BOWLING SCORE KEEPER---BSK

The objective is to develop a program, BSK, that can calculate the score of a single bowling game. There is no graphical user interface. You work only with Java objects and JUnit tests in this assignment.
The program's requirements are divided into a set of user stories, which serve as your "todo" list (see below). If case you need to handle invalid input values, define an exception.

Instructions:

- **KEEP TRACK** of the time (expressed in **MINUTES**) it takes to fulfill the assignment. Be honest, you will not be evaluated based on the completion time---in other words, it does not matter if you complete the assignment in 2 hours, rather than in 4 hours, and so on.
- **DEVELOP** the BSK program **WITHOUT** using **TDD**. Thus, for example, you can use Test-Last Development (i.e., first write all the production code and, in the end, test it) or Incremental Test-Last Development (i.e., first write the production code for a chunk of functionality and then test it, thus repeat the previous step for any other chunk of functionality). Whatever the development approach is, you must **TEST** BSK. Moreover, remember to **APPLY** your design knowledge (e.g., refactor your code when needed). You are allowed to use the IDE you prefer.
- Once the assignment is completed (i.e., you are confident that all the stories are correctly implemented), **FILL OUT** the post-mortem survey available at https://forms.gle/EauYz2j962bDfUu29, and then **SEND ME** your implementation of BSK (e.g., the Eclipse project). The **DEADLINE** to deliver your implementation **May, the 20th at 11:59 (a.m.)**.

Constraints:

- You **MUST NOT** use **TDD** to develop BKS.
- The BSK program **MUST** be developed in **Java**.
- You **CANNOT** use Eclipse plug-ins like EclEmma or Pit (or similar tools) when carrying out this assignment.

User Stories

1. Frame

*Each turn of a bowling game is called a **frame**. 10 pins are arranged in each frame. The goal of the player is to knock down as many pins as possible in each frame. The player has two chances, known as **throws**, to do so. The value of a throw is given by the number of pins knocked down in that throw.*

Requirement: Define a frame as composed of two throws. The first and second throws should be distinguishable (i.e., it should be possible to get the first throw and the second throw).
Example: [2, 4] is a frame with two throws, in which two pins were knocked down in the first throw and four pins were knocked down in the second.

Note that the notation "[<first_throw>, <second_throw>]" is just used to describe a frame in this document. In other words, you can represent each throw with an integer variable.

2. Frame Score

An ordinary frame's score is the sum of its throws.

Requirement: Compute the score of an ordinary frame.
Examples: The score of the frame [2, 6] is 8. The score of the frame [0, 9] is 9.

3. Game

A single game consists of 10 frames.

Requirement: Define a game, which consists of 10 frames.
Example: The sequence of frames [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] represents a game. You will reuse this game from now on to represent different scenarios, modifying only a few frames each time.

4. Game Score

The score of a bowling game is the sum of the individual scores of its frames.

Requirement: Compute the score of a game.
Example: The score of the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] is 81.

5. Strike

*A frame is called a **strike** if all 10 pins are knocked down in the first throw. In this case, there is no second throw. A strike frame can be written as [10, 0]. The score of a strike equals 10 plus the sum of the next two throws of the subsequent frame.*

Requirement: Recognize a strike frame. Compute the score of a strike. Compute the score of a game containing a strike.
Examples: Suppose [10, 0] and [3, 6] are consecutive frames. Then the first frame is a strike and its score is equal to 10 + 3 + 6 = 19. The game [10, 0] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 94.

6. Spare

*A frame is called a **spare** when all 10 pins are knocked down in two throws. The score of a spare frame is 10 plus the value of the first throw from the subsequent frame.*

Requirement: Recognize a spare frame. Compute the score of a spare. Compute the score of a game containing a spare frame.
Examples: [1, 9], [4, 6], [7, 3] are all spares. If you have two frames [1, 9] and [3, 6] in a row, the spare frame's score is 10 + 3 = 13. The game [1, 9] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 88.

7. Strike and Spare

A strike can be followed by a spare. The strike's score is not affected when this happens.

Requirement: Compute the score of a strike when it's followed by a spare. Compute the score of a game with a spare following a strike.
Examples: In the sequence [10, 0] [4, 6] [7, 2], a strike is followed by a spare. In this case, the score of the strike is 10 + 4 + 6 = 20, and the score of the spare is 4 + 6 + 7 = 17. The game [10, 0] [4, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 103.

8. Multiple Strikes

Two strikes in a row are possible. You must take care when this happens for the computation of the first strike's score requires the values of throws from two subsequent frames.

Requirement: Compute the score of a strike that is followed by another strike. Compute the score of a game with two strikes in a row.
Examples: In the sequence [10, 0] [10, 0] [7, 2], the score of the first strike is 10 + 10 + 7 = 27. The score of the second strike is 10 + 7 + 2 = 19. The game [10, 0] [10, 0] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 112.

9. Multiple Spares

Two spares in a row are possible. The first spare's score is not affected when this happens.

Requirement: Compute the score of a game with two spares in a row.
Example: The game [8, 2] [5, 5] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 98.

10. Spare as the Last Frame

When a game's last frame is a spare, the player will be given a bonus throw. However, this bonus throw does not belong to a regular frame. It is only used to calculate the score of the last spare.

Requirement: Compute the score of a spare when it's the last frame of a game. Compute the score of a game when its last frame is a spare.
Example: The last frame in the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 8] is a spare. If the bonus throw is [7], the last frame has a score of 2 + 8 + 7 = 17. The game has a score of 90.

11. Strike as the Last Frame

When a game's last frame is a strike, the player will be given two bonus throws. However, these two bonus throws do not belong to a regular frame. They are only used to calculate score of the last strike frame.

Requirement: Compute the score of a spare when it's the last frame of a game. Compute the score of a game when the last frame is a strike.
Example: The last frame in the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [10, 0] is a strike. If the bonus throws are [7, 2], the last frame's score is 10 + 7 + 2 = 19. The game's score is 92.

12. Bonus is a Strike

Further bonus throws are not granted when a game's last frame is a spare and the bonus throw is a strike.

Requirement: Compute the score of a game in which the last frame is a spare and the bonus throw is a strike.
Example: In the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 8], the last frame is a spare. If the bonus throw is [10], the game's score is 93.

13. Best Score

A perfect game consists of all strikes (a total of 12 of them including the bonus throws), and has a score of 300.

Requirement: Check that the score of a perfect game is 300.
Example: A perfect game looks like [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] with bonus throws [10, 10]. It's score is 300.

14. Real Game

Requirement: Check that the score of the game [6, 3] [7, 1] [8, 2] [7, 2] [10, 0] [6, 2] [7, 3] [10, 0] [8, 0] [7, 3] [10] is 135.

Congratulations, you are done!