WAR PROGAMMING ASSIGNMENT:
THE BREAKING BAD EDITION

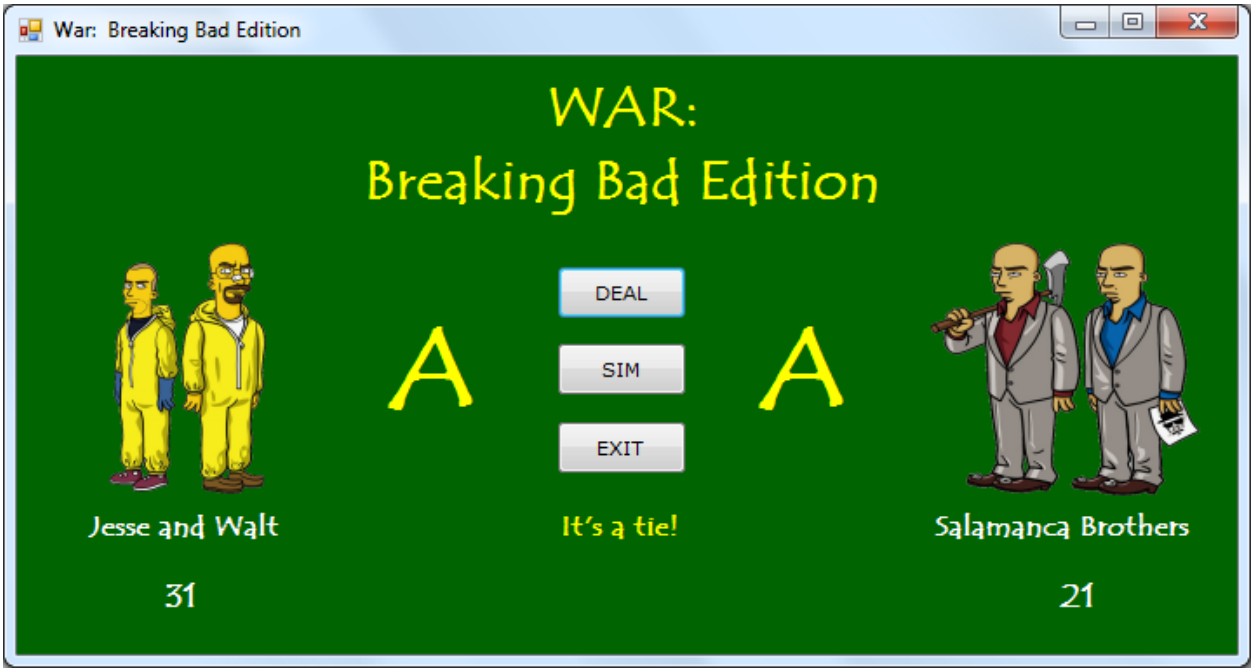
Your assignment is to create a modified version of the card game **War**. In this version you will be creating, each player starts with 26 points. When the game begins a random number between 2 and 14 will be generated for both players, where 11 is a JACK, 12 is a QUEEN, 13 is a KING, 14 is an ACE and the remaining cards are worth their face value. Whoever gets the higher card, wins a point. The loser with the lower card loses a point. In the case of a tie (that is, the same random number is generated for both players), nobody earns or loses a point. This process continues until one of the players loses all of his/her points. The winner is the person who ends up with 52 points.

The following interface has been provided for you. All you need to do is write the code for the three buttons (i.e. DEAL, SIM, and EXIT).

When the program begins, each player's score should be set to 26 and a message should appear at the bottom instructing the user to click **DEAL** or **SIM** to start the game.

When the user clicks the **DEAL** button, a random number is generated for both players and outputted in the labels provided. The program must then check to see who won the hand, add a point to the winner of the hand, deduct a point from the loser, and output the winner of the hand in the label provided.

In the case of a tie, a message should be outputted in the label indicating it was a tie.

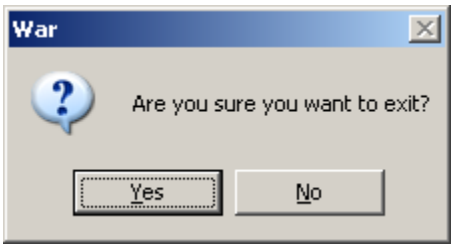


The same process repeats itself each time the user clicks the DEAL button until one of the players has zero points left. Once a winner is declared, the DEAL and SIM buttons should be disabled and a message should be outputted in the label provided indicating who won the game.

The option the user has at this point is to click **EXIT**. When the user clicks **EXIT**, a confirm dialog should be displayed asking the user if he/she wants to play again.

If the user chooses **YES**, the scores need to be reset to 26 and all components need to be set to their original state. If the user chooses **NO**, the following message should be displayed before the program terminates.

If the user clicks the **EXIT** button during game play (i.e. before the game is over), the following message box should be outputted:



If the user chooses **YES**, the game should resume where it left off. If the user chooses NO, the "Thank you for playing War!" message should be outputted to the user before the program terminates.

You'll notice there is also a **SIM** button. The purpose of the SIM button is to allow the program to simulate a game without the user always having to click DEAL. In other words, when the user clicks the SIM button, the program continuously deals cards to both players, updates the score, etc. until one player loses all his points. In order to make this work, you will need to use a **Do While** or **Do Until** loop that continuously deals until a winner is declared.

Once you have finished the program, save the program in a folder called **War** in your **COMPLETED ASSIGNMENTS** folder.

WAR PROGRAM RUBRIC

NAME: _____ TOTAL: / 40

CATEGORY	CRITERIA	LEVEL 1 50 – 59%	LEVEL 2 60 – 69%	LEVEL 3 70 – 79%	LEVEL 4 80 – 100%	MARK
Knowledge and Understanding	Demonstrates an understanding of how to write a program(s) that uses decision structures and looping structures	• Demonstrates limited under-standing of how to use decision structures and loops **5.0-5.9**	• Demonstrates some understand-ing of how to use decision structures and loops **6.0-6.9**	• Demonstrates considerable understanding of how to use decision structures and loops **7.0-7.9**	• Demonstrates thorough understanding of how to use decision structures and loops **8.0-10**	**/10**
Thinking	The program meets all the required specifications Validates program to ensure the program produces correct results	• Program meets a limited number of the required specifications • Validates program with limited success **5.0-5.9**	• Program meets some of the required specifications • Validates program with some success **6.0-6.9**	• Program meets most of the required specifications • Validates program with considerable success **7.0-7.9**	• Program meets all of the required specifications • Validates program with great success **8.0-10**	**/10**
Communication	Provides internal documentation that clearly explains program logic	• Documents program logic with limited success **5.0-5.9**	• Documents program logic with some success **6.0-6.9**	• Documents program logic with considerable success **7.0-7.9**	• Documents program logic with great success **8.0-10**	**/10**
Application	Effectively applies programming skills and knowledge of Visual Basic to create a program	• Applies programming knowledge and skills with limited success **5.0-5.9**	• Applies programming knowledge and skills with some success **6.0-6.9**	• Applies programming knowledge and skills with considerable success **7.0-7.9**	• Applies programming knowledge and skills with great success **8.0-10**	**/10**

CURRICULUM EXPECTATIONS THAT ARE COVERED IN THIS ASSIGNMENT:

B1.4 Determine the expressions and instructions to use in a programming statement, taking into account the order of operations.
B1.5 Identify situations in which decisions and looping structures are required.
B2.2 Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program.
B2.3 Write keyboard input and screen output statements that conform to program specifications.
B2.4 Write a program that includes a decision structure for two or more choices.
B2.5 Write programs that use looping structures effectively.
B3.1 Write clear and maintainable code using proper programming standards.
B3.2 Write clear and maintainable internal documentation to a specific set of standards.
B3.3 Use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program.
B3.4 Demonstrate the ability to validate a computer program using test cases.