**Homework 8: Students of a College**

1 Objective

This project will give you an opportunity to use many of the concepts you’ve learned throughout the quarter. It will include the familiar, like objects, and a new thing like storing user-defined objects in an ArrayList, creating abstract classes, and implementing interfaces. After this project, you will have gotten a good survey of object-oriented programming and its potential. Note that this project uses one Interface and one abstract classes.

*2* What to Create

You are writing the application that deals with information about Pearson at College.

2.1 Enum classes

You should create two enumeration classes **Sex** and **Level**.

The enumeration class Sex has only two elements: MALE and FEMALE.

The enumeration class Level has following elements: ESL1A, ESL1B, ESL2A, ESL2B, ESL3A, ESL3B, ESL4A, ESL4B, ESL5A, ESL5B, ESL6A, ESL6B.

2.2 Interfaces and Classes

For every class, you should decide which private variables are needed. Also, don’t forget about toString() method.

1 Write an interface **Person** that requires the implementation of typical features that return a **First Name**, **Last Name**, **Id**, and **Sex** of person.

2 Write an **abstract class Students**. The constructor of this class gets parameters **id**, **first name**, **last name**, **sex**, and **month**, **day**, **year** of birth. In this class, you should realize all methods of the interface. Also, this class includes one abstract method **getPrice**(), and methods that set and return credits of a student. A ESL student has every time 5 credits.

3 Write the **ESL class** that is a subclass of **Student.** The **getPrice**() method returns $25.0 every time. The constructor of this class gets parameters: **id**, **first** **name**, **last** **name**, **day**, **month**, **year**, **sex**, **level**.

4 Write the **ESLStudents class.** This class must contain ArrayList of ESL students and have methods which let to add a student, remove a student with a specific index, return numbers of students, return ESL student that has a specific index. Also, this class must have a method that called **isFound**(). This method gets the id of student and returns the index of this student or -1 if this id doesn't exist. Besides, you should realize a method that gets level and returns how many students have this level.

5 Write the **College class** that is a subclass **of Student.** The **getPrice**() method returns a value equals the cost of the one credit multiply by the number of credits.

6 Write the **CollegeStudents class.** This class must contain ArrayList of College students and have methods which let to add a student, remove a student with a specific index, return numbers of students, return College student that has a specific index. Also, this class must have a method that called **isFound**(). This method gets the **id** of student and returns the index of this student or -1 if this id doesn't exist. Besides, you should realize a method that returns how many full-time students there are. A full-time student must have 13 or more credits.

2.3 **toString**() method

At the topmost class, create a toString() method that supplies all the information it knows about the students. In each subclass that adds information of interest, concatenate that information to the string created by its superclass.

*3 General*

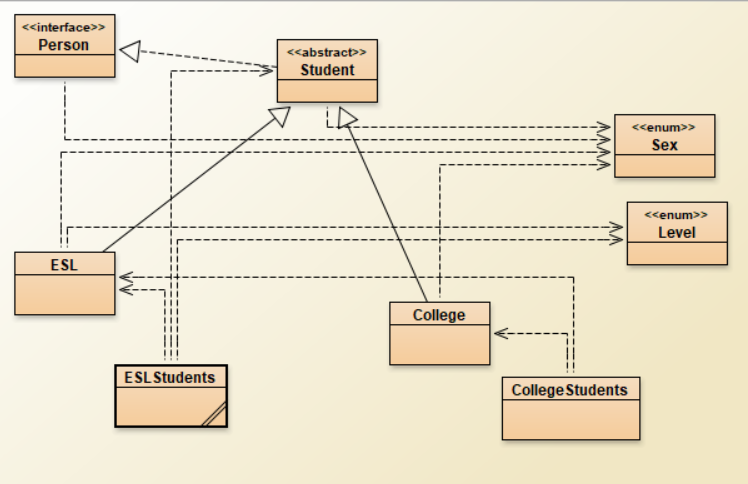
• We don’t need a variety of constructors here; create constructors that include everything needed.

• You know enough that all your UI should guard against bad data types and out-of-range data; handle these gracefully with positive user interactions. Let the user try again if they make a mistake.

• Provide preconditions wherever they make sense and help guard against data from outside the class that might cause crashes or other bad behavior; throw appropriate exceptions in those cases.

• All classes should have toString methods with an attractive output.

*4 UML Class Diagram*



5 Hints

• Pay close attention to the difference between IS A relationship (inheritance) and HAS A relationships (instantiation and use); use the right one here. We’ve done projects with both types of relationships this quarter.

• Build from the ground up. Write the interface first. Then code Student, writing and running the test code for each method you add. Once you’ve shored up that class, move onto ESLStudent. Again, incrementally develop and test. Once that’s solid, move to the next branch.

*6 Testing*

Create test methods for **ESL, ESLStudents, College, CollegeStudents** classes.