute TestCase.runTestCases().

2. Complete the findPerson method in class VaxProgram so that it accepts an id that could belong to an instance of a derived class of BasePerson, and searches through an arraylist an instance that matched the id. If found, foundPerson returns the position at which the person was located. If not found, foundPerson returns -1.

3. Ensure approvals are operational
   a. Adjust the method loadApproved in class VaxProgram so that it is able to read data from a file, the name of specified as an argument. The logic of the method is intact, however the file handling operations are missing. You will need to implement appropriate error handling.
   b. In ApprovedPersons, update the overridden method getName so that it returns names in the format **lastname, firstname**
   Hint… to get parts of a name, you can use syntax like- String[] nameparts = name.split(" ");
   c. Complete the toString() method for ApprovedPerson so it follows the format
   getId()+"\t"+getName()+"\t\t"+getComorbids()

4. Allow vaccine batches to be properly entered
   a. Write the getVaccineBatch method in class EntryScreen.
   b. Write logic for the method loadVCBatches in class VaxProgram. As a design decision, you are instructed not to include exception handling inside of loadVCBatches. You are to throw exceptions form that method instead.
   c. Write the toString method of VaccineBatches to match the format:
   getName()+"\t"+getSize()+"\t"+getBalance()+"\t"+getPreference()+"\t"+getContras

5. Set ordering of vaccine delivery to meet specifications
   a. Write the toString method of FullyVaccinatedPerson to match the template
   getId()+(getPublish()?"*":"")+"\t"+getName()+"\t\t"+getVaxName()
   b. Verify sort orderings in the applyVaxes and applyRemaining methods of VaxProgram to meet the specs
      i. Vaccines are applied in order of preference (highest first)
      ii. Vaccines are given to approved persons in order of age(oldest first) (you can read to verify logic that expects vblist to be sorted by preference, and aplist to be sorted by age. fvlist should be sorted in alphabetical order of name for consistent reporting.)

6. Write the publish method of fully vaccinated person so that it returns the following code ONLY IF the associated person has agreed to publish(ie. getPublish() returns true):
   "<p>"+getName() + " took the " +vaxname + " vaccine!!!</p>"