CPT121 - Programming 1
Study Period 2, 2009
Assignment 2

NOTE: This assignment is to be undertaken individually – no group work is permitted.

Assignment Deadline: 26/07/2009, 11:59 PM - Total Marks: 10

Introduction

The purpose of this assignment is to write Object-Oriented (OO) code. Most of the programmers who make a transition from C to Java find it particularly hard and annoying to code in an OO manner. However these days most of commercial applications are written in OO technologies such as – Java, VB.NET, C#.NET, etc. It is thus very important that you make yourself aware of this methodology of coding.

Assignment specifications

You have to design an application that simulates a University student administration system. The application should consist of three classes:

- Student
- Enrolment
- Admin

The Driver class should be the Admin class. All student objects must be stored in an array. Basic input validations need to be conducted. The details of the individual classes are as follows:

Student class (30 marks)

A student has a student number (8 digits, numbers only), first name (String), last name (String), gender (male or female) (String), date of birth (Gregorian Calendar), contact phone (10 digits) and the year (eg. 2009) they commenced the program (degree). You should have get and set methods for all these data members. In addition to this, you must have a method printDetails()[1] in the Student class.

Enrolment class (30 marks)

The Enrolment class needs to talk to the Student class. This class must have three methods, namely:

- addStudent()[2]
- getStudent()[3]
- printEnrolment()[4]

Admin class (30 marks)

The Admin class should drive the system. It should have a method fillData()[5]. This method should create an instance of enrolment, populate the enrolment with at least 5 students and finally print it. It should also have a method called searchStudent()[6]. This method allows searching for a student based on user input. This user input is asked at runtime on the command line, no menu implementation is needed. This method is repeated until a student is found in the enrolment. If a student is found, the program will terminate.

Finally, all classes except the driver class must have a constructor. In addition to that, your program should be well commented and indented for readability and documentation purposes.

Details of the methods:

Method	**Method Name**	**Description**
1	printDetails()	This method summarizes (*but does not print*) the student data members in a reader friendly manner.
2	addStudent()	Adds new students to the enrolment array; you can hardcode these students, in other words you don't have to ask for a user input.
3	getStudent()	Retrieves a student from the enrolment given its index (*index of array*)
4	printEnrolment()	It calls the printDetails() method to print the details on the console.
5	fillData()	This method must be present inside the *driver* class and should be called by the *main* method. This method: 1. Creates an enrolment list using the addStudent() method 2. Prints the details of the student using the printEnrolment() method
6	searchStudent()	This method must ask the user for input on the command line. The user enters a commencing year and the method searches for the students in the Enrolment array. If there are no matching students, a message is displayed stating that no matching students have been found and the user is asked again to enter a commencing year. Any matching students, if found, are listed with ALL their details. You may want to make use of the printDetails() method to prepare for printing these details.

Thus remember that you have three files: Student.java, Enrolment.java and Admin.java. Since Admin.java is the driver class, the possible program execution could look like the below. This is only a suggestion, not necessary, you can choose your own implementation, but make sure that the following is executed when the program starts:

1. Create an instance of enrolment

2. Populate the enrolment by adding students (use your imagination for the student details) using the fillData() method.

3. Print all students in the enrolment array with ALL of their details

4. Call the searchStudent() method to ask for user input

5. Display the search results and terminate or ask again for input if no results are found

```
javac Admin.java
java Admin

=============================
Student Admin System
=============================
Creating Enrolment...Done
Populating Enrolment...Done

The current students in the enrolment list are:

First / Last Name: John Smith
Gender: Male
Date of Birth: 23/05/1980
Contact: 0401123456
Student Number: 20978734
Commencing Year: 2004

First / Last Name: Danni Dabbler
Gender: Female
Date of Birth: 12/09/1991
Contact: 0401897654
Student Number: 99764253
Commencing Year: 2007

First / Last Name: Bob Builder
Gender: Male
Date of Birth: 30/10/1987
Contact: 0401999878
Student Number: 21530976
Commencing Year: 2007

============================================
Please enter a commencing year to search for:
2003

Sorry, no students in the enrolment list.

============================================
Please enter a commencing year to search for:
2007

The following students commenced in 2007:

Full Name: Danni Dabbler
Student Number: 99764253

Full Name: Bob Builder
Student Number: 21530976
```

Coding style (10 mark):

You should adhere to the following coding style guidelines when implementing your program for this assignment.

A. Different levels of program scope (methods, control structures, etc) should be indented consistently using levels of 3 or 4 spaces (do not use tabs).

B. A new level of indentation should be added for each new class/method/control structure that is “opened”.

C. Indentation should return to the previous level of at the end of a class/method/control structure (before the closing brace if one is being used).

D. Block braces should be aligned and positioned consistently in relation to the method/control structure they are opening/closing.

E. Lines of code should not exceed 80 characters in length (including indentation) - lines which will exceed this limit are split into two or more segments where required.

F. Expressions should be well spaced out – this includes assignment statements, arithmetic expressions and parameter lists supplied to method calls.

G. Source code should be spaced out into logically related code segments.

H. Identifiers used in the program for class/method/variable/constant names should adhere to the naming conventions discussed in the course notes.

I. Identifiers themselves should be meaningful without being overly explicit (long) – you should avoid using abbreviations in identifiers as much as possible (an exception to this rule is a “generic” loop counter used in a for-loop).

J. Each file in your program should have a comment at the top listing your name, student number and a brief (1-2 line) description of the contents of the file (generally the purpose of the class you have implemented).

K. You should include brief (1-2 line) comments describing what each logically related segment of code in your program is doing.

L. Comments should be placed on the line above the statement(s) or code segment they refer to.

Submission details

This assignment will be graded out of a total of 100 marks and contributes **10%** towards your final result for this course.

Please **submit *.java source files** rather than the compiled class files.

This assignment is not a hurdle in itself; rather it contributes towards the assessment work component (Weblearn tests and assignments) for this course (of which you must obtain 50% or better overall in order to pass the course).

This assignment is due to be submitted via **Weblearn** by **11:59pm on Sunday, 26th of July, 2009.**

Late Submissions

There will be **a late submission period of 5 days** for this assignment.

Late submissions that are received before **the late submission deadline** will incur **a late penalty of 10% per day (or part thereof)**, unless some prior arrangement has been made with the instructor regarding an extension. Late penalty deductions will be applied to the marks awarded for the assignment, rather than the total marks available for the assignment.

Submissions that are received after the late submission period will not be assessed unless prior arrangements have been made for an extension.

All queries regarding this assignment should be directed to the Assignment 2 forum in blackboard.