Programación Avanzada 1. Práctica 4.1

Tema 4. Clases Básicas de Java

Ejercicio 1. (proyecto pa1pa41, paquete notas)

Se va a crear una aplicación para anotar las calificaciones obtenidas por estudiantes en una asignatura. Para ello se crearán las clases `Estudiante`, `Asignatura` y `EstudianteException`.

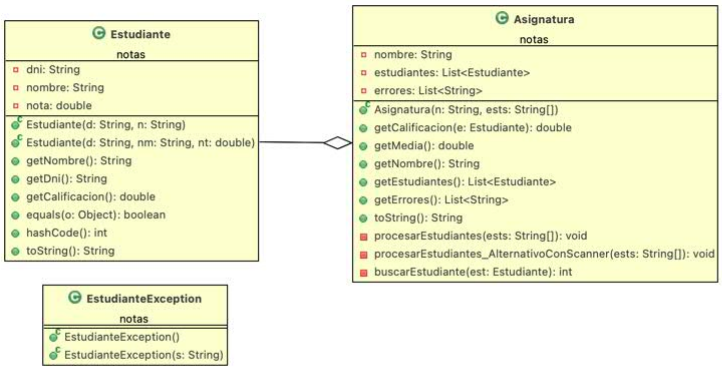

Figura 1: Diagrama de clases UML

Clase `EstudianteException`

Crea la excepción **comprobada** `EstudianteException` para manejar situaciones excepcionales que podrán producirse en las siguientes clases.

Clase `Estudiante`

Crea la clase `Estudiante` que mantiene información de un estudiante de quien se conocen el *dni* (`String`), el *nombre* (`String`) y la *calificación* obtenida en una asignatura (`double`). La clase tendrá dos constructores, uno en el que se proporcionan, en el orden especificado, el dni, el nombre y la calificación y otro en el que se proporcionan, en el orden especificado, sólo el dni y el nombre, siendo en este caso la calificación igual a cero. Si la calificación dada es negativa se deberá lanzar una excepción `EstudianteException` con el mensaje `"Calificación negativa"`.

Los siguientes métodos permiten acceder al nombre, al dni y a la calificación:

```
- String getNombre();
- String getDni();
- double getCalificacion();
```

Dos estudiantes son iguales si coinciden sus nombres y sus dni. La letra del dni podrá estar indistintamente en mayúsculas o minúsculas.

La representación textual de un objeto Estudiante contiene el nombre y el dni, en ese orden, pero no la calificación, según el formato del siguiente ejemplo (nótese que la coma forma parte del nombre, no del formato):

```
Lopez Turo, Manuel 23322443K
```

Aplicación `PruebaEstudiante`

Crea una aplicación (clase distinguida `PruebaEstudiante`) para probar la clase anterior. En esta clase se crean dos estudiantes con los datos siguientes:

```
DNI: 22456784F Nombre: Gonzalez Perez, Juan Nota: 5.5
DNI: 33456777S Nombre: Gonzalez Perez, Juan Nota: 3.4
```

Y se muestra por pantalla el nombre de cada estudiante, así como sus calificaciones. Además, se comprueba si ambos estudiantes son iguales, indicándolo por pantalla. Ten en cuenta que la excepción `EstudianteException` es de obligado tratamiento a la hora de implementar `PruebaEstudiante`. Ejecuta el programa.

A continuación modifica los datos del segundo estudiante tal y como se indica a continuación, ejecuta de nuevo el programa y observa lo que sucede.

```
DNI: 33456777S Nombre: Gonzalez Perez, Juan Nota: -3.4
```

Clase `Asignatura`

Crea la clase `Asignatura`. Una asignatura se crea a partir del nombre de la misma y de un *array* de `String` en la que cada elemento del array contendrá toda la información para crear un estudiante con el siguiente formato (deben aparecer siempre los tres tokens separados por `;`)[1] [2]

```
"<Dni>;<Apellidos, nombre>;<Calificación>"
```

El constructor recibe el nombre de la asignatura y el array de `String` descrito anteriormente y para cada elemento en el array deberá crear, si es posible, un objeto estudiante con el dni, nombre y calificación extraídas del `String`, y almacenarlos en la lista de estudiantes. Por ejemplo, para la siguiente entrada:

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

creará una estudiante con DNI `"55343442L"`, nombre `"Godoy Molina, Marina"`, y nota `6.31`, que será añadido a la lista de estudiantes.

Si no fuera posible crear un determinado estudiante, entonces deberá almacenar esta entrada errónea en otra lista de `String` (denominada `errores`) precedida de

[1] Tanto en el método `split` de la clase `String`, como en el método `useDelimiter` de la clase `Scanner`, se puede utilizar la expresión regular `"\\s*[;]\\s*"` como delimitador de tokens.

[2] En la clase `Scanner`, para poder leer números decimales con el separador punto (ej. `7.1`), se debe usar un objeto `Scanner sc` al que se le envía el mensaje `sc.useLocale(Locale.ENGLISH)`.

un comentario que indique cual ha sido el problema por el que no se ha podido crear el estudiante. Por ejemplo, ante las entradas:

```
"53553421D;Santana Medina, Petra;-7.1"
"55343442L,Godoy Molina, Marina;6.3"
"342424f2J;Fernandez Vara, Pedro;xxx"
```

se incluirá en la lista de `errores` los siguientes `String`:

```
"ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1"
"ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3"
"ERROR. Nota no numérica: 342424f2J;Fernandez Vara, Pedro;xxx"
```

El siguiente método de la clase `Asignatura` devuelve la calificación del estudiante `est` dado, si existe.

```
- double getCalificacion(Estudiante est) throws EstudianteException;
```

Si el estudiante no existe lanzará una excepción `EstudianteException` con un mensaje como en el siguiente ejemplo, donde `"Fernandez Vara, Pedro 34242432J"` es la representación textual del estudiante que no se encuentra.

```
"Estudiante Fernandez Vara, Pedro 34242432J no se encuentra"
```

Los siguientes métodos devuelven las listas de estudiantes y de entradas erróneas respectivamente:

```
- List<Estudiante> getEstudiantes();
- List<String> getErrores();
```

Además, dispondrá de una representación textual de los objetos de la clase como la que se muestra en el siguiente ejemplo (donde los saltos de línea se han introducido para aumentar la legibilidad), se debe utilizar `StringBuilder` para crear la representación.

```
Algebra: { [Garcia Gomez, Juan 25653443S, Lopez Turo, Manuel 23322443K,
Merlo Martinez, Juana 24433522M, Lopez Gama, Luisa 42424312G],
[ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1,
ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3,
ERROR. Calificación no numérica: 34242432J;Fernandez Vara, Pedro;2.k] }
```

Por último, el siguiente método devuelve la media de las calificaciones del estudiantado de la asignatura, considerando que si no hay estudiantes registrados, entonces este método lanzará la excepción `EstudianteException` con el mensaje `"No hay estudiantes"`.

```
- double getMedia() throws EstudianteException;
```

Aplicación `PruebaAsignatura`

Crea una aplicación (clase distinguida `PruebaAsignatura`) para probar la clase `Asignatura`. En esta clase se crea la asignatura `PA1` con tres estudiantes con los

siguientes datos:

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

A continuación muestra la *media* de las calificaciones de la asignatura y accede a los estudiantes de la asignatura, mostrando por pantalla el DNI de cada uno de ellos. Por último, imprime la calificación del estudiante `Lopez Perez, Pedro`. De nuevo ten en cuenta que la excepción `EstudianteException` es de obligado tratamiento a la hora de implementar `PruebaAsignatura`.

Después cambia el nombre del estudiante cuya calificación se ha de imprimir por `Lopez Lopez, Pedro`. Ejecuta de nuevo el programa y observa lo que sucede.

Aplicación `Main`

Para finalizar se presenta un ejemplo de uso más completo de las clases `Estudiante`, `Asignatura` y `EstudianteException` y la salida correspondiente.

```
import notas.EstudianteException;
import notas.Estudiante;
import notas.Asignatura;

public class Main {
    static final String[] als = {
        "25653443S;Garcia Gomez, Juan;8.1",
        "23322443K;Lopez Turo, Manuel;4.3",
        "24433522M;Merlo Martinez, Juana;5.3",
        "53553421D;Santana Medina, Petra;-7.1",
        "55343442L,Godoy Molina, Marina;6.3",
        "34242432J;Fernandez Vara, Pedro;2.k",
        "42424312G;Lopez Gama, Luisa;7.1" };
    public static void main(String[] args) {
        try {
            Asignatura algebra = new Asignatura("Algebra", als);
            try {
                Estudiante al1 = new Estudiante("23322443k", "Lopez Turo, Manuel");
                Estudiante al2 = new Estudiante("34242432J", "Fernandez Vara, Pedro");
                System.out.println("Calificacion de " + al1 + ": "
                                   + algebra.getCalificacion(al1));
                System.out.println("Calificacion de " + al2 + ": "
                                   + algebra.getCalificacion(al2));
            } catch (EstudianteException e) {
                System.err.println(e.getMessage());
            }
            try {
                System.out.printf("Media %4.2f\n", algebra.getMedia());
```

```
            } catch (EstudianteException e) {
                System.err.println(e.getMessage());
            }
            System.out.println("Estudiantes...");
            for (Estudiante estudiante : algebra.getEstudiantes()) {
                System.out.println(estudiante + ": " + estudiante.getCalificacion());
            }
            System.out.println("Errores...");
            for (String error : algebra.getErrores()) {
                System.out.println(error);
            }
            System.out.println(algebra);
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

La salida al ejecutar el programa anterior es:

```
Calificacion de Lopez Turo, Manuel 23322443k: 4.3
Estudiante Fernandez Vara, Pedro 34242432J no se encuentra
Media: 6.20
Estudiantes...
Garcia Gomez, Juan 25653443S: 8.1
Lopez Turo, Manuel 23322443K: 4.3
Merlo Martinez, Juana 24433522M: 5.3
Lopez Gama, Luisa 42424312G: 7.1
Errores...
ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1
ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3
ERROR. Calificación no numérica: 34242432J;Fernandez Vara, Pedro;2.k
Algebra: { [Garcia Gomez, Juan 25653443S, Lopez Turo, Manuel 23322443K,
Merlo Martinez, Juana 24433522M, Lopez Gama, Luisa 42424312G],
[ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1,
ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3,
ERROR. Calificación no numérica: 34242432J;Fernandez Vara, Pedro;2.k] }
```

Ejercicio 2. (proyecto pa1p41, paquete notas)

Se desea extender la clase `Asignatura` del proyecto `pa1p41` para que sea posible indicar la forma de calcular la media que se necesite en cada momento.

Así, se definirá una nueva clase `AsignaturaMedias` en el paquete `notas` que herede de la clase `Asignatura`, añadiendo el siguiente método, considerando que `CalculoMedia` será especificada a continuación:

- `double getMedia(CalculoMedia calc) throws EstudianteException;`

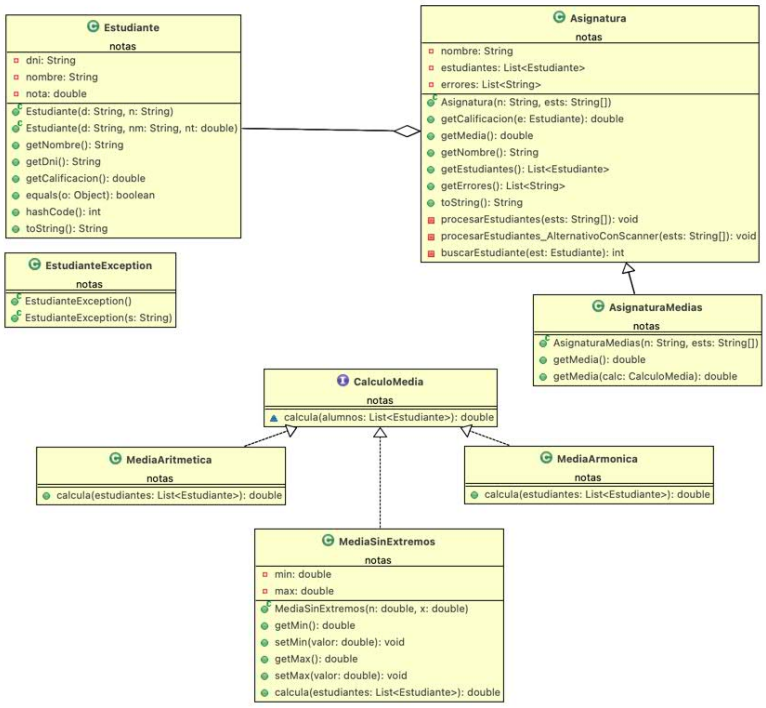

Figura 2: Diagrama de clases UML

De tal forma que este método calculará la nota media del estudiantado invocando al método `calcular` proporcionado por el objeto recibido como parámetro, que implementa la **interfaz** `CalculoMedia`.

También se redefinirá el método `getMedia()` utilizando el método anterior.

Interfaz `CalculoMedia`

Se debe definir la **interfaz** `CalculoMedia` que especifique el siguiente método.

```
- double calcular(ArrayList<Estudiante> estudiantes) throws EstudianteException;
```

Clases `MediaAritmetica`, `MediaArmonica` y `MediaSinExtremos`

Además, se deberán definir las clases `MediaAritmetica`, `MediaArmonica` y `MediaSinExtremos` que implementen la interfaz `CalculoMedia`, según las siguientes especificaciones:

- El método `calcular` proporcionado por la clase `MediaAritmetica` calcula la media aritmética de n estudiantes según la siguiente ecuación. En caso de que no haya estudiantes, lanzará una excepción `EstudianteException` con el mensaje `"No hay estudiantes"`:

$$media = \frac{1}{n} \sum_{i=0}^{n-1} calificacionEstudiante_i$$

- El método `calcular` proporcionado por la clase `MediaArmonica` calcula la media armónica de los k estudiantes con notas superiores a 0 según la siguiente ecuación. En caso de que no haya estudiantes que cumplan el requisito especificado, lanzará una excepción `EstudianteException` con el mensaje `"No hay estudiantes"`:

$$media = \frac{k}{\sum_{j=0}^{k-1} \frac{1}{calificacionEstudiante_j}}$$

- El método `calcular` proporcionado por la clase `MediaSinExtremos` calcula la media aritmética de aquellos valores comprendidos entre los extremos dados, ellos incluidos. En caso de que no haya estudiantes que cumplan el requisito especificado, lanzará una excepción `EstudianteException` con el mensaje `"No hay estudiantes"`. Los valores extremos se pasarán en el constructor de la clase y serán almacenados en sendas variables de instancia, para ser utilizados en el método `calcular`. Nótese que la clase `MediaSinExtremos` también proporciona dos métodos para consultar el valor mínimo y el valor máximo del rango.

```
- double getMin();
- double getMax();
```

Aplicación `Main2`

Para finalizar se presenta un ejemplo de uso más completo de las clases anteriormente especificadas y la salida correspondiente.

```
import notas.*;

public class Main2 {
    static final String[] als = {
        "25653443S;Garcia Gomez, Juan;8.1",
        "23322443K;Lopez Turo, Manuel;4.3",
        "24433522M;Merlo Martinez, Juana;5.3",
        "53553421D;Santana Medina, Petra;-7.1",
        "55343442L,Godoy Molina, Marina;6.3",
        "34242432J;Fernandez Vara, Pedro;2.k",
        "42424312G;Lopez Gama, Luisa;7.1" };
    public static void main(String[] args) {
        try {
            AsignaturaMedias algebra = new AsignaturaMedias("Algebra", als);
            try {
                Estudiante al1 = new Estudiante("23322443k", "Lopez Turo, Manuel");
                Estudiante al2 = new Estudiante("34242432J", "Fernandez Vara, Pedro");
                System.out.println("Calificacion de " + al1 + ": "
                                   + algebra.getCalificacion(al1));
                System.out.println("Calificacion de " + al2 + ": "
                                   + algebra.getCalificacion(al2));
            } catch (EstudianteException e) {
                System.err.println(e.getMessage());
            }
            try {
                CalculoMedia m1 = new MediaAritmetica();
                CalculoMedia m2 = new MediaArmonica();
                MediaSinExtremos m3 = new MediaSinExtremos(5.0, 9.0);
                System.out.println("Media aritmética: " + algebra.getMedia(m1));
                System.out.println("Media armónica: " + algebra.getMedia(m2));
                System.out.println("Media de valores en ["+m3.getMin()+", "+m3.getMax()+"]:
                                   + algebra.getMedia(m3));
            } catch (EstudianteException e) {
                System.err.println(e.getMessage());
            }
            System.out.println("Estudiantes...");
            for (Estudiante estudiante : algebra.getEstudiantes()) {
                System.out.println(estudiante + ": " + estudiante.getCalificacion());
            }
            System.out.println("Errores...");
            for (String error : algebra.getErrores()) {
```

```
                System.out.println(error);
            }
            System.out.println(algebra);
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

La salida al ejecutar el programa anterior es:

```
Calificacion de Lopez Turo, Manuel 23322443k: 4.3
Estudiante Fernandez Vara, Pedro 34242432J no se encuentra
Media aritmética: 6.199999999999999
Media armónica: 5.83482277207447
Media de valores en [5.0, 9.0]: 6.833333333333333
Estudiantes...
Garcia Gomez, Juan 25653443S: 8.1
Lopez Turo, Manuel 23322443K: 4.3
Merlo Martinez, Juana 24433522M: 5.3
Lopez Gama, Luisa 42424312G: 7.1
Errores...
ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1
ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3
ERROR. Calificación no numérica: 34242432J;Fernandez Vara, Pedro;2.k
Algebra: { [Garcia Gomez, Juan 25653443S, Lopez Turo, Manuel 23322443K,
Merlo Martinez, Juana 24433522M, Lopez Gama, Luisa 42424312G],
[ERROR. Calificación negativa: 53553421D;Santana Medina, Petra;-7.1,
ERROR. Faltan datos: 55343442L,Godoy Molina, Marina;6.3,
ERROR. Calificación no numérica: 34242432J;Fernandez Vara, Pedro;2.k] }
```