Object-Oriented Programming. Exercise 4.1

Unit 4. Pre-Defined Java Classes

Exercise 1. (project prGrading)

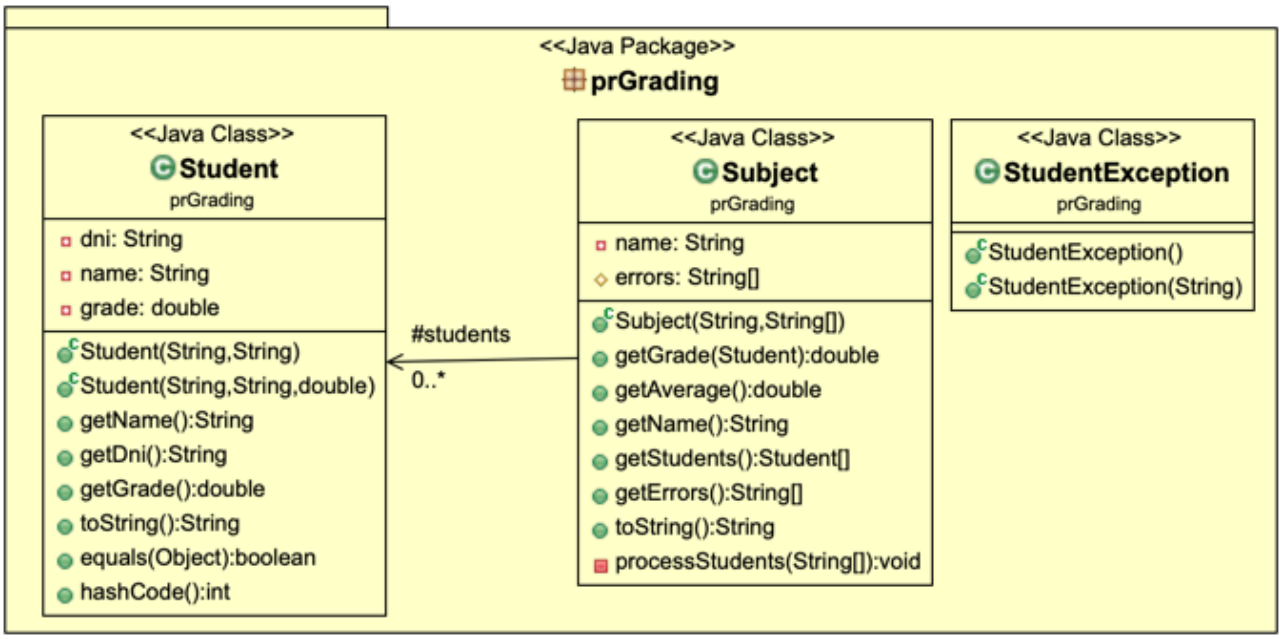

Figura 1: UML class diagram

In this exercise, we will create an application to record the grades obtained by students in a subject. For this, the classes `Student`, `Subject` and `StudentException` will be created in the package `prGrading`.

Class `StudentException`

Create the **checked** `StudentException` exception to handle exceptional situations that may occur in the following classes.

Class `Student`

The `Student` class maintains information about a student: his DNI[1] (`String`), his name (`String`) and his grade in a subject (`double`).

- The class will have two constructors, one in which the DNI, the name and the grade are provided (in this order) and the other with only the DNI and the name, in this case the grade being equal to zero. If the given grade is negative, a `StudentException` exception must be thrown with the message `"Negative grade"`.
- Two students are the considered equal if their names and DNIs match. The letter of the DNI may be in upper or lower case, but the names will distinguish upper and lower case.
- The textual representation of a student must show the name (as it has been provided) and the DNI (in parentheses), but not the grade. For example:

  ```
  Gonzalez Perez, Juan (22456784F)
  ```

- The `getName()`, `getDni()` and `getGrade()` methods allow access to the name, DNI and grade, respectively.

[1] DNI is the Spanish identity card.

Application `MainStudent`

Create a `MainStudent` application to test the previous class. In this class, two students must be created with the following data:

```
DNI: 22456784F, Name: Gonzalez Perez, Juan, Grade: 5.5
DNI: 33456777S, Name: Gonzalez Perez, Juan, Grade: 3.4
```

Next, the DNI and name of each student will be shown on the screen, as well as their grades in the above format. In addition, it will be checked if both students are equal (using `equals`) and the result will be shown on the screen. Finally, a third student will be created with the following data:[2]

```
DNI: 33456776S Name: Gonzalez Perez, Juan Grade: -3.4
```

The output obtained when executing the program should be the following:

```
DNI: 22456784F, Name: Gonzalez Perez, Juan, Grade: 5.5
DNI: 33456777S, Name: Gonzalez Perez, Juan, Grade: 3.4
The students Gonzalez Perez, Juan (22456784F) and Gonzalez Perez, Juan (33456777S) are not equal
prGrading.StudentException: Negative grade
  at prGrading.Student.<init>(Student.java:14)
  at MainStudent.main(MainStudent.java:13)
```

Class `Subject`

An instance of the `Subject` class maintains information about students taking a subject. A subject is created from the name of the subject and an *array* of `String` in which each element of the array will contain all the information to create a student with the following format:[3]

```
<DNI>;<Surname, name>;<Grade>
```

The three components must always appear, and will be separated by `';'`. For example:

```
55343442L;Godoy Molina, Marina;6.31
```

Although we will always give the name in the format `<surname>, <name>` we will not check the student's name, and it will be taken as it appears.

- The class will have a single constructor, which receives the name of the subject and the array of `String` as described above, and for each element in the array we will create, if possible, the student with the given name, DNI and grade, and store it in an array of students.

  The strings with which it is not possible to build a `Student` object will be stored in the `String` array `errors`, preceded by a comment indicating the problem because of which the student could not be created. For example, upon the input

  ```
  342424f2J;Fernandez Vara, Pedro;tr
  ```

  the following `String` will be included in the `errors` array:

  ```
  ERROR. Non numerical grade: 342424f2J;Fernandez Vara, Pedro;tr
  ```

- The `getGrade` method returns the grade of the student with the name and DNI of the student that it gets as an argument, if it exists. If it does not exist, it will throw a `StudentException` exception with the message `"The student` *xxx* `has not been found"`, where *xxx* is the textual representation of the student received as an argument.

[2] Note that the `StudentException` exception is checked. If you let the system handler handle the exception (propagating it in the `main` method) you will get a different output than the one indicated. To get the indicated one, catch the exception and treat it by displaying the execution stack with the `printStackTrace()` method.

[3] Use an auxiliary method `processStudents()` to process the array of strings and initialize the `students` and `errors` arrays. Use `Scanner` to process the array of strings. In order for `Scanner` to read decimal numbers with the period separator (e.g., `7.1`) we must send our `Scanner` object the message `useLocale` with argument `Locale.ENGLISH` (if our `Scanner` object is `sc` you can do `sc.useLocale(Locale.ENGLISH)`). To use `;` as a separator (with an arbitrary number of spaces before or after) you can use the expression `"\\s*[;]\\s*"` (with `sc.useDelimiter("\\s*[;]\\s*")`).

- The `getStudents()` and `getErrors()` methods return, respectively, the students array and the array of bad entries.
- Objects in the `Subject` class will be represented as a string (`toString()`) as shown in the example at the end of this exercise description (use `StringBuilder` or `StringJoiner` to create the representation).
- The `getAverage()` method returns the average of the students' grades for the subject. If there are no registered students, it will throw an exception `StudentException` with the message `"No students"`.

Application `MainSubject`

Create a `MainSubject` application to test the `Subject` class. In this class the subject `POO` will be created with three students with the following data:

```
DNI: 12455666F Name: Lopez Perez, Pedro Grade: 6.7
DNI: 33678999D Name: Merlo Gomez, Isabel Grade: 5.8
DNI: 23555875G Name: Martinez Herrera, Lucia Grade: 9.1
```

Next, the average of the grades of the subject will be shown, the students of the subject will be accessed one by one, showing the DNI of each of them on the screen, and the grade of the student with name `Lopez Perez, Pedro` and DNI `12455666F` will be printed. Finally, an attempt to show the grade of the student with name `Lopez Lopez, Pedro` and DNI `12455666F` will be made. Its execution will produce the following output:

```
Average 7.20
DNIs: 12455666F, 33678999D, 23555875G
Grade of Lopez Perez, Pedro (12455666F): 6.7
The student Lopez Lopez, Pedro (12455666F) has not been found
```

App `MainSubject2`

The output of the `MainSubject2` program provided as part of this exercise's statement must be:[4]

```
Grade of Lopez Turo, Manuel (23322443k): 4.3
The student Fernandez Vara, Pedro (34242432J) has not been found
Average 6.20
Students...
Garcia Gomez, Juan (25653443S): 8.1
Lopez Turo, Manuel (23322443K): 4.3
Merlo Martinez, Juana (24433522M): 5.3
Lopez Gama, Luisa (42424312G): 7.1
Errors...
ERROR. Negative grade: 53553421D;Santana Medina, Petra;-7.1
ERROR. Missing data: 55343442L,Godoy Molina, Marina;6.3
ERROR. Non numerical grade: 34242432J;Fernandez Vara, Pedro;2.k
Subject...
Algebra: { [Garcia Gomez, Juan (25653443S), Lopez Turo, Manuel (23322443K),
            Merlo Martinez, Juana (24433522M), Lopez Gama, Luisa (42424312G)],
           [ERROR. Negative grade: 53553421D;Santana Medina, Petra;-7.1,
            ERROR. Missing data: 55343442L,Godoy Molina, Marina;6.3,
            ERROR. Non numerical grade: 34242432J;Fernandez Vara, Pedro;2.k] }
```

Exercise 2. (project prGrading)

In this exercise we are going to define a `SubjectWithAverages` class that extends the `Subject` class with the possibility of calculating different types of averages.

[4] The last five lines of the displayed output correspond to the `toString()` of the subject, carriage returns have been introduced to be able to view them but they should appear on a single line.

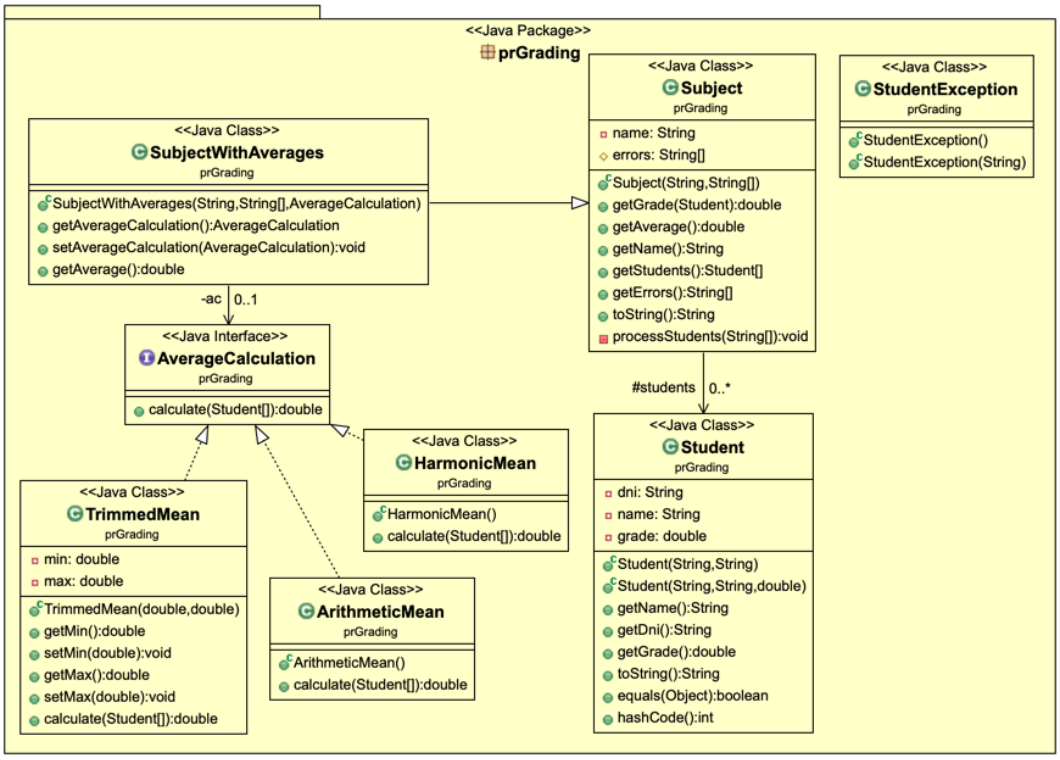

Figura 2: UML class diagram

Class `SubjectWithAverages`

Given the `AverageCalculation` interface that we introduce below, the `SubjectWithAverages` class extends the `Subject` class by redefining the `getAverage()` method. Grade averages are then calculated by invoking the `calculate(Student[] method)` provided by the `AverageCalculation` object that the `SubjectWithAverages` object has.

The class provides a single constructor with three arguments in which it receives, in addition to the subject name and the array of strings to process the students of the subject (such as the `Subject` class it inherits from), an instance of a class that implements the `AverageCalculation` interface. The class also provides `get` and`set` methods to modify this object.

Interface `AverageCalculation`

The `AverageCalculation` **interface** includes a single `calculate(Student[])` method that computes the average of the grades of the students in the array passed as argument.

Classes `ArithmeticMean`, `HarmonicMean` y `TrimmedMean`

The `ArithmeticMean`, `HarmonicMean` and `TrimmedMean` classes implement the `AverageCalculation` interface according to the following specifications:

- The `calculate` method provided by the `ArithmeticMean` class computes the arithmetic mean of the n students in the array it receives. In case there are no students, it will throw a `StudentException` exception with the message `"No students"`.

$$mean = \frac{1}{n}\sum_{i=0}^{n-1} studentGrade_i$$

- The `calculate` method provided by the `HarmonicMean` class calculates the harmonic mean of n students with grades above *0* according to the following equation:

$$mean = \frac{n}{\sum_{i=0}^{n-1} \frac{1}{studentGrade_i}}$$

In case there are no students that meet the specified requirement, it will throw a `StudentException` exception with the message `"No students"`.

- The `calculate` method provided by the `TrimmedMean` class computes the arithmetic mean of the grades of the students in the array it receives that are within the range defined by the values given in its constructor or by the `setMin(double)` and `setMax(double)` methods. In case there are no students with grades in this range, a `StudentException` exception will be thrown with the message `"No students"`.

App `MainSubjectWithAverages`

The output obtained when executing the `MainSubjectWithAverages` program provided as part of the statement must be:[5]

```
Grade of Lopez Turo, Manuel (23322443k): 4.3
The student Fernandez Vara, Pedro (34242432J) has not been found
Arithmetic mean: 6.20
Harmonic mean: 5.83
Mean of values in the range [5.00, 9.00]: 6.83
Students...
Garcia Gomez, Juan (25653443S): 8.1
Lopez Turo, Manuel (23322443K): 4.3
Merlo Martinez, Juana (24433522M): 5.3
Lopez Gama, Luisa (42424312G): 7.1
Errors...
ERROR. Negative grade: 53553421D;Santana Medina, Petra;-7.1
ERROR. Missing data: 55343442L,Godoy Molina, Marina;6.3
ERROR. Non numerical grade: 34242432J;Fernandez Vara, Pedro;2.k
Subject...
Algebra: { [Garcia Gomez, Juan (25653443S), Lopez Turo, Manuel (23322443K),
            Merlo Martinez, Juana (24433522M), Lopez Gama, Luisa (42424312G)],
           [ERROR. Negative grade: 53553421D;Santana Medina, Petra;-7.1,
            ERROR. Missing data: 55343442L,Godoy Molina, Marina;6.3,
            ERROR. Non numerical grade: 34242432J;Fernandez Vara, Pedro;2.k] }
```

Exercise 3. (project prGrading)

The enumerated type `Degree`

Create the enumerated type `Degree` with the following values: `COMPUTADORES`, `INFORMATICA`, `SOFTWARE` and `SALUD`.

Class `StudentWithDegree`

The `StudentWithDegree` class extends the `Student` class with a new attribute of type `Degree`.

- The class provides four constructors, in which, in addition to the constructors arguments of its superclass, it adds an additional argument of type `String` or `Degree` to specify the student's degree. In the case of receiving the title as a `String`, upper and lower case letters will be considered equivalent. You can use the `valueOf(String)` method to convert a `String` to the value of the corresponding enumerated type. Note that this method can throw an exception of type `IllegalArgumentException`, which should be caught and thrown one of type `StudentException` with the message `"Wrong degree"`.

[5] The last five lines of the output shown correspond to the `toString()` of the subject, carriage returns have been introduced to be able to view them but they should appear on a single line.

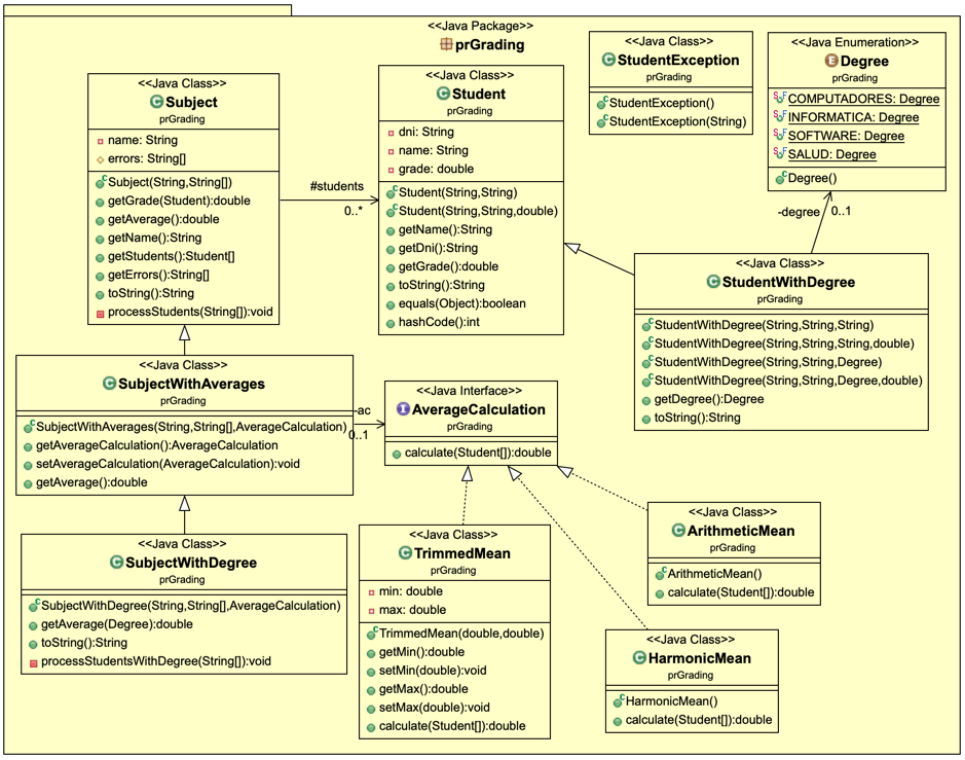

Figura 3: UML class diagram

- This class redefines the `toString()` method so that the instances of the class are rendered as the following example shows (note that the grade does not appear):

  ```
  Garcia Gomez, Juan (25653443S) SOFTWARE
  ```

Class `SubjectWithDegree`

The `SubjectWithDegree` class inherits from `SubjectWithAverages` so the subjects now contain instances of the `StudentWithDegree` class. The processing of the strings in the input array will now be performed by a method `processStudentsWithDegree(String[])` so that the student strings will now include the student's title. These will come in the format:

```
<DNI>;<Surname, Name>;<Degree>;<Grade>
```

The four components must always appear, and will be separated by the delimiter `;`. For example:

```
55343442L;Godoy Molina, Marina;INFORMATICA;6.31
```

The class redefines the `toString()` method, so that now the subject representation groups the students by degree. See example below.

Finally, the class provides an additional `getAverage(Degree)` method that returns the mean (based on the subject's `AverageCalculation` object) of the qualifications of students in the degree that it receives as an argument.

App `MainSubjectWithDegree`

The output obtained when executing the `MainSubjectWithDegree` program provided as part of the exercise's description must be:[6]

```
Grade of Lopez Turo, Manuel (23322443k) INFORMATICA: 4.3
The student Vara Riera, Pedro (34242432J) COMPUTADORES has not been found
```

[6]The last eleven lines of the output shown corresponds to the `toString()` of the subject, carriage returns have been introduced to be able to view them but they must appear on a single line.

```
Average: 6.73
Students...
Garcia Gomez, Juan (25653443S) SOFTWARE: 8.1
Lopez Turo, Manuel (23322443K) INFORMATICA: 4.3
Merlo Martinez, Juana (24433522M) COMPUTADORES: 5.3
Lopez Gama, Luisa (42424312G) SOFTWARE: 7.1
Gomez Ayuso, Laura (76213433Q) SALUD: 8.3
Navas Perez, Luis (89234024N) INFORMATICA: 6.5
Leon Zarate, Maria (92748362F) COMPUTADORES: 7.5
Errors...
ERROR. Negative grade: 53553421D;Santana Medina, Petra;Software;-7.1
ERROR. Missing data: 55343442L,Godoy Molina, Marina;Informatica;6.3
ERROR. Non numerical grade: 34242432J;Vara Riera, Pedro;Computadores;2.k
ERROR. Wrong degree: 21658324F;Jimenez Ayo, Pepe;Medicina;8.3
Subject...
Algebra: { [Merlo Martinez, Juana (24433522M) COMPUTADORES,
            Leon Zarate, Maria (92748362F) COMPUTADORES,
            Lopez Turo, Manuel (23322443K) INFORMATICA,
            Navas Perez, Luis (89234024N) INFORMATICA,
            Garcia Gomez, Juan (25653443S) SOFTWARE,
            Lopez Gama, Luisa (42424312G) SOFTWARE,
            Gomez Ayuso, Laura (76213433Q) SALUD],
           [ERROR. Negative grade: 53553421D;Santana Medina, Petra;Software;-7.1,
            ERROR. Missing data: 55343442L,Godoy Molina, Marina;Informatica;6.3,
            ERROR. Non numerical grade: 34242432J;Vara Riera, Pedro;Computadores;2.k,
            ERROR. Wrong degree: 21658324F;Jimenez Ayo, Pepe;Medicina;8.3] }
```