CIS 162 Project 3
A Game of Pig

Due Date
- Wednesday, 6/21/16

Before Starting the Project
- Read chapter 4 and 5.1-5.3
- Read this entire project description before starting

Learning Objectives
After completing this project you should be able to:
- *use* a Graphical User Interface (GUI)
- write methods to meet specific requirements
- *write* conditional statements with boolean expressions
- *write* a looping construct

Game Rules
- The game requires two six-sided dice.
- Players take turns until someone wins by earning at least 100 points
- Players begin a turn by rolling both dice and tallying the points. The current player continues rolling and accumulating points until rolling a '1' or choosing to 'hold'. If the player rolls a 1 then all points from this round are lost. If the player rolls a pair of 1s then all player points are lost.
- Play rotates to the next player
- Your game will be between a person and the computer. The player always goes first.

Step 1: Create a New BlueJ Project

Step 2: Class Definition: GVdie
Rather than writing your own Die class, we are providing a completed class for you. Create a new class in BlueJ called `GVdie` and delete all of the provided code. Cut and paste the provided code from (GVdie.java) into the newly created class. It should compile with no errors.

Step 3: Create a class called Pig (60 pts)
- Provide appropriate names and data types for each of the instance variables. Maintain two `GVdie` objects, player's score, computer's score, current round score and a `final int` for the winning score of 100.

- `public Pig ( )` - a constructor that initializes all of the instance variables to appropriate values, *instantiates* two dice and prints a welcome message.
- `public int getRoundScore ( )` – return the current round score.
- `public int getPlayerScore ( )` – return the player's score.
- `public int getComputerScore ( )` – return the computer's score.
- `private void rollDice ( )` - roll both dice. If either value is '1' then set the round score to 0. Otherwise, increment the round score by the total of the dice. Print the round score. This method is used by both the player and the computer.
- `public void playerRolls ( )` - performs the first half of the player turn: 1) invoke the `rollDice( )` method, 2) print the player's score if the round is over, or 3) print an appropriate message if the player wins. Refer to the sample output below. Set the player score to zero if double 1s are rolled.
- `public void playerHolds ( )` - performs the second half of the player's turn: 1) update the player's score, 2) reset the round score to zero and 3) print the player's score. Refer to the sample output below.
- `public void computerTurn ( )` - performs the computer's entire turn: continue rolling the dice and updating the round score until: 1) a '1' is rolled or 2) the round score surpasses 19 or 3) the computer wins. Set the computer score to zero if double 1s are rolled. Use a `while loop` to repeatedly roll the dice. Update the computer's score and the round score after each. Print the computer's score. Refer to the sample output below.
- `public void restart ( )` – reset all instance variables to start a new game. **Do not** instantiate new dice objects.
- `public GVdie getDie (int num)` - return the requested die. Legal values for the parameter are 1 or 2.
- `public boolean playerWon ( )` - return `true` if the player score is currently high enough to win. Otherwise, return `false`.
- `public boolean computerWon ( )` - return `true` if the computer score is currently high enough to win. Otherwise, return `false`.

Step 4: Software Testing (15 pts)

Software developers must plan from the beginning that their solution is correct. BlueJ allows you to instantiate objects and invoke individual methods. You can carefully check each method and compare actual results with expected results. However, this gets tedious. Another approach is to write methods that automatically play a game and therefore call all of the other methods. You can carefully review the results of a game to confirm that it works correctly.

- `public void playerTurn ( )` - simulates the player's entire turn: continue rolling the dice and updating the round score until a '1' is rolled or the round score surpasses 19 or if the player wins. Update the player's score and the round score. Print the player's score.
- `public void autoGame ( )` - automates an entire game by alternating between a player's turn and computer's turn until one of them wins. Keep track and report the total number of turns taken. Report who wins. Reset all values before starting the new game.

JUnit Testing

As an additional test, JUnit is a Java library that helps to automate software testing. A JUnit test file `PigTest.java` has been provided on BlackBoard. Use the instructions from Project 2 for additional information about JUnit Testing

1. Name your class `Pig`. **Exact spelling is required**. Using a different name will fail the tests.
2. Complete all methods described in Step 3 and Step 4.
3. Download `PigTest.java` to the same folder as your `Pig.java`
4. Restart Bluej (if PigTest does not show up on your BlueJ window). BlueJ should recognize PigTest as a "<<unit test>>" file.

Sample Results

The following sample shows partial results of a game. Your messages can be more creative as long as they convey the necessary information. Note: totals turns is only displayed during automated games.

```
Welcome to the Game of Pig!


Round Score: 0
---- Your score: 0


Round Score: 6
Round Score: 17
Round Score: 0
---- Computer score: 0


Round Score: 6
Round Score: 13
Round Score: 22
---- Your score: 22
```

later in the game…

```
Round Score: 8
Round Score: 16
Round Score: 24
---- Your score: 92


Round Score: 7
```

```
Round Score: 18
Round Score: 23
---- Computer score: 73

Round Score: 9
---- Your score: 101

You won!
Total turns: 12
```

Step 5: Adapt for a GUI (15 pts)

Now that you have the basic game working within it's own class it is time to create a more interesting graphical user interface for the player to use. Add the following methods to the Pig class.

- Add a `boolean` instance variable to indicate if it is currently the player's turn. You will need to make several changes throughout the Pig class to set this variable to `true` when it is the player's turn and `false` when it is the computer's turn. All games start with the player.
- `public boolean isPlayerTurn ( )` - this one line method returns `true` if it is the player's turn or `false` if it is the computer's turn.

Rather than writing your own GUI class, we are providing an almost complete class to get you started. Create a new class in BlueJ called `GUI` and delete all of the provided code. Cut and paste the provided code from (GUI.java) into the newly created class. It should compile with no errors. If it doesn't compile then it probably means you used different names for your methods than what was specified.

Make the following changes to the GUI class

- Change the `JFrame` title to include your name
- Add statements in the constructor to register the hold and computer buttons with the ActionListener. Read the internal comment that shows where to add the statements.

Make the following changes to the `actionPerformed()` method. Read the internal comments for clues.

- Update the three labels with the current scores.

```
round.setText("Round: " + game.getRoundScore());
```

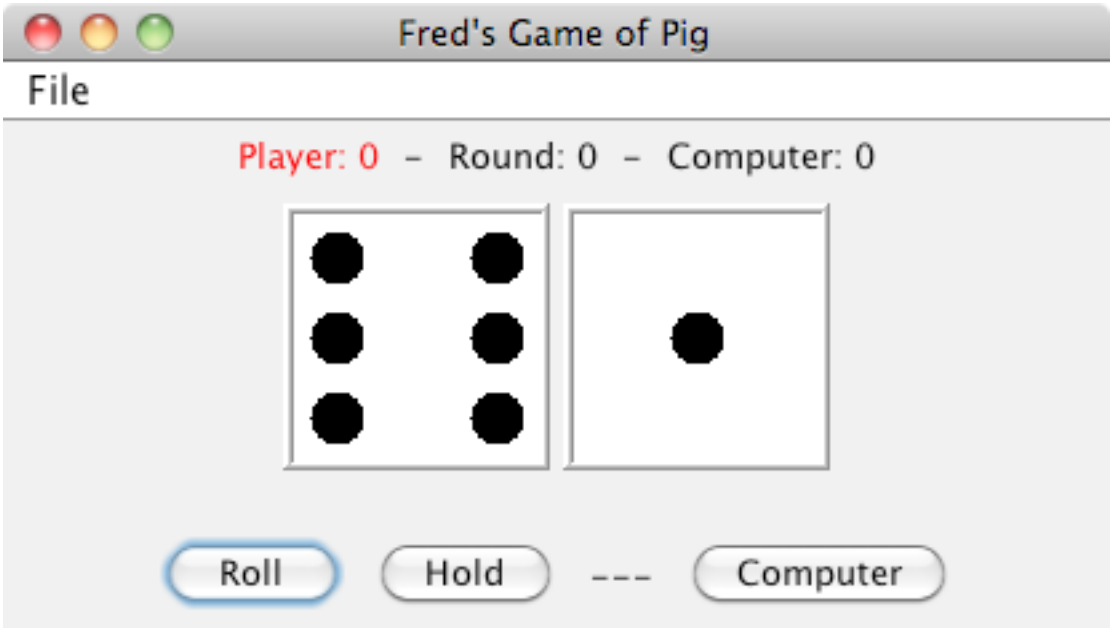
- Add if statements for each of the buttons and invoke the appropriate game method:

```
if (buttonPressed == roll)
     game.playerRolls();
```

- Display a pop up message after either the player or computer wins. Use a `JOptionPane.showMessageDialog` as described in section 6.6.

```
if (game.playerWon()){
     // display message
}
```

Sample GUI



Grading Criteria

There is a 50% penalty on programming projects if your solution does not compile.

- Stapled cover page with your name and signed pledge. (-5 pts if missing)
- Project requirements as specified above. (90 pts)
- Elegant source code that follows the GVSU Java Style Guide. (10 pts)

Late Policy

Projects are due at the START of the class period. However, you are encouraged to complete a project even if you must turn it in late.

- The first 24 hours (-20 pts)
- Each subsequent weekday is an additional -10 pts
- Weekends are free days and the maximum late penalty is 50 pts.

Turn In

A professional document **is stapled** with an attractive cover page. Do not expect the lab to have a working stapler!

- Cover page - Your project must have a cover page that includes your name, a title, an interesting graphic or photograph related to the project topic and the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " You are responsible for understanding and adhering to the School of CIS Guidelines for Academic Honesty.
- Source code - a printout of your elegant source code for the `Pig` and `GUI` classes. You do not need to provide the `GVdie` class.
- Demo – be prepared to demo your project on a lab computer or your laptop. I will ask you to perform a variety of tasks including play the game and invoke the `autoPlay()` method.