FULL SAIL UNIVERSITY

LAB: BLACKJACK GAME

CONTENTS

OBJECTIVE

Add the BlackjackGame class to the FullSailCasino project.

You are going to create a Blackjack game. Add the functionality for the "Play Blackjack" menu option.

NOTE: If you do not have the Blackjack Objects Lab finished to the point of using it in the Project, then you can use the mock classes provided in the mock.zip file. After unzipping, drag-n-drop the files onto your class library. Using these mock classes will allow you to move forward with the game logic. Keep in mind that these classes are incomplete – they only provide the API (application programming interface) so you can use the objects in your game logic code. **However, using them will result in a points deduction of 15 points**. But that's better than nothing.

NOTE: Blackjack also contains additional rules such as splitting, doubling, and surrendering, but these are **not required** for this project.

FULL SAIL UNIVERSITY

THE BLACKJACKGAME CLASS

Create a **BlackjackGame** class in your class library.

FIELDS

NAME	TYPE	COMMENTS
_dealer	BlackjackHand	This is your dealer hand.
_player	BlackjackHand	This is your player hand.
_deck	Deck	This is your deck of cards.

METHODS

NAME	RETURNS	PARAMETERS	COMMENTS
PlayRound	void		This method is responsible for the overall game logic for a round of blackjack. It will call the other methods. Initialize the fields (_dealer, _player, _deck) to new instances of each type.
DealInitialCards	void		Deals 2 cards to the player and dealer in proper dealing order. See the DealInitialCards section for more details.
PlayersTurn	void		This method handles the player's turn logic. See the PlayersTurn section for more details.
DealersTurn	void		This method handles the dealer's turn logic. See the DealersTurn section for more details.
DeclareWinner	void		This method will show the result of the round of blackjack. See the DeclareWinner section for more details.
DrawTable	Void		Clears the screen and draws the player hand and dealer hand at specific places in the console.

PlayRound Method

This method should first create new instances of each field: _dealer, _player, _deck. Then the method should call each of the other methods: 1) DealInitialCards, 2) PlayersTurn, 3) DealersTurn, 4) DeclareWinner.

After calling **DealInitialCards**, check if either the player or dealer have 21. If **no one** has 21 call the turn methods: **PlayersTurn** then **DealersTurn**,

Afterwards, call the **DeclareWinner**.

FULL SAIL UNIVERSITY

DealInitialCards Method

This method will deal the initial 2 cards to everyone at the table.

Dealing cards has a specific order: deal around the table 1 card to everyone starting with the player and ending with the dealer. Stop after everyone has 2 cards. Ex: deal 1 card to player, deal 1 card to dealer, deal 1 card to the player, deal 1 card to the dealer.

PlayersTurn Method

This method has the logic for the player's turn.

While the player's score is < 21, you ask them a question: Hit or Stand? If the player chooses to hit, add another card to the hand. If the player chooses to stand, exit the loop.

NOTE: use **ReadChoice** to get the user's input.

DealersTurn Method

This method has the logic for the dealer's turn.

Remember to show all the dealer's cards at this stage of the round.

The dealer's logic is simple – add cards to the dealer's hand while the dealer's score is less than 17.

DeclareWinner Method

This method will show the result of the round of blackjack. Compare the scores of the _dealer and _player.

1. Player's score > 21: dealer wins
2. Dealer's score > 21: player wins
3. Player's score EQUALS dealer's score: no one wins (it's a tie)
4. Player's score > Dealer's score: player wins
5. Dealer wins

DrawTable Method

You'll want your game to look nice, so everything should show up correctly and in a place that makes sense.

Only reveal the dealer's complete hand and score after the player's turn is over.

Clearly label the player's hand and dealer's hand in the console.

To make sure everything shows up in a predictable place in the console, set the cursor position before printing anything. Determine ahead of time where (x,y coordinates) to draw the player, dealer, hit/stand, play again, and winner.

This method will draw the table for the game. It should clear the screen before drawing the table. It should call the draw method of the player hand and the dealer hand. It should clearly label each hand.

Call this method any time a card is added to a hand.

THE MAIN MENU

To complete the game, you'll need to add code to **case 1** of the main menu switch statement.

- Create a BlackjackGame instance.
- Start a loop and call the PlayRound of the BlackjackGame instance.
- After PlayRound returns, ask the user if they want to play again. (HINT: use ReadChoice)
- If they choose no, exit the loop.

FULL SAIL UNIVERSITY

RUBRIC

FEATURE	VALUE	GRADE
BlackjackGame Class	5	
PlayRound method	10	
DealInitialCards method	10	
PlayersTurn method	20	
DealersTurn method	20	
DeclareWinner method	20	
DrawTable method	10	
Main Menu	5	
TOTAL	100	

Common mistakes:

-100: choosing to not follow the lab requirements and instead use code from the internet.

-15: using the mock classes.

-3: showing all the dealer's cards and/or score before the dealer's turn

-3: not revealing the dealer's cards and/or score after the player's turn is over.

-3: items drawn to the screen are overlapping

-2: not ending the player's turn if the player's score is 21

-2: not ending the dealer's turn if the dealer's score >= 17

-5: not asking the user if they want to play again

-5: not using ReadChoice for the user's input in the game.

PROGRAMMER'S CHALLENGE

As with every programmer's challenge, remember the following…

1. Do the rubric first. Make sure you have something to turn in for the assignment.
2. When attempting the challenge, don't break your other code.
3. You have other assignments so don't sacrifice them to work on the challenges.

FULL SAIL UNIVERSITY

Challenge 1: Multiple Player's

Add support for multiple players.

Challenge 2: Betting for each player

Add the ability for each player to place a bet. This would probably require you to create a Player class that has info about the player's name, current monies, and current bet.