HW4: The Game of Pig
Due date: Tuesday, Mar 15th at 9pm. Late turn-in deadline is Thursday, Mar 17th at 9pm.

1. Background:
Pig is a folk jeopardy dice game described by John Scarne in 1945, and was an ancestor of the modern game Pass the Pigs® (originally called PigMania®).

The rules are simple: Two players race to reach 100 points. Each turn, a player repeatedly rolls a standard die until either a 1 is rolled or the player holds and scores the sum of the rolls (i.e. the turn total). At any time during a player's turn, the player is faced with two decisions:

a) roll - If the player rolls a
   - 1: the player scores nothing and it becomes the opponent's turn.
   - 2 - 6: the number is added to the player's turn total and the player's turn continues.

b) hold - The turn total is added to the player's score and it becomes the opponent's turn.

To familiarize yourself with play, you can play an optimal Pig opponent online. The key decision facing a player is how large a turn total should be risked to possibly get an even larger total. In this homework assignment, you will be writing a relatively aggressive (although somewhat naïve) algorithm. In this homework, once an automated player reaches a turn total of 20 or more, they will hold. Otherwise, they will continue rolling.

2. Assignment: By the end of this homework, you should be able to simulate a 2 player game of Pig. But we’ll break it up into pieces (methods) which you can work on individually. In Section 3, there is a listing of methods to write. You MUST write these in order, as some methods will need to invoke other methods. Name your file HW4.java.

3. Methods to implement:

   a) roll: this method should return a random integer, 1-6 inclusive. You have already written this method for a previous lab, and you are free to use that method.

   b) holdAt20: This method should mimic a single turn of Pig where the player rolls until a 1 is rolled or until the turn total is greater than or equal to 20

      • input parameter(s): none
      • return value: an integer value representing the number of points the player scored on this turn (ie, their turn total).
      • output: your method should print the value of each roll the player takes. When the player's turn is over, the method should print out the total points scored.

      • Examples:

      • holdAt20() might generate the output:
        Roll: 4
Roll: 6
Roll: 5
Roll: 5
Turn total: 20
In this case, the method should return 20 (the player's score for this turn)

Another call to holdAt20() might generate:
Roll: 4
Roll: 3
Roll: 1
Turn total: 0
In this case, the method would return 0 (the player's score for this turn)

c) Once you have part b working, modify this method by adding a parameter to your
holdAt20 method. This parameter should be an integer and will represent the player's
current score. Modify your method to immediately hold if the turn would put the player's
score at 100 or greater. The method should still return the score for the current turn.

• Example:
  holdAt20(90) might generate the output
  Roll: 2
  Roll: 3
  Roll: 4
  Roll: 4
  Turn total: 13
  This call should return 13

d) holdUI: This method will also simulate a single player turn, but this method will
incorporate user input. Specifically, this method will ask the user whether or not to continue
rolling at each step.

• Input parameter(s): the method takes no parameters, but it will need to use a Scanner
to read user input. Make the Scanner a class variable and initialize it to read from
System.in. In other words, the Scanner variable should be declared and initialized
outside of any method.

• Return value: this method should return the turn's total score, like the holdAt20
method.

• Output format: The output for this method would also include a prompt to the user, and
  a running total as the user rolls. It should also include roll values, same as holdAt20.

• Example of a single call to holdUI. User input is in blue.
  Roll: 2
  Turn total: 2  (1 to Roll, 2 to Hold): 1
  Roll: 2
  Turn total: 4  (1 to Roll, 2 to Hold): 1
  Roll: 4
In this example, the method should return 24 (total score of turn)

- The user should make ALL decisions. In other words, the player can continue rolling as many times as they would like, and they can hold whenever they would like. They should also be able to continue rolling even if their current turn total would surpass 100 points.

e) playGame: This method should implement a 2-player game of Pig where the user plays against a computer player that always rolls until a 1 ("pig") is rolled, or the turn total is greater than or equal to 20, or the score plus the turn total is greater than or equal to 100 (ie, the behavior implemented in holdAt20). The human player can make a decision when to roll or hold (ie, the behavior implemented in holdUI). The first player is chosen randomly.

- Input parameter(s): One String, which is the human player's name/identifier.
- Return value: this method has no return value
- Output format:
  - Before the game, randomly select which player the user will be, and print the line "[Computer or User's name], you will be player #.", where # is the user's player number: either 1 or 2.
  - Before each turn, print a line with "Player 1 score: " and player 1’s score. Print another line with "Player 2 score: " and player 2’s score. Finally, print a line with "It is player #’s turn.", where "#" is replaced by the current player number. Play starts with player 1 and then alternates.
  - After a "pig" roll of 1, or a "hold", print a line with "Turn total: " followed by the turn total. In the case of a "pig", this turn total is 0. Then, print a line with "New score: " followed by the new score for the current player.
  - When the game ends (ie when one player reaches 100 points), you should print out a message indicating the game is over and which player won.
- Example game (long!), invoked with playGame("Valerie")

Valerie, you will be player 2.
Player 1 score: 0
Player 2 score: 0
It is player 1's turn.
Roll: 5
Roll: 3
Roll: 5
Roll: 1
Turn total: 0
New score: 0
Player 1 score: 0
Player 2 score: 0
It is player 2's turn.
Roll: 6
Turn total: 6  (1 to Roll, 2 to Hold): 1
Roll: 5
Turn total: 11 (1 to Roll, 2 to Hold): 1
Roll: 6
Turn total: 17 (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 19 (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 21 (1 to Roll, 2 to Hold): 2
Turn total: 21
New score: 21
Player 1 score: 0
Player 2 score: 21
It is player 1's turn.
Roll: 5
Roll: 6
Roll: 3
Roll: 5
Roll: 1
Turn total: 0
New score: 0
Player 1 score: 0
Player 2 score: 21
It is player 2's turn.
Roll: 6
Turn total: 6  (1 to Roll, 2 to Hold): 1
Roll: 6
Turn total: 12 (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 14 (1 to Roll, 2 to Hold): 1
Roll: 6
Turn total: 20 (1 to Roll, 2 to Hold): 2
Turn total: 20
New score: 41
Player 1 score: 0
Player 2 score: 41
It is player 1's turn.
Roll: 3
Roll: 3
Roll: 6
Roll: 4
Roll: 4
Turn total: 20
New score: 20
Player 1 score: 20
Player 2 score: 41
It is player 2's turn.
Roll: 3
Turn total: 3  (1 to Roll, 2 to Hold): 1
Roll: 3
Turn total: 6  (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 8  (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 10 (1 to Roll, 2 to Hold): 1
Roll: 4
Turn total: 14 (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 16 (1 to Roll, 2 to Hold): 1
Roll: 4
Turn total: 20 (1 to Roll, 2 to Hold): 2
Turn total: 20
New score: 61
Player 1 score: 20
Player 2 score: 61
It is player 1's turn.
Roll: 5
Roll: 1
Turn total: 0
New score: 20
Player 1 score: 20
Player 2 score: 61
It is player 2's turn.
Roll: 3
Turn total: 3  (1 to Roll, 2 to Hold): 1
Roll: 3
Turn total: 6  (1 to Roll, 2 to Hold): 1
Roll: 5
Turn total: 11  (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 13  (1 to Roll, 2 to Hold): 1
Roll: 6
Turn total: 19  (1 to Roll, 2 to Hold): 2
Turn total: 19
New score: 80
Player 1 score: 20
Player 2 score: 80
It is player 1's turn.
Roll: 3
Roll: 1
Turn total: 0
New score: 20
Player 1 score: 20
Player 2 score: 80
It is player 2's turn.
Roll: 2
Turn total: 2  (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 4  (1 to Roll, 2 to Hold): 1
Roll: 4
Turn total: 8  (1 to Roll, 2 to Hold): 1
Roll: 2
Turn total: 10  (1 to Roll, 2 to Hold): 1
Roll: 3
Turn total: 13  (1 to Roll, 2 to Hold): 1
Roll: 6
Turn total: 19  (1 to Roll, 2 to Hold): 1
Roll: 5
Turn total: 24  (1 to Roll, 2 to Hold): 2
Turn total: 24
New score: 104

Game over!!! Valerie wins! The score was 104-20

4. Hints and Tips:
   a) Consider using boolean flags in this assignment. It can be easier to set a boolean flag like `computerGoesFirst` to true or false (based on a random number) and then base decisions about what to print on that variable later in your program. Likewise, keep track of whose turn it is with a flag like `computersTurn` which can alternately be set to true or false as a turn ends.
   b) Although it's tempting, avoid using class variables to keep track of players' scores in this assignment. Trust me, they are quite confusing to keep track of.

5. Submission:
   a) Don't forget your Name, userid, section number, and collaboration statement as comments at the top of your file.
   b) You may submit as many times as you want, but we only grade the last submission.
   c) Submit your .java file (not your .class file!) to the HW4 assignment on BB.
   d) Due date is Tuesday, Mar. 15th at 9pm. Late turn-in deadline is Thursday, Mar. 19th at 9pm.