Project 1 Design Proposal

The StrikeZone Game

Jean-Luc Jackson
10/10/21
W200 Project 1

Program Description

StrikeZone is played from the perspective of a Major League Baseball player on the mound. The objective is to last as long as you can without the manager pulling you from the game. Using the command line, the user decides what type of pitch to throw and where to locate it. The user earns points when they get an out and loses them when batters get on base. When your points run out, your manager pulls you and the game is over.

User Interaction

All user interaction occurs on the command line. The user will be shown input options depending on the screen they are on. When the program begins, the user interacts with the main menu option. During gameplay, the user will have pitching options to input.

Scoring

Outcomes in favor of the pitcher such as strikeouts and puts will be worth positive points for the user. Bad outcomes such as a base hit or a walk earn negative points. When the points first approach or zero the manager visits the mound for a warning and pep-talk. If the user hits zero points again, the manager pulls the user from the game and the game ends.

Program Organization

Classes

MainMenu

This class presents the start-up screen and game options to the user. It will show a leaderboard with a history of high scores and their saved usernames. The methods include:

- load_high_scores()
    - Load saved users and user scores from a score history text file.
- play_ball()
    - Asks user which Pitcher they would like to use, which team they would like to pitch against, and how many innings they would like to play (if they want less than 9). Begins game by initiating a new BaseballGame object. When the game has ended, the returned user score is saved to the score history text file.

- explain_game()
    - Explains how to play the game and how pitch outcomes are calculated.

BaseballGame

This is the outermost class once the game has started. Similar to a real-life baseball game, this class keeps track of the user's score and the current inning relative to the total innings to be played. The methods include:

- play_inning()
    - This method initiates an inning and updates the game's running attributes after the inning has finished (user score, current inning, innings to play).

Inning

The Inning class is responsible for managing the game's information at the individual-inning level. The object attributes include outs, batting_order_index, hits_this_inning, and strikeouts. The methods include:

- batter_up()
    - Initiate the next PlateAppearance object by passing it the next Batter on the roster's lineup and the Pitcher.
    - Update the user's score by adding to or subtracting from it according to the PlateAppearance's outcome (see scoring guide below).
    - At the end of the inning, return the total score to the BaseballGame.

PlateAppearance

The PlateAppearance class manages the interaction between the Pitcher and the current Batter. It keeps track of the pitches that have been thrown (the count) and their outcomes; it updates the StrikeZone object to display past pitches to the user; and it is the primary point of interaction for the user during game-mode. The methods include:

- at_bat()
    - This is the main method for this class. First, take the user's input (the equivalent of a catcher's "sign") and check for any errors.
    - Pass the sign to the Pitcher object and handle the returning dictionary of probabilities (see Pitcher class below).
    - Pass the probabilities to the Batter. Handle the Batter's returned code about the outcome with conditionals:
        - If a strike or ball occurred, call the StrikeZone's add_pitch_result() to update the display string.
        - If the outcome ends this PlateAppearance (strikeout, walk, base hit, or put out), return it to the Inning class above.

StrikeZone

The StrikeZone class holds a string representation of the strike zone for this PlateAppearance. It keeps track of the pitch history by updating the display string with an "X" or an "O" for strikes or balls in the zone where they occurred. The methods include:

- get_blank_strikezone()
    - Return an empty strike zone.
- add_pitch_result()
    - Takes a string code communicating the pitch's result and the location. It then adds an "X" or "O" to display strikes and balls.
- get_strikezone()
    - Return the current string representation of the strike zone.

Player

The Player class is the parent class to the Pitcher and Batter subclasses. It holds the player's name, their team, their position, and their batting average.

Pitcher

The Pitcher class is a child of the Player class. It is modeled after a real-life MLB pitcher, having as attributes the pitcher's pitch repertoire and respective strike zone outcome probabilities for each zone and for each pitch type. Historical statistics are sourced from Statcast via MLB's Baseball Savant website.

- A pitcher's probabilities will be saved in a POPZ Table (Probabilities of Outcomes by Pitch type and Zone), saved as a dictionary of objects.
- Probabilities are found for each zone in the strike zone (14 total) and for each type of pitch (fastball, offspeed, breaking: 3 total), resulting in 42 sub-dictionaries.
    - Outcomes of interest include: Called Strike, Swinging Strike, Ball, Foul Ball, In Play Out, Single, Double, Triple, Homerun, Hit By Pitch
- The outermost dictionary's keys are pitch types: "Fastball", "Offspeed", and "Breaking". Its values will be tuples whose items are dictionaries pertaining to each zone. These zone dictionaries (42 total) have keys matching the outcomes of interest above and values matching the probabilities of each outcome.

The Pitcher methods include:

- get_pitch_thrown()
    - Based on the user's inputted pitch type and zone, get the probability dictionary to be passed to the Batter.

Batter

The Batter class is a child of the Player class. It is an object representation of a baseball Player in the phase of Batter. The methods include:

- get_pitch_outcome()
    - Given a dictionary with probabilities of outcomes (from the Pitcher), determine which outcome will occur and return it.