Crazy Eights Test Plan

Controllers:

PlayerController

Method	Tested?	Test Notes	User Story Covered
PlayerController(Player* player)	☐	- Test Null Player Pointer - Called **once** for each player	-
sortHand() : virtual void	☐	- Called **once** every time a new card is added to the hand - Test sorting by different Suits, - Test sorting by different Ranks **How to test:** - Add cards in random order to hand, call sortHand() and check if cards are sorted.	-
playCard() : virtual Card*	☐	- Called **once** per player turn **Mostly tested indirectly by methods in Player, Hand, Deck:** - Tested when card is removed from hand - Tested when card is added to discard correctly - Test when player tries to play an incorrect card not held by player - Tested in a situation where duplicate cards held by player	-
receiveCard(Card* card) : virtual void	☐	**Tested indirectly** by **addToHand()** for each player. - Test for null pointer.	-

Game

Method	Tested?	Test Notes	User Story Covered
Game(vector <PlayerController*>)	☐	- Called **once** for when the game is started. - Test for nullptr being passed to constructor. - Test for a valid pointer passed. Valid pointer will be **tested indirectly** with the	-

		other methods in Game.	
getPlayers() : virtual vector <PlayerController*>)	☐	- Called **once** before starting the main game loop. - Expected to return the number of players in the game.	#2 - Select number of players
getTopCard() : virtual Card*	☐	**Tested indirectly:** - By Deck::getTopCard - Test for null pointer.	#3/14/15 - Make player draw one/two/ four cards
isGameOver() : virtual bool	☐	- Called **twice** to test each state. - After one turn, check if anyone has an empty hand to test the true state. - Ensure everyone has cards in their hand. Call again to test the false state.	#11 - Win Condition
setUI(UI*) : virtual void	☐	**Tested Indirectly:** - Views will be mocked and will be switched in the game loop test.	-
startGame() : virtual void	☐	**Tested indirectly**: - Called **once** to start the game. - Using methods in Deck, Player and Hand classes.	#10 - Starting the game
endGame() : virtual void	☐	- Called **once** at the end of the game when a winner is decided. **Tested indirectly**: - Using methods in Deck, Player and Hand classes. Called either if the player quits or after isGameOver() is true.	#8 - End Game
swapDecks() : virtual void	☐	- Called **once** when the stock pile runs out. **How to test:** - Empty the stock pile, create a play pile that has more than one card. - Transfer all but the top card from the play pile to stock pile. - Shuffle stock pile. - Test if = Card left in play pile is the prev top card - Test if = sizeStockPile == (playPile - 1)	#9 - Stock is exhausted
determineWinner() : virtual void	☐	**Tested indirectly:** - Using methods in Deck, Player and Hand classes. Called after isGameOver() is true / a player's hand is empty.	

dealCards : virtual void()	☐	- Called **once** when dealing a card. **How to test - 2 players:** - Create a game with 2 players. - Call dealCards() and deal each player 7 cards. - Check the size of each player's hand. - The hand size should be 7. **How to test - >2 players:** - Create a game with 3 or more players. - Call dealCards() and deal each player 5 cards. - Check the size of each player's hand. - The hand size should be 5.	

Models:

Player

Method	Tested?	Test Notes	User Story Covered
Player(string name, int score, Hand* hand)	☐	- Test Null Hand Pointer - Test empty name - Test negative score - Test positive score - Called once for each player in the game	#2 - Select number of players
getHand() : Hand*	☐	- Test returns the correct pointer/cards - Called **once** each turn for a player **How to test:** - Create a game with 2 players, each player gets 7 cards. - Play 1 turn for both players. Each discards 1 card. - Call method and check if the number of cards in each player's hand are 6 each.	#7 - Play cards on turn #5 - Deal cards
getScore() : int	☐	**Tested indirectly in addToScore()** - Called once for **each** player after each round/game.	#8 - End game
getName() : string	☐	- Called **once** for each player after each round/game to display their name. - Test returns same string as in constructor	#8 - End game
addToScore(int score) : void	☐	- Called **once** after each round/game is over - Test the value is added correctly to the player's score. - Test negative score values **How to test:** - Create a game with 2 players, 7 cards each.	#6 - count winning player points

		Value of each hand is 50. - Player 1 plays 1 card worth 5 points. Value of hand 45. - Player 2 plays all their cards.. - Call **addToScore()** for each player. Add 45 to player 2's score as they won the game. - Call **getScore()** and check if player 2's score is 45.	
playCard() : virtual Card*	☐	- Pure Virtual, relevant implementations tested. **Tested indirectly**: - By methods in Player, PlayerController, Card, Hand	#7/12 - Play an 8/play card on turn

CI (Implementation of Player)

Method	Tested?	Test Notes	User Story Covered
CI(string name, int score, Hand* hand)	☐	- Same tests as Constructor for Player	N/A

playCard() : Card*	☐	- Called **once** for each player on their turn
		- Test valid card played.
		  - Play a card
		  - Call matches(). Check if the move is valid.
		- Test invalid card played.
		- Test when 8 is played and that 8s change suit.
		  - Play an 8 card.
		  - Call matches() to make sure the move is valid.
		  - Change the suit of the next card to be played.
		  - Call match() again to ensure the new suit can be played.
			See Player:: playCard()

Human (Implementation of Player)

Method	Tested?	Test Notes	User Story Covered
Human(string name, int score, Hand* hand)	☐	- Same tests as Constructor for Player	N/A

playCard() : Card*	☐	- Same tests as playCard for CI	See Player:: playCard()

Card

Method	Tested?	Test Notes	User Story Covered
Card(rank:int, suit: string)	☐	- Called **once** for every Card in the game. - Test for negative rank - Test for positive suit - Test for empty suit(string) - Test for a correct suit.	Starting the game, Playing an Eight, Shuffle cards, etc.
matches(Card* other): bool	☐	- Called **for every time** a card is played to check if it is a valid card. - Test with the *other* Card having the same rank but a different suit. - Test with the *other* Card having the same rank and same suit. - Test with the *other* Card having a different rank and suit. - Test with the *other* Card being an 8 card. - Test with *this/current card* being an 8 card.	#12 - Playing an Eight, #7 - Play card on turn
getRank(): int	☐	- Called **for every time** a card is played. - Test that rank the Card object was initialized with is the same value returned by this function. - Called when Cards are compared in the *matches(Card*): bool* method. - Called when a Player's hand is displayed during their turn and we need to check for the card's ranks. - Called when the current top card is played.	#12 - Playing an Eight, #7 - Play card on turn.
getSuit(): string	☐	- Test that rank the Card object was initialized with is the same value returned by this function. - Called when Cards are compared in the *matches(Card*): bool* method. - Called when a Player's hand is displayed during their turn. - Called when the current top card is played.	#12 - Playing an Eight, #7 - Play card on turn.
isEight() : bool	☐	**Tested indirectly:** - Tested as part of playing a card by player. If true, any card can be played by the player. If false, only the correct suit or rank can be played.	#12 - Playing an Eight, #7 - Play card on

			turn.
setSuit(string) : void	☐	- Called **once** when an 8 card is played by a player **How to test:** - Play an 8 card. - Call matches() to make sure the move is valid. - Call SetSuit() to change the suit of the next card to be played. - Test if the suit has been changed. - Test if a card from the new suit can be played. - Test and make sure that the old suit cannot be played.	

Deck

Method	Tested?	Test Notes	User Story Covered
Deck(vector<Card*>)	☐	- **Check for proper destruction.** Test that the Cards in the vector are destroyed when the Deck object is destroyed. - Called **once** when the discard and playing Decks are created at the start of the game.	#10 - Starting game
shuffleCards(): void	☐	- Called **once** when the playing Deck is created at the start of the game. - Called **once** on the discard Deck when the draw pile/deck is empty. In this situation, the top Card is first removed and then the discard deck is shuffled and put into the playing Deck. The discard Deck should contain only the top Card after this. - Test that all Cards are still in the Deck after being shuffled. - Test that the order of Cards is different.	#4 - Shuffle cards
getTopCard(): Card*	☐	- Tested for both discard and stock piles. - **Generally:** Test that the returned Card is the last Card in the Deck object's vector after this function is called. **Discard pile / Discard cards :** - Called **once** when we show the player the top card in the discard pile. This is when the game information is being displayed to the Player(s) on the screen during their turns. **Stock pile / Draw cards:**	#12 - Playing an Eight, #7 - Play card on turn.

		- When a player draws, whether they play that card or not will change how many times this is called. - Is called **once** to test the scenario when the first card in stock **is** the correct card. - Is called **at least twice** to test the scenario when the first card in stock **is not** the correct card. - For issue #3 - called once. - For issue #14 - called two times - For issue #15 - called four times **How to test:** - Create a deck with a few cards - Get the top card of the deck - Check if card returned is the top card.	
receiveCard(Card*): void	☐	- Test that the *received* Card is the last Card in the Deck object's vector after this function is called. - Called when a Player plays a Card to the playing Deck.	#7 - Play card on turn.
getSize() : int	☐	**Tested indirectly:** - Called as part of playing a card.	-

DeckFactory

Method	Tested?	Test Notes	User Story Covered
static Deck* makeDeck()	☐	**Tested Indirectly**: - It will be tested when the Deck class is tested. - Called once when the game starts to make the discard Deck from which Players are dealt Cards for their Hands.	#10 - Starting game

Hand

Method	Tested?	Test Notes	User Story Covered
Hand(vector<Card*>)	☐	- Test that Card pointers are destroyed after Hand is destroyed.	#10 - Starting game

		- Called **once** for each Player when they are receiving their hands.	
getCards(): vector<Card*>&	☐	- Called **once** when the cards in a Player's hand is needed. - Test for null pointer **How to test:** - Create a Hand with 5 cards. - Call getCards() and check if 5 cards are returned by the method. - Remove a card from the player's hand. - Call getCards() and check if 4 cards are left in the player's hand.	#7 - Play card on turn.
addToHand(Card*): void	☐	- Called **every time** a Player receives a Card. - Test that the received Card is not the null pointer / also test for null ptr being received. - Called at least **5 times** per player turn. If 2 players, call 7 times. If >2, call 5 times. Also called when drawing cards. - Called **0 times** if no cards are drawn - Called **twice** if 2 cards are drawn (issue 14) - Called **4 times** if 4 cards are drawn (issue 15) **How to test:** - Add a single card/multiples, to hand, check if it gets added correctly. - Test null card pointer	#14 - Make player draw four cards #15 - Make player draw two cards, #10 - Starting game #3 - Draw a Card
removeFromHand(Card*): void	☐	- Called **every time** a Player plays a card. - Test to make sure Card is valid i.e. not a null pointer. - Called at least **1 times** per player turn. **How to test:** - Remove a single card/multiples, from hand, check if it gets removed correctly. - Check if the card is added to the discard pile correctly. - Check the size of the hand. It should have be reduced by the number of cards removed.	

Views (Mocked):

UI

Method	Tested?	Test Notes	User Story Covered

display() : virtual void	N/A	- N/A - View will be mocked so the method doesn't need to be tested.	N/A

TurnUI

Method	Tested ?	Test Notes	User Story Covered
TurnUI(PlayerController*): virtual void	N/A	- N/A - View will be mocked so the method doesn't need to be tested.	N/A
display() : virtual void	N/A	- N/A - Used to display content of view	N/A
getPlacedCard() : virtual string	N/A	- N/A - View will be mocked so the method doesn't need to be tested.	N/A

StartGameUI

Method	Tested?	Test Notes	User Story Covered
display() : virtual void	N/A	- N/A - Used to display content of view	N/A
getNumOfPlayers() : virtual int	N/A	- N/A - View will be mocked so the method doesn't need to be tested.	N/A

EndGameUI

Method	Tested ?	Test Notes	User Story Covered
display(): virtual void	N/A	- N/A - Used to display content of view	N/A
getQuit(): virtual bool	N/A	- N/A - View will be mocked so the method doesn't need to be tested.	N/A

HelpUI

Method	Tested?	Test Notes	User Story Covered
display(): virtual void	N/A	- N/A - Used to display content of view	N/A