CST8132 Object-Oriented Programming

Lab 3: Inheritance, Abstract Class, Interface - College System

Due Date: Check Brightspace for the due dates

Marks: 20 marks (worth 7% of term mark)

Demo: Demo your code and output to your lab professor during your own lab hours.

Recommended Reading: Chapter 9 & 10 of Deitel and Deitel, Java How to Program book

Exercise

In this lab, we create a few classes to create a college system. A college (name has to be entered by the user) has students registered with them (number of students has to be entered by the user). Each student has common attributes of a person (first name, last name, email, and phone number) and a gpa, which is calculated based on the marks obtained in various courses (number of courses can vary). We have fulltime students and part time students, each of them has different attributes. We have certain policies which is stored in the interface named Policies. The classes that you need to create are Person, Student, FulltimeStudent, ParttimeStudent, College and an interface named Policies. Policies will have two constants – maxMarks as 100 and maxGpa as 4.0. This system will ask the user about the college name and the number of students. Then, the system will read information of each student. After reading, it will calculate the GPA. Once the reading and calculations are done, it will print the full information of all students including the GPA. Before reading further, you need to draw the UML of this system to create a rough plan. Then, continue reading and make changes, if needed, accordingly. In order to implement this system, you need to create the following classes:

`Person` class (abstract)

Instance variables: `firstName`(String), `lastName`(String), `emailId`(String), `phoneNumber` (long). Student class will be inheriting this class, and you will be using these in Student class. So, think about the access specifiers for the instance variables.
Constructor: do you need a no-arg constructor??? Think and decide!
Abstract Methods: `readInfo(), printInfo()` (both methods accepts nothing, returns nothing)

`Policies` Interface

Variables: `maxMarks` which is 100, `maxGpa` which is 4.0.
Methods: `calculateGpa`() – accepts a double array of marks.

Student class (extends Person class, implements Policies interface)

Instance variables: `studentNumber` (int), `programName` (String), `gpa` (double). This class will be extended by FulltimeStudent and ParttimeStudent classes. Think about the access specifiers for these instance variables. It should be based on whether you need to use them in subclasses or not. This class also implements Policies interface.

Constructor: do you need a no-arg constructor??? Think and decide!

Methods:

1. `readInfo()`: accepts nothing, return nothing. Reads all student information and save them in the instance variables. As you are inheriting Person class, you have access to instance variables of that class. Not permitted to create local variables. In order to read marks, `readMarks()` method should be called from this method. Include the required annotation also.
2. `readMarks()`: accepts nothing, return nothing. Reads number of courses, and then reads marks of all courses and stored them in a local double array. This method is always called from within the class (private method???) After reading the marks, this method will call `calculateGpa()`.
3. `calculateGpa()`: accepts a double array, returns nothing. This method is an implementation of the method in Policies interface. This method calculates the gpa and store it in the instance variable gpa. (you can go with a simple calculation to get a score out of 4. Example: if marks are 98, 96,95,97,94, gpa will be (480/5 * 100)*4.0 = 3.84). The formula for gpa calculation should use maxMarks and maxGpa variables of Policies interface.
4. `printInfo()`: accepts nothing, returns nothing. This method prints details of a student using formatted output (printf). Include the required annotation also.

FulltimeStudent Class (extends Student class)

Instance Variables: `tuitionFees` (double)
Methods:

1. `readInfo()`: accepts nothing, returns nothing. Calls `readInfo()` of the superclass. Then, this method reads tuition fees.
2. `printInfo()`: accepts nothing, returns nothing. Calls `printInfo()` of the superclass. This method then prints the tuition fees.

ParttimeStudent Class (extends Student class)

Instance Variables: `courseFeesTotal` (double), `credits` (double)
Methods:

1. `readInfo()`: accepts nothing, returns nothing. Calls `readInfo()` of the superclass. Then, this method reads total course fees and credit hours.
2. `printInfo()`: accepts nothing, returns nothing. Calls `printInfo()` of the superclass. This method then prints the total course fees and credit hours.

College Class

Instance variables: `name` (String – to store the name of the college), `[]students` (Student) (`students` is an array of Student)

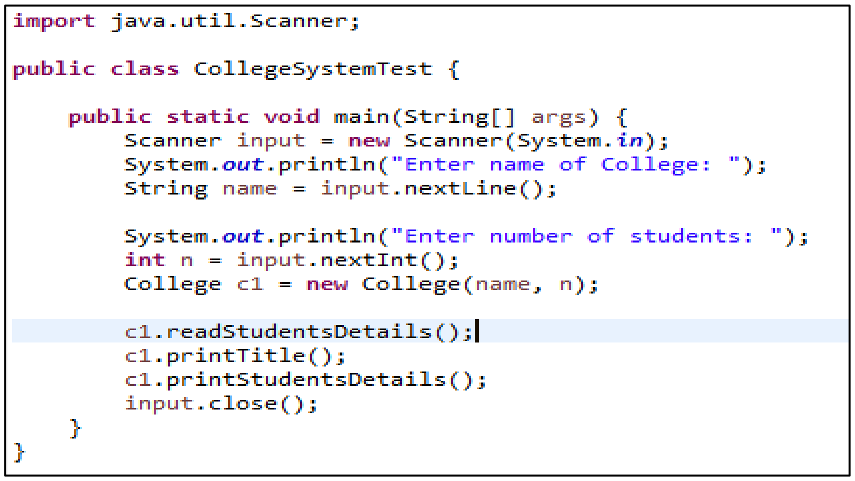
Constructor: Parameterized constructor gets a name that represents the name of the college and a number that represents the number of students in the college. This constructor sets `name` (the name of the college) and creates the `students` array.

Methods:

1. `printTitle()`: prints the title and the header row of the output (See output)
2. `ReadStudentsDetails()`: The purpose of this method is to read details of all students (in a loop). For each student, you need to first read the type of the student – 1 for full time or 2 for part time. If the user enters a number other than 1 or 2, a message "Wrong student type" should be displayed. Once the student type is read correctly, the right student object should be created. (Here, the student object is taking the form of a `FulltimeStudent` or a `ParttimeStudent`, based on the type of the student…. Polymorphism). Then, `readInfo()` should be invoked to read details of all students. Based on the type of the student, the right method will be invoked.
3. `printStudentDetails()`: this method has a for loop that calls `printInfo()` for all students in the `students[]` array.

CollegeSystemTest class

This is the driver class (test class), which means this class has the main method.



```java
import java.util.Scanner;

public class CollegeSystemTest {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter name of College: ");
        String name = input.nextLine();

        System.out.println("Enter number of students: ");
        int n = input.nextInt();
        College c1 = new College(name, n);

        c1.readStudentsDetails();
        c1.printTitle();
        c1.printStudentsDetails();
        input.close();
    }
}
```

Format your code with proper indentation and other formatting. Your code should be properly commented. Test plan and external documentation are not required for this exercise, but in future labs they will be required.

Expected Output (green – user input)

```
Enter name of College: Algonquin College
Enter number of students: 3
Enter details of student 1:
===========================
1 - Fulltime student
2 - Parttime Student
Enter Student type: 5
Wrong student type
1 - Fulltime student
2 - Parttime Student
Enter Student type: 1
Enter program name: CET
Enter student number: 1002
Enter first name: John
Enter last name: Doe
Enter email Id: doe@test.com
Enter phone number: 123456789
Enter number of courses: 5
Enter mark 1: 98
Enter mark 2: 96
Enter mark 3: 95
Enter mark 4: 97
Enter mark 5: 94
Enter tuition fees: 956.25
Enter details of student 2:
===========================
1 - Fulltime student
2 - Parttime Student
Enter Student type: 2
Enter program name: CST
Enter student number: 1005
Enter first name: Mark
Enter last name: Williams
Enter email Id: mark@test.com
Enter phone number: 213546789
Enter number of courses: 2
Enter mark 1: 85
Enter mark 2: 78
Enter total course fees: 950
Enter credit hours: 3.5
Enter details of student 3:
===========================
1 - Fulltime student
2 - Parttime Student
Enter Student type: 1
Enter program name: CP
Enter student number: 1008
Enter first name: Sam
Enter last name: Thomas
Enter email Id: sam@test.com
Enter phone number: 145236787
Enter number of courses: 4
Enter mark 1: 100
Enter mark 2: 99
Enter mark 3: 98
Enter mark 4: 96
Enter tuition fees: 963.50

Algonquin College - List of Students
************************************
     Program|Student#|           Name|          Email|          Phone|  GPA|     Fees|  Credits|
         CET|    1002|       John Doe|   doe@test.com|      123456789| 3.84|   956.25|
         CST|    1005|  Mark Williams|  mark@test.com|      213546789| 3.26|   950.00|     3.50|
          CP|    1008|     Sam Thomas|   sam@test.com|      145236787| 3.93|   963.50|
```

Submission

Submit your work (all java files, uml, javadoc) to Brightspace by the due date given in Brightspace. Also, demonstrate your work to your lab professor. Both submission and the demo of submitted code are required to get grades.

Grading Scheme

Item	**Marks**
Person class (correct access specifiers, abstract methods)	2
Student class (properly extended, correct access specifiers, 4 methods)	3
FullTimeStudent class (properly extended, correct access specifiers, 2 methods)	2
ParttimeStudent class (properly extended, correct access specifiers, 2 methods)	2
College class (correct access specifiers, constructors, 3 methods)	3
Policies interface (2 variables, 1 method)	2
CollegeSystemTest class (should NOT change)	Requirement
Comments & javadoc	3
UML	3