COMP1161

LAB4

PREAMBLE

This lab aims to provide experience in the use of abstract classes and interfaces. It specifically focuses on the following skills:

1. Writing and using interfaces and abstract methods
2. Completing concrete class implementations
3. Applying the Comparable and Comparator interfaces

THE PROBLEM

The Ministry of Labour plans to upgrade it's System for Monitoring Labour (SML), while tracking earnings. They are particular interested in the activity of consultants (who may be either local or expatriate) as well as (local) nine-to-five employees (SML protocols currently do not support 9-5 expatriate positions). All local workers are expected to be citizens who have a TRN. All expatriates are expected to be registered and have an active work permit. All consultants are expected to earn from a skill. The SML data architect decides to model all potential workers as persons, with the knowledge that details like date representations and payment calculations will vary across groups. Your job during this lab is to assist the architect by writing one interface and three classes, as well as making adjustments to other classes.

Please note that a total of 10 marks are allocated to the lab. 5 marks will be earned via hackerrank, and the other 5 marks are earned through assessment from lab-techs.

Hackerrank link @ www.hackerrank.com/comp1161-lab4-23

MARKING CRITERIA (PLEASE REVIEW BEFORE STARTING)

Criterion	**Mark(s)**
Ability fully explain the definition and use of abstract classes and interfaces.	1
Demonstrable understanding the process of populating an instance of the superclass.	1
Knowledge of the effect of using the comparable and comparator interfaces	1
Ability to explain process of implementing customized orderings, including changing from ascending to descending order, and primary/secondary orderings	1

Coding style and readability.	1
A PENALTY OF 1 MARK WILL BE APPLIED FOR ANY OF THE FOLLOWING: - Ignoring specified interfaces or base classes based on the UML diagram (See Figure 1)	

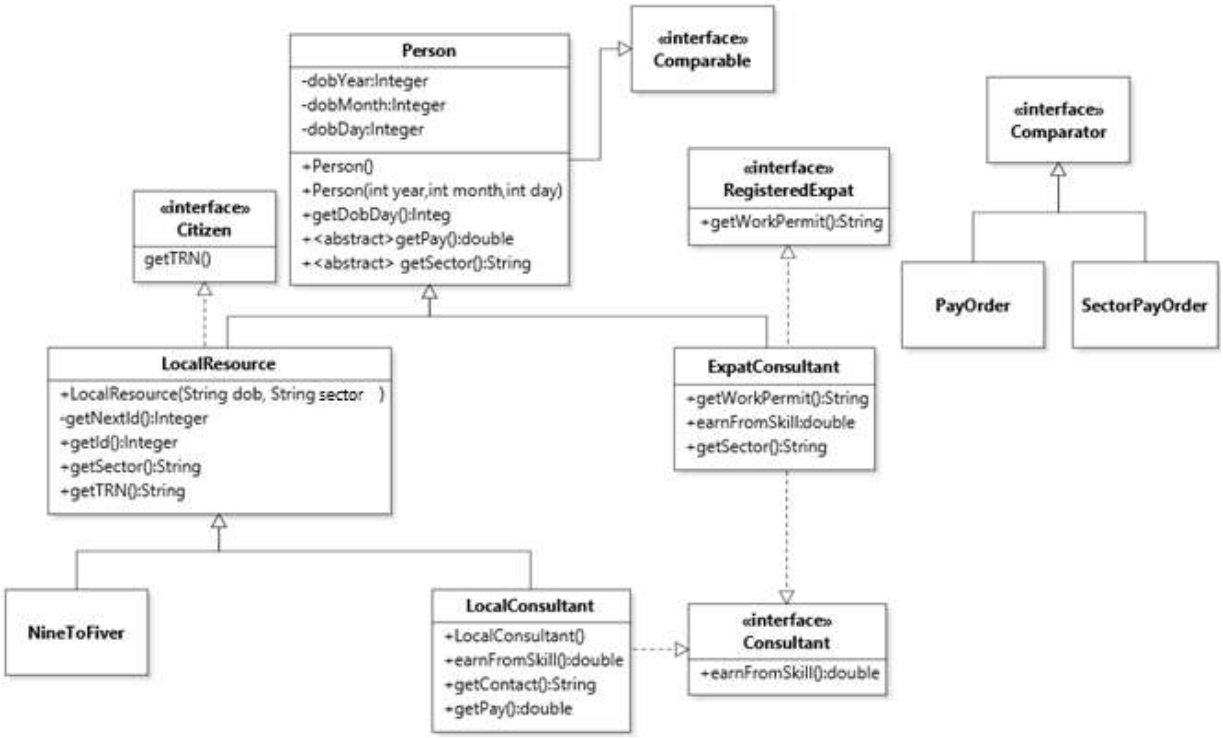

Figure 1: Simplified UML Class Diagram for SML

Figure 1 depicts a simplified model of the classes that are needed to solve this problem. Your specific tasks include creating the **RegisteredExpat** interface, writing the **LocalResource**, **LocalConsultant** and **SectorPayOrder** classes, as well as making minor edits to other classes to demonstrate functionality. Note that to reduce risk of crashes you are required to uncomment sections of code that refer to yet unwritten classes, as classes are written.

LAB EXERCISES

1. Create interface RegisteredExpat. The interface should define one method named getWorkPermit that accepts no arguments, and returns a string.
2. Modify the ExpatConsultant class to ensure it correctly implements RegisteredExpat along with any other interface it currently implements. The work permit number of an expatriate consultant is obtained by joining the string “WP00” with the id number of the expatriate consultant.

3. Write class LocalResource which extends the Person class. LocalResource exposes the following public methods:

    a. LocalResource(String date, String sector)- Actions of the constructor include accepting a date in the format "dd/mm/yyyy", initializing the base class, storing the sector and storing an id as a consecutively increasing integer. Note that the constructor must ensure that the date of birth information is recorded.
    b. getId():Integer - returns the ID of the current instance of LocalResource
    c. getSector():String – returns the sector associated with the current instance of localResource
    d. getTRN():String – returns the trn number of the current instance of LocalResource. The process used to determine the TRN is to add the id to the number 100000000 and then returning the string equivalent.
    e. Update the NineToFiver class to ensure it properly extends the LocalResource class
    f. In method getContact() of NineToFiver, remove the comments so that the method returns "Local Employee #"+and the id of the contact.

4. Write a concrete class LocalConsultant that extends LocalResource, and implements Citizen and Consultant. LocalConsultant exposes the following public methods:
    a. LocalConsultant(String dob, String sector, double skillPrice, double taxRate)
        - Saves local instance data – populates superclass data and calculates and saves a permit tax for the instance with a value given by taxRate*skillPrice.
    b. earnFromSkill():double- return the skillPrice for the instance
    c. getContact():String – return the string value obtained by joining the text "LocalConsultant#" with the id associated with the person
    d. getPay():double – returns the value obtained by subtracting the permit tax from the value earned from the skill.

5. Assume that all paychecks are ready on the day of birth of the worker (ie all people born on the first of a month get paid on the first of a month, people born on the second get paid on the second, and so on). The SML managers want to track the cash flow in the system. Modify the SML driver class at the comment that is labeled "//QUESTION 5" to include the code that will ensure the person list is sorted in ascending order of the day of birth.

6. Write a comparator for Person named SectorPayOrder that will allow a primary sort in ascending order of sector, and a secondary sort in ascending order of payrate where sectors match.