CS 31 : Introduction To Computer Science I

Howard A. Stahl

Project 7

- The Goal: A Working Blackjack Game
- Background: Please Play A Few Games With This Free Game
  - http://www.wizardsofodds.com/play/blackjack
- Truth In Advertising:
  - We'll Only Be Dealing With The Following Concepts: Game, Player, Dealer, Card, Deck, Hit, Stand
  - No Need To Worry About Betting, Splitting, Double-Down

Project 7

- Unlike Earlier Assignments, I Am Supplying You With A Partial "Skeleton" Of The Code Solution
- It Will Run Right Out Of The Box
  - Some Important Pieces Are Stubbed Out…
  - These Are The Parts You Need To Complete
- Hint 1: Acquire The Skeleton!
- Hint 2: Build And The Run The Skeleton!
  - Look At What Is Working And What Is Not

Project 7

- The Work Product: The Implementation Of The Public API Of The Classes Described Here And In The Assignment.
- You Are Free To Do It However You Like, But You Must Provide The Public API I Am Looking For…
  - You Can Add Classes, Methods, Members As You Feel Appropriate
  - But I Honestly Don't Think You'll Need To…
- In What Follows, It Is The **Bolded** Portions That You Need To Complete

Some Old Friends…

- Using The RandomNumber Class, We'll Have Decks Of Cards That Shuffles Randomly, Like In The Real World…

RandomNumber
-mMinimum : int -mMaximum : int
+RandomNumber(min : int, max : int, minInclusive : bool = true, maxInclusive : bool = true) +random( ) : int

Some Old Friends…

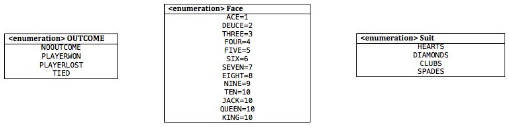
- Using The RandomNumber Class, We'll Have Decks Of Cards That Shuffles Randomly, Like In The Real World…

RandomNumber
-mMinimum : int -mMaximum : int
+RandomNumber(min : int, max : int, minInclusive : bool = true, maxInclusive : bool = true) +random( ) : int

YAY! Ain't Nothing To Do Here…

Some Old Friends…

- I Really Like Enumerations…



<enumeration> OUTCOME
NOOUTCOME
PLAYERWON
PLAYERLOST
TIED

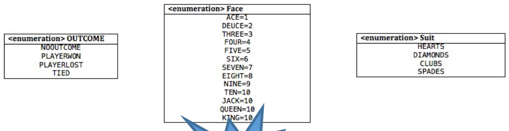
<enumeration> Face
ACE=1
DEUCE=2
THREE=3
FOUR=4
FIVE=5
SIX=6
SEVEN=7
EIGHT=8
NINE=9
TEN=10
JACK=10
QUEEN=10
KING=10

<enumeration> Suit
HEARTS
DIAMONDS
CLUBS
SPADES

- OUTCOME Is Used To Represent The Result Of Playing A Game
- Face And Suit Are Used To Represent One Playing Card

Some Old Friends…

- I Really Like Enumerations…



<enumeration> OUTCOME
NOOUTCOME
PLAYERWON
PLAYERLOST
TIED

<enumeration> Face
ACE=1
DEUCE=2
THREE=3
FOUR=4
FIVE=5
SIX=6
SEVEN=7
EIGHT=8
NINE=9
TEN=10
JACK=10
QUEEN=10
KING=10

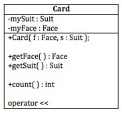
<enumeration> Suit
HEARTS
DIAMONDS
CLUBS
SPADES

- OUTCOME Is Used To Represent The Result Of Playing A Game
- Face And Suit Are Used To Represent One Playing Card

YAY! Ain't Nothing To Do Here…

The Card Class

- A Card Has A Face And Suit Value
  - getters Have Been Provided
  - int count( )
    - Supply The Value Of Each Individual Card
  - operator <<
    - Displays Textually The Value Of A Card To Passed Stream

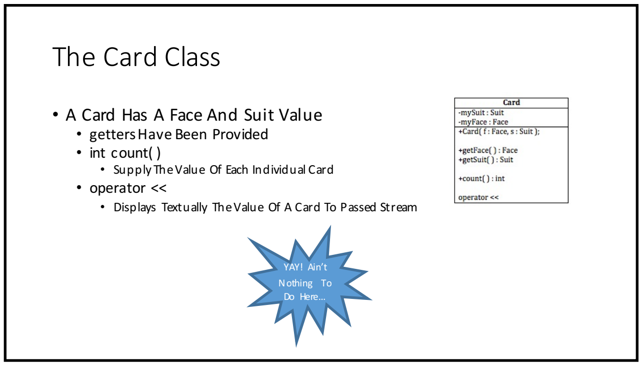


Card
-mySuit : Suit
-myFace : Face
+Card(f : Face, s : Suit);

+getFace( ) : Face
+getSuit( ) : Suit

+count( ) : int

operator <<

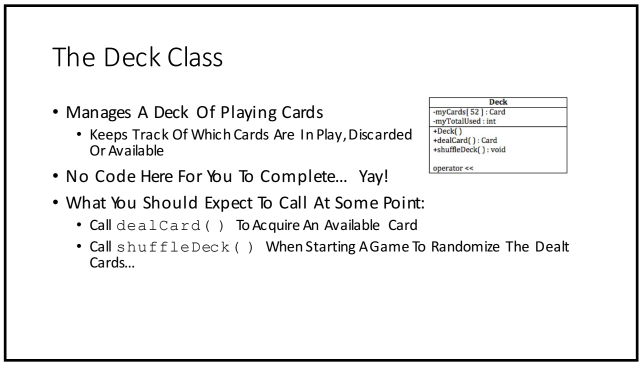


The Card Class

- A Card Has A Face And Suit Value
  - getters Have Been Provided
  - int count( )
    - Supply The Value Of Each Individual Card
  - operator <<
    - Displays Textually The Value Of A Card To Passed Stream

```
Card
-mySuit : Suit
-myFace : Face
+Card( f : Face, s : Suit );

+getFace( ) : Face
+getSuit( ) : Suit

+count( ) : int

operator <<
```

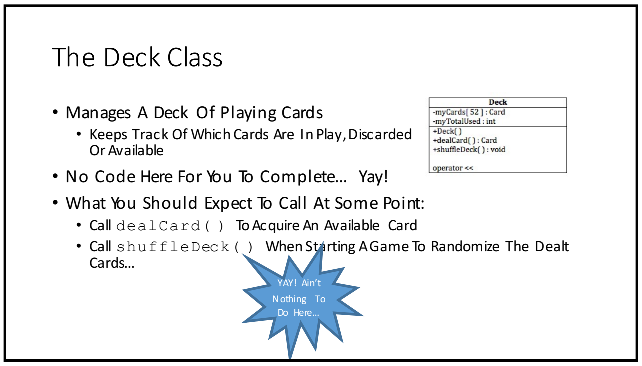
YAY! Ain't Nothing To Do Here…



The Deck Class

- Manages A Deck Of Playing Cards
  - Keeps Track Of Which Cards Are In Play, Discarded Or Available
- No Code Here For You To Complete… Yay!
- What You Should Expect To Call At Some Point:
  - Call `dealCard( )` To Acquire An Available Card
  - Call `shuffleDeck( )` When Starting A Game To Randomize The Dealt Cards…

```
Deck
-myCards[ 52 ] : Card
-myTotalUsed : int
+Deck( )
+dealCard( ) : Card
+shuffleDeck( ) : void

operator <<
```



The Deck Class

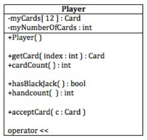
- Manages A Deck Of Playing Cards
  - Keeps Track Of Which Cards Are In Play, Discarded Or Available
- No Code Here For You To Complete… Yay!
- What You Should Expect To Call At Some Point:
  - Call `dealCard( )` To Acquire An Available Card
  - Call `shuffleDeck( )` When Starting A Game To Randomize The Dealt Cards…

```
Deck
-myCards[ 52 ] : Card
-myTotalUsed : int
+Deck( )
+dealCard( ) : Card
+shuffleDeck( ) : void

operator <<
```

YAY! Ain't Nothing To Do Here…

The Player Class

- The Player Holds A Number Of Cards And Might Have BlackJack!
- A Player Stores Its Cards In An Array Of Card Named `myCards[ ]`



Player

-myCards[12] : Card
-myNumberOfCards : int

+Player( )

+getCard(index : int) : Card
+cardCount( ) : int

+hasBlackJack( ) : bool
+handcount( ) : int

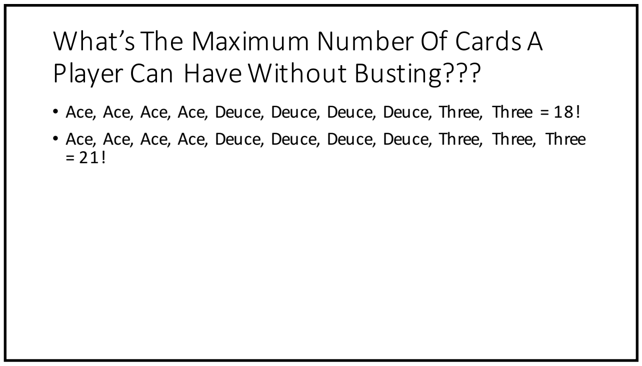
+acceptCard(c : Card)

operator <<

What's The Maximum Number Of Cards A Player Can Have Without Busting???

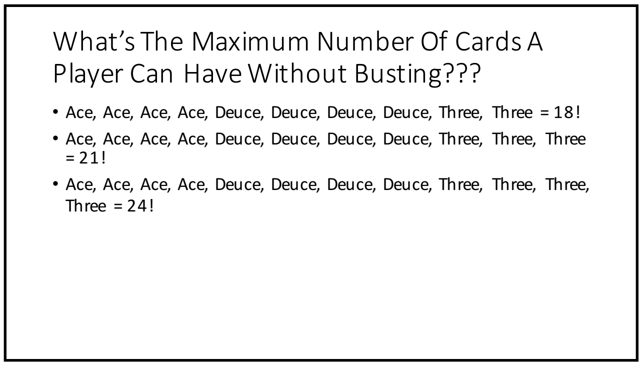
What's The Maximum Number Of Cards A Player Can Have Without Busting???

- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three = 18!



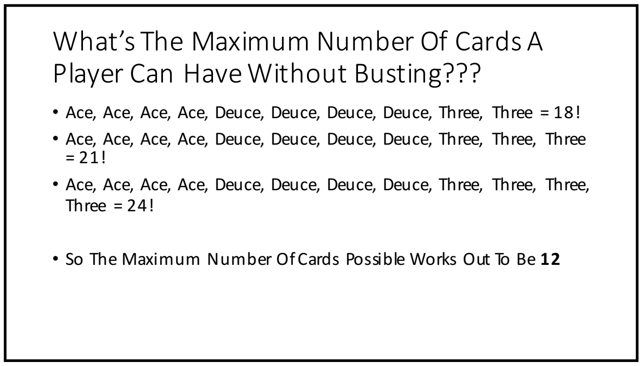
What’s The Maximum Number Of Cards A Player Can Have Without Busting???

- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three = 18!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three = 21!



What’s The Maximum Number Of Cards A Player Can Have Without Busting???

- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three = 18!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three = 21!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three, Three = 24!



What’s The Maximum Number Of Cards A Player Can Have Without Busting???

- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three = 18!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three = 21!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three, Three = 24!

- So The Maximum Number Of Cards Possible Works Out To Be **12**

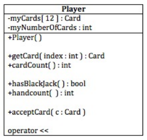
What's The Maximum Number Of Cards A Player Can Have Without Busting???

- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three = 18!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three = 21!
- Ace, Ace, Ace, Ace, Deuce, Deuce, Deuce, Deuce, Three, Three, Three, Three = 24!

- So The Maximum Number Of Cards Possible Works Out To Be **12**
- In All Likelihood, A Player Will Bust With Many Fewer Cards!!

The Player Class

- The Player Holds A Number Of Cards And Might Have BlackJack!
- A Player Stores Its Cards In An Array Of Card Named `myCards[ ]`
  - Maximum Size Of The Array Is 12
  - But Cards Are Dealt One At A Time... So The Array Will Be Partially Full And "Grow" Over Time...



Player

-myCards[12] : Card
-myNumberOfCards : int

+Player( )

+getCard(index : int) : Card
+cardCount( ) : int

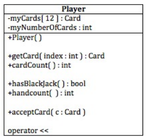
+hasBlackJack( ) : bool
+handcount( ) : int

+acceptCard(c : Card)

operator <<

The Player Class

- The Player Holds A Number Of Cards And Might Have BlackJack!
- A Player Stores Its Cards In An Array Of Card Named `myCards[ ]`
  - Maximum Size Of The Array Is 12
  - But Cards Are Dealt One At A Time... So The Array Will Be Partially Full And "Grow" Over Time...
- `myNumberOfCards` Tells How Many Of The `myCards` Elements Are In Play



Player

-myCards[12] : Card
-myNumberOfCards : int

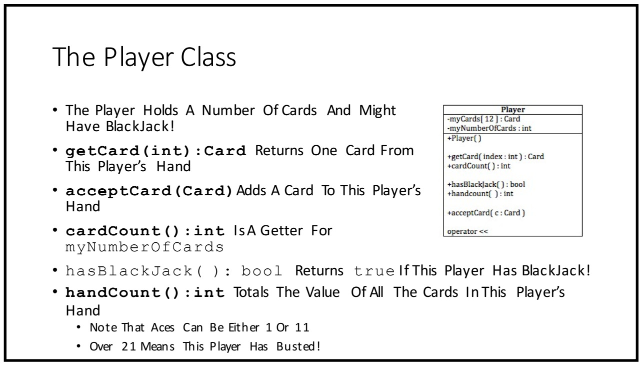
+Player( )

+getCard(index : int) : Card
+cardCount( ) : int

+hasBlackJack( ) : bool
+handcount( ) : int

+acceptCard(c : Card)

operator <<



The Player Class

- The Player Holds A Number Of Cards And Might Have BlackJack!
- **`getCard(int):Card`** Returns One Card From This Player's Hand
- **`acceptCard(Card)`** Adds A Card To This Player's Hand
- **`cardCount():int`** Is A Getter For `myNumberOfCards`
- `hasBlackJack( ): bool` Returns `true` If This Player Has BlackJack!
- **`handCount():int`** Totals The Value Of All The Cards In This Player's Hand
  - Note That Aces Can Be Either 1 Or 11
  - Over 21 Means This Player Has Busted!

Player

-myCards[12] : Card
-myNumberOfCards : int

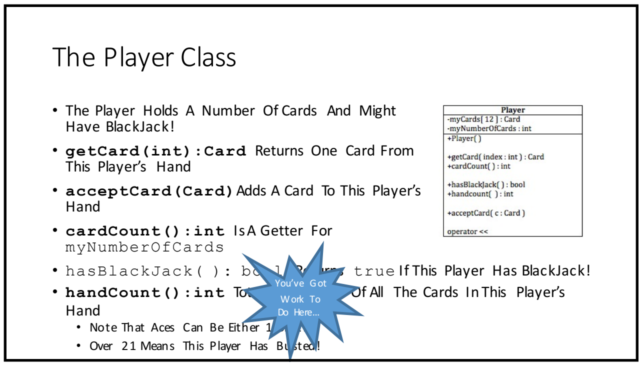
+Player( )

+getCard(index : int) : Card
+cardCount( ) : int

+hasBlackJack( ) : bool
+handcount( ) : int

+acceptCard(c : Card)

operator <<



The Player Class

- The Player Holds A Number Of Cards And Might Have BlackJack!
- **`getCard(int):Card`** Returns One Card From This Player's Hand
- **`acceptCard(Card)`** Adds A Card To This Player's Hand
- **`cardCount():int`** Is A Getter For `myNumberOfCards`
- `hasBlackJack( ): bo` `true` If This Player Has BlackJack!
- **`handCount():int`** To Of All The Cards In This Player's Hand
  - Note That Aces Can Be Either 1
  - Over 21 Means This Player Has Bu ted!

You've Got Work To Do Here…

Player

-myCards[12] : Card
-myNumberOfCards : int

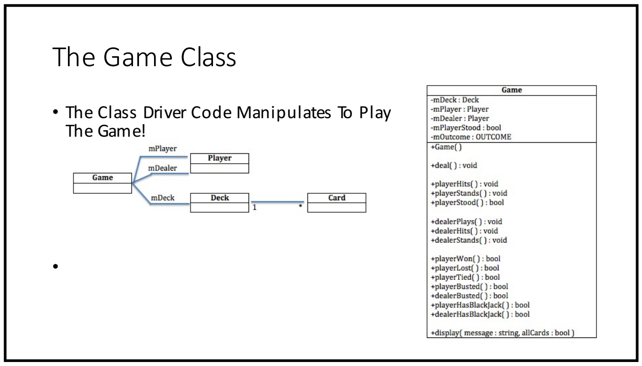
+Player( )

+getCard(index : int) : Card
+cardCount( ) : int

+hasBlackJack( ) : bool
+handcount( ) : int

+acceptCard(c : Card)

operator <<



The Game Class

- The Class Driver Code Manipulates To Play The Game!
-

Game

-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME

+Game( )

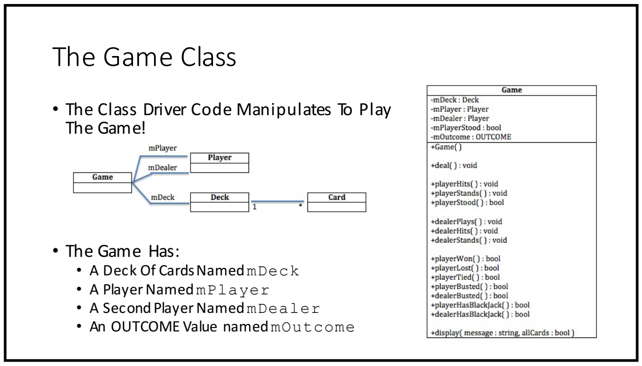
+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

+display(message : string, allCards : bool)



The Game Class

- The Class Driver Code Manipulates To Play The Game!

- The Game Has:
  - A Deck Of Cards Named `mDeck`
  - A Player Named `mPlayer`
  - A Second Player Named `mDealer`
  - An OUTCOME Value named `mOutcome`

Game

-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME

+Game( )

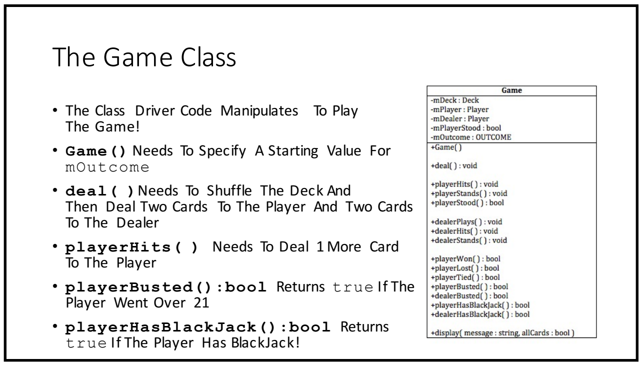
+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

+display(message : string, allCards : bool)



The Game Class

- The Class Driver Code Manipulates To Play The Game!
- **`Game()`** Needs To Specify A Starting Value For `mOutcome`
- **`deal( )`** Needs To Shuffle The Deck And Then Deal Two Cards To The Player And Two Cards To The Dealer
- **`playerHits( )`** Needs To Deal 1 More Card To The Player
- **`playerBusted():bool`** Returns `true` If The Player Went Over 21
- **`playerHasBlackJack():bool`** Returns `true` If The Player Has BlackJack!

Game

-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME

+Game( )

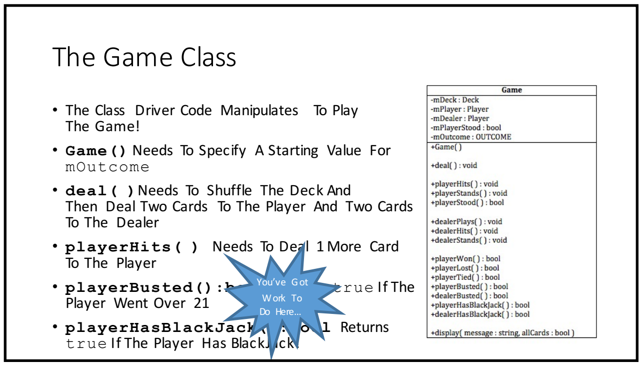
+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

+display(message : string, allCards : bool)



The Game Class

- The Class Driver Code Manipulates To Play The Game!
- **`Game()`** Needs To Specify A Starting Value For `mOutcome`
- **`deal( )`** Needs To Shuffle The Deck And Then Deal Two Cards To The Player And Two Cards To The Dealer
- **`playerHits( )`** Needs To Deal 1 More Card To The Player
- **`playerBusted():b`** `true` If The Player Went Over 21
- **`playerHasBlackJack`** **`ol`** Returns `true` If The Player Has BlackJ ck

Game

-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME

+Game( )

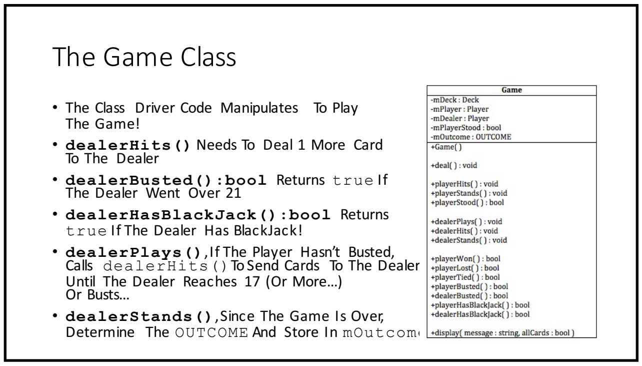
+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

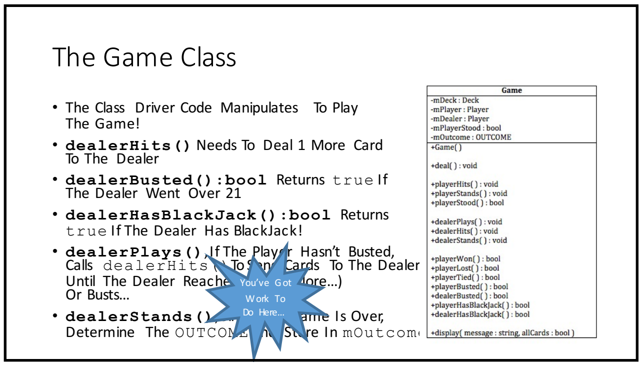
+display(message : string, allCards : bool)



The Game Class

- The Class Driver Code Manipulates To Play The Game!
- **`dealerHits()`** Needs To Deal 1 More Card To The Dealer
- **`dealerBusted():bool`** Returns `true` If The Dealer Went Over 21
- **`dealerHasBlackJack():bool`** Returns `true` If The Dealer Has BlackJack!
- **`dealerPlays()`**, If The Player Hasn't Busted, Calls `dealerHits()` To Send Cards To The Dealer Until The Dealer Reaches 17 (Or More…) Or Busts…
- **`dealerStands()`**, Since The Game Is Over, Determine The `OUTCOME` And Store In `mOutcom`

```
Game
-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME
+Game( )

+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

+display( message : string, allCards : bool )
```



The Game Class

- The Class Driver Code Manipulates To Play The Game!
- **`dealerHits()`** Needs To Deal 1 More Card To The Dealer
- **`dealerBusted():bool`** Returns `true` If The Dealer Went Over 21
- **`dealerHasBlackJack():bool`** Returns `true` If The Dealer Has BlackJack!
- **`dealerPlays()`**, If The Play r Hasn't Busted, Calls `dealerHits` To Se d Cards To The Dealer Until The Dealer Reache ore…) Or Busts…
- **`dealerStands()`** ame Is Over, Determine The `OUTCOM` St re In `mOutcom`

```
Game
-mDeck : Deck
-mPlayer : Player
-mDealer : Player
-mPlayerStood : bool
-mOutcome : OUTCOME
+Game( )

+deal( ) : void

+playerHits( ) : void
+playerStands( ) : void
+playerStood( ) : bool

+dealerPlays( ) : void
+dealerHits( ) : void
+dealerStands( ) : void

+playerWon( ) : bool
+playerLost( ) : bool
+playerTied( ) : bool
+playerBusted( ) : bool
+dealerBusted( ) : bool
+playerHasBlackJack( ) : bool
+dealerHasBlackJack( ) : bool

+display( message : string, allCards : bool )
```

You've Got Work To Do Here…