Advanced Programming

Lab Sheet 11 – Collections & Generics

Introduction

In this lab, you are going to work on a number of exercises requiring you to create and navigate Java collections, and to use generics. Most Java programs use collections heavily, and you'll need to become very proficient in writing the type of code discussed here to develop software in industry.

You are going to create a number of classes that, in extended form, could form the core of a University record system. The classes will keep track of lecturers, the units they teach, and the students taking them. You will need the following classes:

Person Abstract class that contains private properties for idNumber and name, along with appropriate getter and setter methods for these properties.

Lecturer Should extend Person, and provide a teachesUnit(Unit u) and dropUnuit(Unit u) method, to keep track of which units a lecturer teaches. The Lecturer class will also need a getAllUnitsTaught() method, which will return an ArrayList<Unit> of units that this Lecturer teaches. Note: each unit is only taught by one Lecturer in this example (one-to-many relationship).

Student Should extend Person, and provide a takesUnit(Unit u) and dropUnit(Unit u) method, to keep track of which units a student takes. The Student class will also need a getAllUnitsTaken() method that returns an ArrayList<Unit> of units that this student takes. Note: Many students can take a unit, and a student may take many units (many-to-many relationship).

Unit Will need a getLecturer() and setLecturer(Lecturer l) method to keep track of the Lecturer teaching the unit, as well as enrolStudent(Student s) and removeStudent(Student s) to keep track of the students taking the unit. The Unit class will also need a getTitle() and setTitle(String s) method to allow each unit to be given a name.

The Lecturer, Student and Unit classes will each need an ArrayList to keep track of their related entities.

Test Data for Tasks

Write a simple class, with a constructor method, to construct the following object graph, manipulating all of the Units, Lecturers & Students. You will also need to create associations between objects (i.e. update David to show that he teaches Programming, and update Programming to show that its taught by David).

Table 1: Lecturers to Create

Name	ID Number
Kris	1234
David	3456
Kaleem	5678
Darren	7890

Table 2: Units to Create

Unit Name	Lecturer
Advanced Programming	Kris
Algorithms and Data Structures	David
Programming	David
Enterprise Programming	Kaleem
Software Engineering Methodologies	Darren
Introduction to Matthew-Taming	Darren

Table 3: Students to Create

Name	ID Number	Units Taken
Mohammed	191234	Programming, Algorithms and Data Structures
Soufiene	192345	Advanced Programming, Programming
Matthew	193456	Algorithms and Data Structures, Programming
Robert	194567	Software Engineering Methodologies, Enterprise Programming
Huw	195678	Enterprise Programming, Introduction to Matthew-Taming
John	196789	Introduction to Matthew-Taming, Enterprise Programming
Elaine	197890	Programming, Enterprise Programming
Annabel	198901	Algorithms and Data Structures, Enterprise Programming
Susan	199012	Introduction to Matthew-Taming, Advanced Programming

Tasks

Each of the following exercises depends on you having created the object graph correctly. Be extra careful that the right units are taken by the right students, taught by the right lecturer. You should create a method in the class in which you created the object graph for each of the following exercises.

You can test your code by creating a main() method that constructs an instance of your class, thus creating the lecturers, units and students, and then calls the method for each exercise in turn. You should keep an ArrayList of all Students, one of all Lecturers, and one of all Units, to serve as starting points for each of these tasks.

1. Print out the name of each of the units taken by John.

2. Print out the names and ID numbers of all the students taught by Kris.

3. Print out the names of all of the units taken by more than two students.

4. Print out the names of all of the students unfortunate enough to not get taught either by Kris *or* by Darren.

5. Print out the name of the most popular unit.