David

Bowling Game kata

Dejo aquí escrito el kata *Bowling Game* tal y como aparece en el libro de Emily Blache.

Clase String Java

Para resolver este kata es necesario conocer en detalle cómo funciona la clase `String` en Java.

En el **cap. 11: *Strings*** del libro *Beginning Java 8/9 Fundamentals* encontrarás todo lo que necesitas conocer sobre la clase `String`. Está muy bien explicado.

- Nos interesan sobretodo los métodos de la clase `String` para manipularlos, explicados en el epígrafe ***String Operations***.

```
Boolean b = str1.equals(str2);
int len = str.length();
int caracter = str.charAt(i);
Boolean b = str1.equalsIgnoreCase(str2);
Boolean b = str1.equals(str2);
Boolean b = str.isEmpty();
String str2 = str1.toUpperCase();
String str3 = str1.toLowerCase();
int index = str.indexOf('p');
i t index = str.lastIndexOf('p');
String s2 = String.valueOf("10"); // s2 has "10" OJO: devuelve un string, no int
String s2 = "Hello".substring(1, 4);
"\n  \r   \t    hello\n\n\n\r\r".trim() will return "hello"
String newStr = oldStr.replace('o', 'e');
Boolean b = str.startsWith("This");
Boolean b = str.endsWith("program");
String[] parts = str.split(",");
String str = String.join(",", "AL", "FL", "NY", "CA", "GA");
```

Lógica del negocio o reglas del juego

Create a program, which, given a valid sequence of rolls for one line of American Ten-Pin Bowling, produces the total score for the game. This is a summary of the rules of the game:

- Each **game**, or “line” of bowling, includes ten turns, or “**frames**” for the bowler.
- In each frame, the bowler gets up to two **tries** to knock down all the **pins**.

- If in two tries, he fails to knock them all down, his **score** for that frame is the total number of pins knocked down in his two tries.
- If in two tries he knocks them all down, this is called a “**spare**” and his score for the frame is ten plus the number of pins knocked down on his next **throw** (in his next **turn**).
- If on his first try in the frame he knocks down all the pins, this is called a “**strike**”. His turn is over, and his score for the frame is ten plus the simple total of the pins knocked down in his next two **rolls**.
- If he gets a spare or strike in the **last (tenth) frame**, the bowler gets to throw one or two more **bonus balls**, respectively. - These bonus throws are taken as part of the same turn. If the bonus throws knock down all the pins, the process does not repeat: the bonus throws are only used to calculate the score of the final frame.
- The **game score** is the total of all **frame scores**.

Requisitos funcionales: qué NO hace el programa

Here are some things that the program will not do:

- We will not check for valid rolls.
- We will not check for correct number of rolls and frames.
- We will not provide scores for intermediate frames.

Glosario de términos

https://en.wikipedia.org/wiki/Glossary_of_bowling

Casos test

The input is a scorecard from a finished bowling game, where “X” stands for a strike, “-” for no pins bowled, and “/” means a spare. Otherwise figures 1-9 indicate how many pins were knocked down in that throw.

Sample games:

```
12345123451234512345
```

always hitting pins without getting spares or strikes, a total score of 60.

```
XXXXXXXXXXXX
```

a perfect game, 12 strikes, giving a score of 300.

```
9-9-9-9-9-9-9-9-9-9-
```

heartbreak - 9 pins down each round, giving a score of 90.

```
5/5/5/5/5/5/5/5/5/5/5
```

a spare every round, giving a score of 150.

Como puedes imaginar, estos casos no van a ser suficientes. Intenta conseguir más ejemplos de partidas reales para chequear todos los posibles estados de las variables del programa.

Ejemplos reales:

http://www.hanoverhornets.org/pe/wp-content/uploads/2015/11/bowling-score-32.jpg

https://4.bp.blogspot.com/-IQyJRkPZnmM/WoxeL8p0_rI/AAAAAAAAKg0/pbOKCIIqofYuL0CAfqyNABIeqH2n6NEAACLcBGAs/s1600/Scoring%2Bbowling%2Bpractice%2Bsheets.jpg

https://qph.fs.quoracdn.net/main-qimg-be11e73c08fd01e8c60d2653cf7a1af0-c

Retrospectiva

Retrospecterrrr

- Did you do any design before you started coding? If so, does your code have this design now?
- At what point did you realize you can't simply loop over frames, that you in fact need to refer to the previous frame as well as the current one in order to calculate the score? In an ideal world, when should you have realized this?
- Look at the code you have ended up with, and compare it with the above description of the rules of bowling. Is there any similarity in the words used in each?
- Did you do enough refactoring? How would you know?
- How did you decide which tests to write, and in which order? Did it matter what order you implemented them in?

Kata resuelto por Robert Martin

El kata está resuelto en el libro en nuestro Drive:

Chapter 6. A Programming Episode, Agile-Principles-Patterns-and-Practices-in-C:
https://drive.google.com/open?id=18ehxsv7HXMatNp1egeKUySzbOVQP5dIM

Ideas for after the Dojo
Read this article "Engineer Notebook: An Extreme Programming Episode" by Robert C. Martin, where he describes solving this kata together with Robert S. Koss. Follow along in your editor. Does he do the kata the same way as you would?

Mi solución (con un *bug*)

https://github.com/dfleta/bowling-game-kata

Aquí otra en *youtube* (pendiente de chequear):
https://www.youtube.com/watch?v=OPGTPQ4kURU