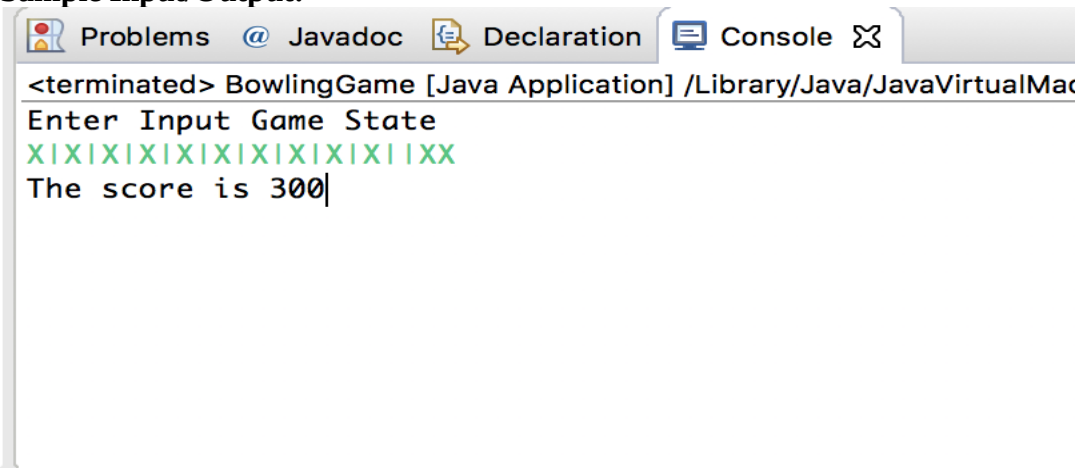
Documentation Guide For 10-pin Bowling Game

Problem Description: The following application returns the current score of a 10 pin Bowling Game given the input state of the Game.

Input: A string in the given format "--|--|--|--|--|--|--|--|--|4/||X"
Output: The total score for the given input state i.e. 20 for this case.
Sample Input/Output:



The implementation consists of two modules BowlingGame.java and BowlingScorer.Java

Module 1(BowlingGame.java):

This module basically implements the conversion of each chance in each frame in the string input into its corresponding numerical equivalents along with keeping a track of the Frame in which the conversion is being done.

In this module we have implemented the following methods:

private void inputStringToPoints(String input):
This method basically assigns points corresponding to the character representing the chance in the string input.

private void add(int pins):
This method adds the points corresponding to the chance in an array throwball[].

private void adjustCurrentFrame(int pins):
This method adjusts the current frame number based on the points in that chance.

private boolean adjustFrameForStrike(int pins):
This method adjusts the current frame number when a strike occurs in that chance.

private void nextFrame():
This method advances the current frame to the next one and makes sure that the number of frames doesn't exceed 10.

public int scoreinFrame(int theFrame):
This method returns the current score achieved till that frame.

Module 2(BowlingScorer.java):

This module basically implements the scoring algorithm for each chance in each frame.

In this module we have implemented the following methods:

public void addChance(int pins):
This method adds the points corresponding to the chance in an array throwball[].

public int scoreinFrame(int frameNo):
This method calculates the store for the current frame and returns the current score achieved till that frame.

private boolean isStrike():
This method validates whether the current chance results in a strike or not.

private boolean isSpare():
This method validates whether the current chance results in a spare or not.

private int twochancesInFrame():
Calculates score for a frame without any strike or spare.

private int strikeBonus():
Calculates score for a frame in case of a strike.

private int spareBonus():
Calculates score for a frame in case of a spare.

Unit Tests:

The following unit tests were used while designing the applications and ensured maximum test coverage:

TestCase1:
We checked for the score of a single frame.

TestCase2:
We checked for the score of a simple spare.

TestCase3:
We checked for the score of a simple strike.

TestCase4 :
We checked for the score of a perfect spare scenario in all the frames.

TestCase5 :
We checked for the score of a perfect strike scenario in all the frames.

TestCase6 :
We checked for the score of a perfect miss scenario in all the frames.

TestCase7 :

We checked for the score of a alternate miss scenario in all the frames.

TestCase8 :
We checked for the score of a game with no strike or spare scenario in any of the frames.

TestCase8 :
We checked for the score of a game with no strike or spare scenario in any of the frames.

TestCase9 :
We checked for the score of a game for a simple frame after spare.

TestCase10 :
We checked for the score of a game where we test bonus strike after spare in last frame.

TestCase11 :
We checked for the score of a game where we test for a miss in last bonus chance after strikes in all the frames.

TestCase12 :
We checked for the score of a game where we test for bonus chance after spare in the last frames.