COMP 1600 Fall 2025

Project 4: Pig

Due by: Friday, November 14, 2025 at 11:59 p.m.

The game of Pig is a jeopardy dice game. The rules are very simple:

1. Two players take turns rolling a single, six-sided die.
2. The first player to score 100 or more wins.
3. If you roll a 2, 3, 4, 5, or 6, that number is added to your turn's score.
4. After rolling a 2, 3, 4, 5, or 6, you can choose to roll again or to hold.
5. If you hold, the points you have accumulated during your turn will be added to your game score.
6. If you roll again, you get the opportunity to add more points to your turn score.
7. However, if you ever roll a 1 (called the pig), you lose all the points in your current turn, and the other player gets to roll.

Your mission is to write a Java program that will allow two players to play Pig. The two players may be two humans, a human and a computer, or two computers. Unsurprisingly, you must also write this program using static methods.

Specification

You will be required to write four static methods to implement the game of Pig. You are permitted to implement additional methods if you like, but these four are required.

1. **`main()` Method**
2. **`playGame()` Method**
3. **`playTurn()` Method**
4. **`getDecision()` Method**

`main()` Method

Your `main()` method is where the program starts, of course. In the `main()` method, you must prompt the user to select a Human vs. Human, Human vs. Computer, or Computer vs. Computer game.

Sample output from `main()` looks like the following:

```
The Game of Pig
-------------------------------------
1. Human vs. Human
2. Human vs. Computer
3. Computer vs. Computer

What kind of game do you want to play?
```

When the user enters 1, 2, or 3, you should called the `playGame()` method with the appropriate arguments.

`playGame()` Method

The signature of the `playGame()` method is as follows:

```
public static void playGame (boolean player1, boolean player2)
```

The parameters to the method indicate whether each player is human (`true`) or computer (`false`).

Despite its name, the `playGame()` method is not where a great deal of the implementation of the game takes place. The `playGame()` method contains a loop that continues as long as both players have a score less than 100. Each turn of the loop, the method prints out the current scores for both Player 1 and Player 2 and then calls the `playTurn()` for the appropriate player.
The `playTurn()` method returns the score earned in the current player's turn. Then, the method switches the current player to the other player and runs the loop again.

After one of the players reaches (or passes) a score of 100 (ties are impossible), the loop ends, the method prints out the score for each player, and declares the winner.

`playTurn()` Method

The signature of the `playTurn()` method is as follows:

```
public static int playTurn (boolean player, int number, int totalScore)
```

The `player` parameter indicates whether the player for this turn is human (`true`) or computer (`false`). The `number` parameter is either 1 (for Player 1) or 2 (for Player 2).
The `totalScore` parameter gives the total score, excluding the current turn, for the current player. This parameter is not needed for a human player, but a computer player requires this information as

part of its strategy. The return value is the score earned on a given turn (which is non-zero if the player decided to hold before rolling a 1 or zero if a 1 was rolled).

The `playTurn()` method goes through the mechanics of a single turn of Pig. This method begins by announcing whether turn is for Player 1 or Player 2 and whether that player is Human or Computer.

Next, the method will roll the die (by generating a random number between 1 and 6). Then, the method will loop as long as the die does not show 1 and as long as the player continues to decide to roll. At each roll (other than 1), the current turn score increases by the value of the die. The player is given the choice to hold or roll again after each roll (unless 1 was rolled). The player's choice is made in the `getDecision()` method.

After the loop ends, either by the player holding or rolling a 1, the method prints `Turn over` and returns the score earned in that turn.

`getDecision()` Method

The signature of the `getDecision()` method is as follows:

```
public static boolean getDecision (boolean player, int turnScore, int
totalScore)
```

The `player` parameter indicates whether the player for this turn is human (`true`) or computer (`false`). The `turnScore` gives the current score earned this turn. The `totalScore` parameter gives the total score, excluding the current turn, for the current player. The return value is either a `false` to hold or a `true` to roll again.

This method begins by printing the current turn score.

If the player is human, the method prompts the user to hold or roll and then waits for an `h` or `r` to be entered. If an `r` is entered, the method returns `true`, otherwise, it returns `false`. Before returning, the method prints `Human player holds` or `Human player rolls`, as appropriate. Note: You will need to create a new `Scanner` object in this method to read in the input.

If the player is a computer, the method will always adopt the same strategy, called Hold-at-20-or-Goal. That is, if its current turn score is 20 or more, it will hold. Also, if its current turn score plus its game score is greater than or equal to 100, it will hold. For example, if its game score is 93 and its turn score is 9, it will hold because 93 + 9 = 102, winning the game. This strategy is a good one, but it is not truly optimal. Optimal strategies take the opponent's score into account as well. In this case,

before returning, the method prints `Computer player holds` or `Computer player rolls`, as appropriate.

Sample Execution

The output from these games can be long, and, because of the random nature of the game, the game itself is not deterministic. Below are links to partial output from a Human vs. Human game, partial output from a Human vs. Computer game, and the full output from a Computer vs. Computer game. Please examine the Computer vs. Computer game to see how the end of a game should be properly formatted.

- **Human vs. Human Game**
- **Human vs. Computer Game**
- **Computer vs. Computer Game**

Note: Your formatting must match **exactly** to the character to get full credit.

Hints

Methods

You must implement the three (four, counting `main()`) methods exactly as I have described them, with the same parameters and return values. Please copy and paste the headers I have given into your code. This project is actually very short in terms of code. Try to figure out the method (ha, ha) behind the madness. What do these methods do, and why? Understanding how they are supposed to work can save you a lot of time and typing.

Random Numbers

Recall that you will need to use `Math.random()` to generate random numbers. It's your job to figure out how to get a random integer in the range from 1 to 6, inclusive.

Input

The game only has two input statements, the first reading the integer corresponding to the type of game, and the second being the `r` or `h` indicating whether to roll or hold. The first is an `int`, and the second is a `String`. Normal `Scanner` usage should work for both, though you will have to create a new `Scanner` object in your `getDecision()` method.

Testing

Test heavily. It can be annoying trying to get to the end of a game, but make sure that the computer properly implements both parts of the Hold-at-20-or-Goal strategy. For testing purposes, you can generate non-random numbers to see how the program behaves.

Turn In

Your IntelliJ project should be called `Project4`, but the class you create inside should be called `Pig`. Upload the `Pig.java` file from the `Project4\src` folder wherever you created your project to **Brightspace**.

All work must be submitted before Friday, November 14, 2025 at 11:59 p.m. unless you are going to use a grace day.

All work must be done individually. You may discuss general concepts with your classmates, but it is never acceptable for you to look at someone else's code. Please refer to the course policies if you have any questions about academic integrity. If you have trouble with the assignment, I am always available for assistance.

Grading

Your grade will be determined by the following categories:

Category	Weight
Functionality of `main()`	15%
Functionality of `playGame()`	20%
Functionality of `playTurn()`	25%
Functionality of `getDecision()`	20%
Output formatting	10%
Style and comments	10%

Under no circumstances should any student look at the code written by another student. Tools will be used to detect code similarity automatically.

Code that does not compile will automatically score zero points.