Programação Orientada pelos Objectos

Exame (época normal)
2008/06/27

Atenção: A fraude numa prova de avaliação, mesmo quando detectada após a prova, é punida com a reprovação na cadeira.

Os estudantes do Politécnico de Almada (PA) são modelados pela interface IStudent:

```
public interface IStudent
{
    public String getName();
      // returns the student name
    public int getNumber();
      // returns the student number
    public Iterator<ICourse> getEnrollment();
      // the courses where the student is enrolled
    public boolean enrolled(ICourse aCourse);
      // is the student enrolled in the given course?
    public void enrollCourse(ICourse aCourse);
      // enrolls the student in the course
      // precondition: !enrolled(aCourse)
      // postcondition: aCourse.enrolled(this)
      // postcondition: enrolled(aCourse)
    public int grade(ICourse aCourse);
      // student's grade on the given course;
      // precondition: enrolled(aCourse)
}
```

A interface ICourse representa uma disciplina:

```
public interface ICourse
{
    public int getId();  // get course id number
    public String getName();  // get course name
    public int numberEnrolled();
      // get the number of students enrolled
    public Iterable<IStudent> getStudents();
      // the students enrolled in the course
    public boolean enrolled(IStudent aStudent);
      // the course has an enrollment for this student?
    public void enrollStudent(IStudent aStudent);
      // enrolls the student in the course
      // precondition1: !enrolled(aStudent)
      // precondition2: aStudent.enrolled(this)
    public boolean hasGrade(IStudent aStudent);
      // true if the student has an assigned grade
      // precondition: enrolled(aStudent)
    public int gradeOf(IStudent aStudent);
      // the grade of the student
      // precondition: hasGrade(aStudent)
}
```

Em todo o código que produzir, use a construção assert para verificar as pré-condições.

Primeiro grupo

Programe a classe abstracta Student que implementa a interface IStudent. Esta classe deverá conter os seguintes campos:

```
private static int lastNumber;
  // global counter for students' inscription number
private int number; // the student number
private String name; // the student name
protected ArrayList<ICourse> enrollment;
  // the courses the student has enrolled
```

Esta classe deverá, para além do construtor e dos métodos da interface que implementa, conter o método setInitialNumber(int number) que permite inicializar a variável estática lastNumber que representa o número do último aluno que se inscreveu no politécnico de Almada. Atenção que no construtor o número de aluno é automaticamente criado com base na referida variável estática.
Programe também os métodos toString() e equals(Student otherStudent). Os métodos grade(ICourse aCourse) e getEnrollment() são abstractos.

Segundo grupo

Programe as classes First (aluno do 1º ciclo) e Second (aluno do 2º ciclo) que herdam de Student. A nota (método grade) de um aluno do 1º ciclo numa disciplina pode ser obtida através de uma chamada ao objecto que representa a disciplina em causa (parâmetro aCourse), mas tal requer que o aluno tenha uma nota atribuída a essa disciplina.
No Politécnico de Almada os alunos admitidos no 2º ciclo têm obrigatoriamente de ter sido anteriormente alunos do 1º ciclo dessa mesma escola. Por isso a classe Second tem o seguinte

atributo adicional que representa quem tinha sido este aluno no 1º ciclo:

```
private First formerFirst;
```

A nota de um aluno do 2º ciclo numa disciplina é a nota que ele obteve enquanto aluno do 1º ou do 2º ciclo a essa disciplina. As disciplinas a que um aluno do 2º ciclo tem registo de inscrição (método `getEnrollment`) são a união do conjunto das que ele se inscreveu enquanto aluno de 1º ciclo com as que se inscreveu no 2º ciclo.

Terceiro grupo

Programe a classe `Course` que implementa a interface `ICourse`. Esta classe deverá conter os seguintes campos:

```
private int id;  //  course id number
private String name;  // course name
private ArrayList<Pair<IStudent, Integer>> grades;
// course grade for each enroled student
// Note: in the enrollment the grade is set to -1
```

Considere que está definida a classe genérica `Pair` com os seguintes métodos públicos:

```
public Pair(G first, H second)
public G getFirst()
public H getSecond()
```

Sugere-se que programe os métodos definidos na interface pela ordem proposta.

Quarto grupo

Desenhe um diagrama de classes em UML contendo todas as classes e interfaces propostas neste enunciado. Não se esqueça de representar os atributos, as operações, as relações de herança, de implementação e as dependências.