**BlackJack Game**

**Test Plan Document**

**GROUP 2**

**Table of Contents**

1. **Test Items**

1.1 Classes to be tested

1.2 Features not to be tested

1. **Class Methods to be Tested**

1.1 Classes

1.1.2 Methods within the class that are tested

1. **Specific Test Cases**
2. **Instructions to run Java Suite**

4.1 Perfect compilation

4.2 JUnit TestFramework

4.3 Running the Tests

4.4 Expected output

4.5 Self-contained execution

1. **Test Items**

**1.1 Classes to be Tested:**

1.1.1 Client

1.1.2 Dealer

1.1.3 Deck

1.1.4 Game

1.1.5 GUI

1.1.6 Hand

1.1.7 Lobby

1.1.8 Message

1.1.9 Player

1.1.10 Server

1.1.11 Shoe

1.1.12 Table

**1.2 Classes to not be Tested**

1.2.1 Card

1.2.2 DealerPanel

1.2.3 LobbyPanel

1.2.4 LoginPanel

1.2.5 MessageType

1.2.6 PlayerPanel

1.2.7 Suit

1.2.8 Symbol

1.2.9 TablePanel

1. **Class Methods to be Tested:**

* Dealer
* Constructor()
* getHandValue()
* hitPlayer()
* startGame()
* DealStartingCards()
* clearDealerHand()
* Hand
* getCards()
* getHandTotal()
* isBusted()
* isBlackJack()
* addCard()
* clearHand()
* Lobby
* addTable()
* RemoveTable()
* getTables()
* Message
* getID()
* getBalance()
* getUsername()
* getPassword()
* getType()
* getText()
* getCard()
* Shoe
* getNumDecks()
* getCard()
* getNumCards()
* isEmpty()
* Table
* addPlayer()
* removePlayer()
* getDealer()
* setDealer()
* isEmpty()
* getGame()
* Player
* getTable()
* setTable()
* setBalance()
* setCurrBet()
* withdrawalMoney()
* depositMoney()
* setHand()
* addCard()
* getHandValue()
* stay()
* hasStayed()
* clearPlayerHand()
* getID()

1. **Specific Test Cases**

| **Test Case** | **Class** | **Method** | **Description** |
| --- | --- | --- | --- |
| TC01 | Dealer | Constructor() | Ensures constructor can be called outside of class with proper parameters |
| TC02 | Dealer | getHandValue() | Ensures handValue of Dealer is returned |
| TC03 | Dealer | hitPlayer() | Ensures player is given another card |
| TC04 | Dealer | startGame() | Ensures game is started and all previous hands are cleared |
| TC05 | Dealer | dealCard() | Dealer and player are given 2 cards at start of game |
| TC06 | Dealer | clearHand() | Ensures dealer hand is empty |
| TC07 | Hand | getCards() | Ensures one or more cards are retrieved by dealer then put into hand |
| TC08 | Hand | getHandTotal() | Returns the total value of the dealer/player hand |
| TC09 | Hand | isBusted() | Ensures boolean is set to true if HandTotal surpasses 21 |
| TC10 | Hand | isBlackJack() | Ensures boolean is set to true if HandTotal == 21 |
| TC11 | Hand | addCard() | Ensures a new card is added to the hand |
| TC12 | Hand | clearHand() | Ensures all cards are deleted from hand, leaving hand Total == null |
| TC13 | Lobby | addTable() | Ensures table can be added only within the lobby’s table limit of 3 |
| TC14 | Lobby | removeTable() | Ensures table can be removed from lobby |
| TC15 | Lobby | getTables() | Returns the list of tables and it’s total number |
| TC16 | Message | getID() | Returns client’s ID that may be modified |
| TC17 | Message | getBalance() | Returns clients' balance that may be modified |
| TC18 | Message | getUsername() | Returns client username that may be null or wrong or correct |
| TC19 | Message | getPassword() | Returns client password that may be null from not being set or wrong or correct |
| TC20 | Message | getType() | Tests method that is null from not being set or loop through all types to ensure they match |
| TC21 | Message | getText() | Ensure that text can be sent whether null or made or modified afterwards and sent again |
| TC22 | Message | getCard() | For each new card in a deck, a message can be generated with it |
| TC23 | Shoe | getNumDecks() | Return the number of decks contained in shoe, which must be 3 |
| TC24 | Shoe | getCard() | Draw a card from Shoe() |
| TC25 | Shoe | getNumCards() | Must return the total number of cards in the shoe not related to numDecks |
| TC26 | Shoe | isEmpty() | Checks if the Shoe is empty |
| TC27 | Table | addPlayer() | Ensure a player can be indeed at the table without exceeding max Players |
| TC28 | Table | removePlayer() | Ensure a player can be taken out of the table |
| TC29 | Table | getDealer() | Retrieve a dealer for the table |
| TC30 | Table | setDealer() | Set the dealer at the dealer for this table |
| TC31 | Table | isEmpty() | Checks if table is empty and ensures that you can add a player after the fact |
| TC32 | Table | getGame() | Ensures a game can be played in the table |
| TC33 | Player | getTableID() | Ensures that player can access the game table’s ID |
| TC34 | Player | setTableID() | Ensures that player can modify the game table’s ID |
| TC35 | Player | setBalance() | Ensures that player balance can be set to a positive number |
| TC36 | Player | SetBalance() | Ensures that player balance can’t be set to a negative number |
| TC37 | Player | setCurrBet() | Ensures that the bet made by player can be made player’s balance |
| TC38 | Player | setCurrBet() | Ensures that the bet made by the player cant exceed the player’s balance & returns false if it tries. |
| TC39 | Player | withdrawlMoney() | Ensures player can withdraw money from their balance. |
| TC40 | Player | withdrawlMoney() | Ensures that player cannot withdraw money that exceeds their balance. |
| TC40 | Player | depositMoney() | Ensure that player can deposit money into account. |
| TC41 | Player | setHand() | Ensure player’s hand equals to the two card in hand class constructor |
| TC42 | Player | addCard() | Ensures that when player calls hit is given an extra card |
| TC43 | Player | getHandValue() | Ensures that the value of the hand can be read and returned |
| TC44 | Player | stay(), hasStayed() | Ensure that player refuses to be given another card returns a boolean to show that. |
| TC45 | Player | ClearPlayerHand() | Ensures that the hand of the player is cleared for the next round of blackJack |
| TC46 | Player | getID() | Ensures all IDs of different players are unique |
|  |  |  |  |

1. **Instructions to run the test suite**

4.1. Perfect Compilation:

Ensure that the classes in the project have no build errors and are able to compile. Like Shoe.java, Dealer.java, Player.java, etc. Also make sure any of their corresponding test files also compile.

4.2. JUnit Test Framework

Ensure the Test files are organized properly. Ensure that the classes are accessible in the src/ directory. Each Test Class should follow naming conventions for the organization’s sake, like classNameTest. Annotate all test methods using “@Test.”

4.3. Running the Tests

Use a JUnit-compatible test runner through either a preferred IDE or command-line interface. Added a TestRunner to run the test suite for all the test classes

4.4. Expected Output

The tests have been developed independently of any runtime input, configuration, or user interaction. No need to set up any game state or load external resources. The test classes will create and manage all objects and data of the class it is testing internally.

4.5. Self-Contained Execution

The test suite does not rely on any third-party services or files. It’s self-contained and portable. Anyone running the project source code and test suite should receive the same test results regardless of environment, assuming that Java and JUnit are properly configured.

By following the instructions listed above, you can verify the validity of the BlackJackProject through automated unit testing.