

Rock, Paper, Scissors, Lizard, Spock Game

Create a windows form application letting the user play the game Rock, Paper, Scissors, Lizard, Spock against the computer. The program should work as follows:

1. When the game begins, a random number in the range of 1 through 5 is generated for the computer's choice – DO NOT SHOW THIS VALUE YET!
 - a. If the computer/player chooses 1, they have chosen Rock
 - b. If the computer/player chooses 2, they have chosen Paper
 - c. If the computer/player chooses 3, they have chosen Scissors
 - d. If the computer/player chooses 4, they have chosen Lizard
 - e. If the computer/player chooses 5, they have chosen Spock
2. The user must select from the 5 choices (Rock, Paper, Scissors, Lizard, Spock) each time before they play the game. It is your choice how you get the player's input (radiobutton, textbox, etc.).
 - a. Do not use defaults – The user MUST make a choice each time.
 - b. The user should have the textual choices to choose from NOT the number choice. I.e., do not have the user select 1, 2, 3, etc., they should select Rock, Paper, Scissors, etc.
3. Include an area on the form showing running totals of wins, losses, and ties.
 - a. The output will be like the following replacing the <> with actual values:
 - i. Total Games Played: <total played since starting the program>
 - ii. Player Won: <total games player won>.
 - iii. Computer Won: <total games computer won>
 - iv. Ties: <total ties>
 - b. Example output might look like the following:
 - i. Total Games Played: 8
 - ii. Player Won: 3
 - iii. Computer Won: 2
 - iv. Ties: 3
4. There should be the following buttons on the form. Place them on the form in the order/location you feel works best for your expected user. You might consider grouping the buttons in different areas instead of just listing them all in order at the bottom, top, side of your form.
 - a. About. When clicked, this button at a minimum will provide the following information to the user (replace the information inside the <> with appropriate information):
 - i. Created By <your name>
 - ii. Created For < the name of this class>
 - iii. Date Created < the date you created the program>
 - b. Winning Combinations. When clicked this will show the following winning combinations so it is easy for your user to identify all the winning combinations. Hint: mashing all this in one line with no punctuation is not easy for a user. You may read in this information from a file (see chapter 5) or hard code it in your program.
 - i. Scissors cuts paper
 - ii. Paper covers rock
 - iii. Rock crushes lizard
 - iv. Lizard poisons Spock

- v. Spock smashes scissors
 - vi. Scissors decapitates lizard
 - vii. Lizard eats paper
 - viii. Paper disproves Spock
 - ix. Spock vaporizes rock
 - x. Rock crushes scissors
- c. Background. When clicked this will show the following background information. You may read in this information from a file (see chapter 5) or hard code it in your program.
- i. Rock Paper Scissors Lizard Spock is an extension of the classic game of chance, Rock Paper Scissors, created by Sam Kass and Karen Bryla in 1998. This was created because it seems like when you know someone well enough, 75-80% of any Rock-Paper-Scissors games you play with that person end up in a tie. This is a slight variation reducing that probability and this version is nice because it satisfies the Law of Fives. In the Big Bang Theory, it is first used to settle a dispute about what to watch on TV between Sheldon and Raj in the "The Lizard-Spock Expansion" and then mentioned again in the "The Rothman Disintegration", where Sheldon explains the rules to Penny and Barry Kripke.
- d. Reset. When clicked you should reset everything back to what it was when the user first started the program. This way the player does not have to exit the game and restart it just to start over.
- i. I.e., zero games played/won/tied, etc.
- e. Exit. When clicked this exits your program.
- f. Play. When clicked AND the user has not selected a choice, tell them they need to make a choice before they can play the game OR you can disable the Play button until they make a choice (disabling it after each game). When clicked AND the user has selected a choice the program will use the player's and computers choices to determine a winner and display that information to the player based on the below guidance:
- i. The output will be like the following (replacing the information between the <> with actual information) depending on the game's outcome:
 - 1. When the player wins: Player Wins - <use the winning combination text>!
 - a. For example, if Player selected Rock and Computer selected Scissors the display would read:
Player Wins - Rock crushes scissors!
 - 2. When the computer wins: OH NO! Player loses and Computer Wins! - <use the winning combination text>!
 - a. For example, if Player selected Scissors and Computer selected Rock the display would read:
OH NO! Player loses and Computer Wins! - Rock crushes scissors!
 - 3. When there is a tie: It is a Tie! Player and Computer both selected <players/computers selection>. Try again!
 - a. For example, if Player and Computer both selected Scissors the display would read:

It is a Tie! Player and Computer both selected Scissors. Try again!

- g. After the winner has been displayed, make sure you select the computers next choice (again do not display it to the player yet), update the running totals, and clear the user's choice so they can play again.
- 5. You may adjust your form or control's color, fonts, etc. but make sure these do not distract from playing the game or reading the form's text. You may also add appropriate pictures to your form again if it does not distract from playing the game or reading the form's text.
- 6. Make sure you use what you have learned in chapters 1 – 6 in your program at a minimum making sure:
 - a. To validate all user entry.
 - b. To include exception handling and appropriate error messages so users know what they did wrong and can fix it.
 - c. To comment your variables, methods, and processing so reviewers know what you are doing in your program.
 - d. Your tab order is correct on your form and would make sense to your user.
 - e. To name your game's form, buttons, labels, and controls appropriately, so the user does not have to guess what they are.
 - f. To name your methods, variables, and controls appropriately in code, so reviewers do not have to guess what they are.
 - g. To modularize the program into methods performing each major task.
- 7. **DO NOT include any functionality not asked for above. If you have questions or feel something is missed in the above information, ask!**