Object-Oriented Programming. Exercise 3.1

Unit 2. Exception management

Exercise 1. (project prData)

In this exercise, we will learn to define new exceptions, both checked and unchecked, and the differences in the treatment and propagation of both (unlike unchecked exceptions, checked exceptions must be announced in case a certain method throws or propagates them, or otherwise the method is obliged to capture it).

Thus, we will define methods that throw exceptions, methods that raise exceptions (that is, methods that receive an exception and do not catch it, but instead throw it), and methods that catch exceptions and deal with them, recovering from the error. It will also define constructors and methods that catch exceptions thrown by the system.

- Works with **unchecked** and **checked** exceptions (must be announced or caught).
- The `calcAverage` method throws an exception.
- The `calcStandardDeviation` method throws an exception thrown by `calcAverage`.
- The `toString` method catches exceptions and handles them, recovering from the error.
- The `setRange` method catches the exceptions thrown by the system, and *throws* another type of exception.
- The constructor catches the exceptions thrown by the system (`Double.parseDouble`), handles them and recovers from the error.
- The main program receives the data of the arguments, and in case they are not correct, it catches the exception, displays an error message and terminates the application. Also, in case of invoking the `calcAverage` or `calcStandardDeviation` methods, you should also catch the exceptions, and display an error message.

The unchecked exception `DataException`

Create the **unchecked** exception `DataException` (in the`prData` package) to handle exceptional situations that may occur in the following classes.

- `DataException()` *// Constructor without message*
- `DataException(String)` *// Constructor with a given message*

Class `Data`

The `Data` class (in the `prData` package) contains an array with a sequence of real numbers, and also contains an array with the possible errors that have occurred in its construction. In addition, it also contains the `min` and `max` values that will be used in some operations.

- `Data(String[],double,double)`

Constructs the object. The first parameter contains a data sequence, from where the *double* value of each element (of type `String`) must be extracted and stored in the data array. If any element of the

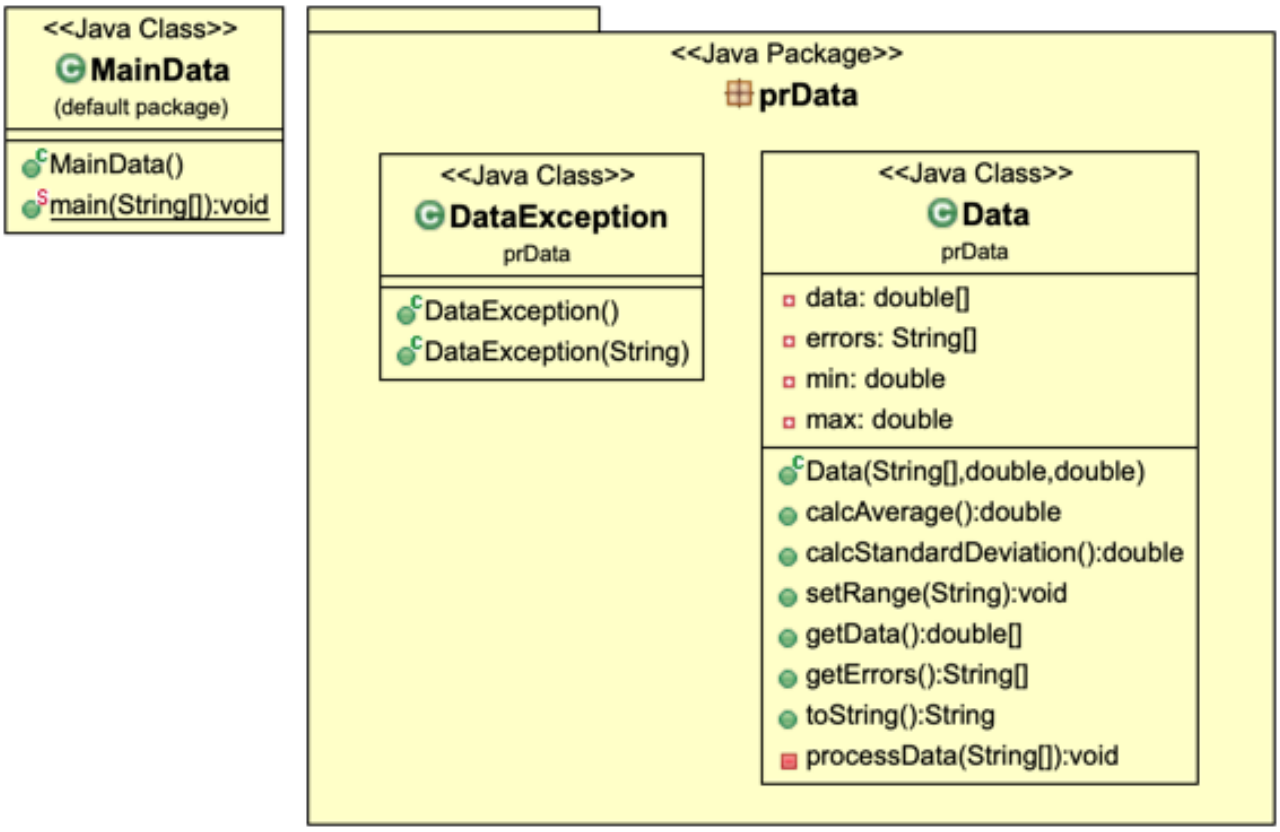

Figura 1: UML class diagram

input array cannot be converted to real number, then it will be added to the error array. In addition, it also receives the values of the `min` and `max` attributes as a parameter. [1]

- `calcAverage(): double`

  Calculates and returns the arithmetic mean for the stored data that is within the range delimited by the `min` and `max` values (inclusive). Throws the `DataException` exception with the message "There is no data in the specified range" if there are no elements within the specified range.

$$average = \frac{1}{n_{d_i}} \sum_i d_i : min \leq d_i \leq max$$

$$n_{d_i} = \#\{d_i : min \leq d_i \leq max\}$$

- `calcStandardDeviation(): double`

  Calculates and returns the standard deviation corresponding to the stored data that are within the range delimited by the `min` and `max` values (both inclusive). Propagates the `DataException` exception thrown by the `calcAverage` method if there are no elements within the specified range.[2]

$$\sigma = \sqrt{\frac{1}{n_{d_i}} \sum_i (d_i - average)^2} : min \leq d_i \leq max$$

$$n_{d_i} = \#\{d_i : min \leq d_i \leq max\}$$

[1] To calculate the numerical value represented in a `String` you can use the *class method* `parseDouble` of the `Double` class , which throws the `NumberFormatException` exception if it cannot be returned. Also, to adjust the size of the arrays to the number of elements they contain, you can use the *class method* `copyOf` of the `Arrays` class.

[2] To compute the square of a number, use the *class method* `pow` of the `Math` class. Also, to calculate the square root of a number you can use the *class method* `sqrt` of the `Math` class.

- `setRange(String): void`

  Updates the values of the *min* and *max* attributes to the values that should be extracted from the `String` parameter, assuming that it has the form "min;max", that is, that the string contains two numeric values separated by a semicolon (`;`), and where the first value corresponds to *min* and the second value corresponds to *max*. Throws the `DataException` exception (with the message "Data error setting range") in the event that *some error* occurs in the extraction of the two values.[3]

- `getData(): double[]`

  Returns the data array of the `Data` object.

- `getErrors(): String[]`

  Returns the array of errors that contains the object.

- `toString(): String  // @Redefinition`

  Returns the textual representation of the object, in the following format (without the line breaks):

  ```
  Min: 10.0, Max: 20.0,
  [5.0, 9.0, 10.0, 12.0, 13.0, 17.0, 20.0, 25.0],
  [Pepe, María, Paco, Ana, Juan, Lola],
  Average: 14.4, StandardDeviation: 3.6110940170535577
  ```

  In case of errors in calculating the arithmetic mean or standard deviation, a textual representation similar to the following example will be returned (without the line breaks):

  ```
  Min: 0.0, Max: 4.0,
  [5.0, 9.0, 10.0, 12.0, 13.0, 17.0, 20.0, 25.0],
  [Pepe, María, Paco, Ana, Juan, Lola],
  Average: ERROR, StandardDeviation: ERROR
  ```

Application `MainData`

Develop an application (in the anonymous package) that allows you to test the previous classes. To do this, a `data` object of the `Data` class will be created, where the necessary values for its creation will be received as arguments of the `main` method: *min* value, *max* value, and the values to fill the array of strings (to construct it, it will be necessary to use the `copyOfRange` method of the `Arrays` class), and its representation will be shown on the screen. Note that there must be at least three input values, if there is not enough it will be reported with an "Error, there are not enough values" message. The first two values must be numerical, otherwise it will give the message "Error, when converting a value to a real number (N)", being N the first value with incorrect format.[4]

Then, the message `setRange` will be sent to the `data` object with argument `"0;4"`, and the representation of the `data` object will be shown on the screen again. Then, a new `setRange` message will be sent, but this time with argument `"15 25"`.

If an error occurs with the received arguments (there is insufficient data or some of the *min* or *max* values cannot be properly calculated), the application will end up showing the corresponding error message on the screen.

We will run the application three times:

- For the first execution, the following data will be introduced as arguments to the `main` method:

  `10 20 5 9 Pepe 10 Maria 12 13 Paco 17 20 Ana 25 Juan Lola`

[3] To search for the position where the character `;` is in a `String`, you can use the `indexOf` method. Also, to get a certain substring you can use the `substring` method.

[4] To know how to pass values as arguments to the main program, from the programming environment, see the Java Development Environment Guide (Eclipse).

The output will be as follows (without the line breaks of each representation of the data object, that is, in three lines):

```
Min: 10.0, Max: 20.0,
[5.0, 9.0, 10.0, 12.0, 13.0, 17.0, 20.0, 25.0],
[Pepe, María, Paco, Ana, Juan, Lola],
Average: 14.4, StandardDeviation: 3.6110940170535577

Min: 0.0, Max: 4.0,
[5.0, 9.0, 10.0, 12.0, 13.0, 17.0, 20.0, 25.0],
[Pepe, María, Paco, Ana, Juan, Lola],
Average: ERROR, StandardDeviation: ERROR

Data error setting range
```

- For the second execution, the following data will be introduced as arguments to the `main` method:

  ```
  10
  ```

  La salida por pantalla será la siguiente:

  ```
  Error, there are not enough values
  ```

- For the third execution, the following data will be introduced as arguments to the `main` method:

  ```
  10 hola 5 9 Pepe 10 María 12 13 Paco 17 20 Ana 25 Juan Lola
  ```

  The output will be as follows (to show the part that appears between parentheses, use the `getMessage` method):

  ```
  Error, when converting a value to a real number (For input string: "hola")
  ```

Exercise 2. (project prData)

Create a copy of the 'prData' package and call it 'prData2', and create a copy of the 'MainData' class named 'MainData2' in the anonymous package (which imports `prData2` instead of `prData`). In these new package and class, make the necessary changes so that the `DataException` exception (from the 'prData2' package) is now **checked**.