Table of Contents

[**Test Suite No.1** *(Class: NotationReader; Method:void parseDiceNotation(String diceNotation); type: Unit/Black Box )* 3](#_Toc464379909)

[Test Execution 3](#_Toc464379910)

[Source Code 3](#_Toc464379911)

[Output 5](#_Toc464379912)

[Test Result 6](#_Toc464379913)

[Test Summary 6](#_Toc464379914)

[**Test Suite No.2** *(Class: NotationReader; Method:void parseDiceNotation(String diceNotation); type: UnitWhite Box )* 7](#_Toc464379915)

[Test Execution 8](#_Toc464379916)

[Source Code 8](#_Toc464379917)

[Output 9](#_Toc464379918)

[Test Summary 9](#_Toc464379919)

[**Test Suite No.3** 10](#_Toc464379920)

[Test Execution 10](#_Toc464379921)

[Source Code 10](#_Toc464379922)

[Output 12](#_Toc464379923)

[Test Result 12](#_Toc464379924)

[Test Summary 12](#_Toc464379925)

[**Test Suite No.4** 13](#_Toc464379926)

[Test Execution 13](#_Toc464379927)

[Source Code 13](#_Toc464379928)

* Invalid inputs/parameters cannot materialize in the methods of objects that are below **NotationReader class** i.e. that are dependent on **NotationReader** class. This is because, all data generated is dependent on dice notation. And dice notation is **validated by NotationReader** before methods of other objects are invoked.
* Therefore, all objects dependent upon the **NotationReader** object, when executes its behavior will receive the correct input/parameter be it object passed by reference or primitive type passed by value.
* Taking account of the above points, all test data that follows besides that of methods in NotationReader will have **valid test data type only.**
* In order to **test private methods, from test package**, without breaking the encapsulation of original class. I have made duplicate class of the original class, in test package (Example – **Dice** class was duplicated in test package as “**publicForTest\_Dice**”) & changed the private modifier to protected in the duplicate class only.

## Test Suite No.1

**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void parseDiceNotation(String diceNotation) | 4d4-2 | Valid (2 delimiters –  ‘d’ & ‘-’) | Sets values of instance variables as following:   * diceNotation = 4d4-2 * numDices = 4 * numFaces = 4 * toRemove = 2 |
| 2 | void parseDiceNotation(String diceNotation) | 4d4 | Valid (1 delimiter – ‘d’) | Sets values instance variables as following:   * diceNotation = 4d4 * numDices = 4 * numFaces = 4 * toRemove = 0 |
| 3 | void parseDiceNotation(String diceNotation) | xyz | Invalid | Throws NumberFormatException with message - "\n\nINVALID DICE NOTATION!" |
| 4 | void parseDiceNotation(String diceNotation) | “” | Null | Throws NumberFormatException with message - "\n\nINVALID DICE NOTATION!" |

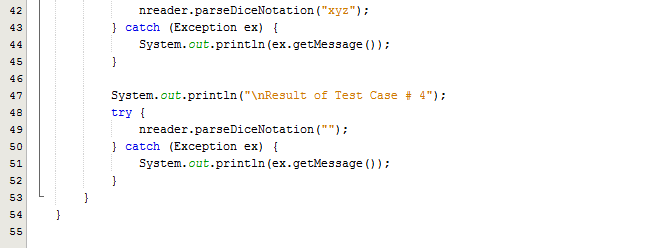
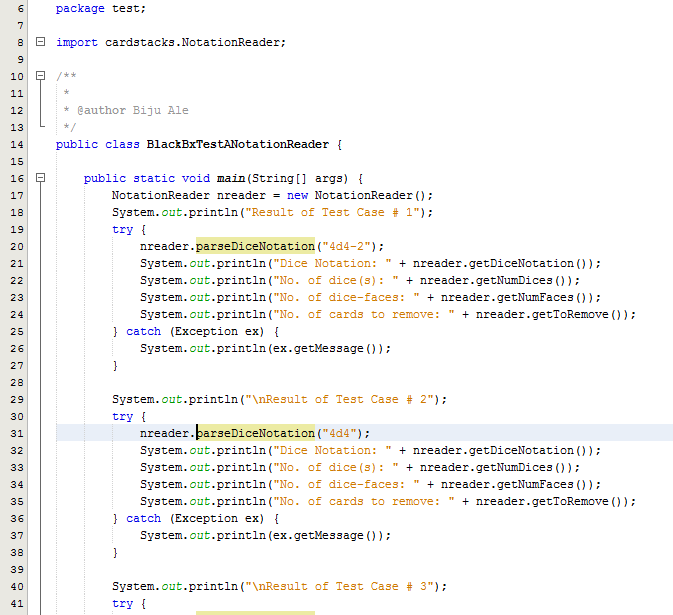
**Method of Equivalence partitioning:**

* Two types of valid (validated by Regex Pattern matcher) input is present – 1 sample was selected from each equivalence partition.
* Anything besides valid input’s Regex Pattern is another partition. 1 sample was selected.
* If no input is given, this is taken as another partition. Null is selected.
* No boundary value analysis required as per the nature of expected parameter.

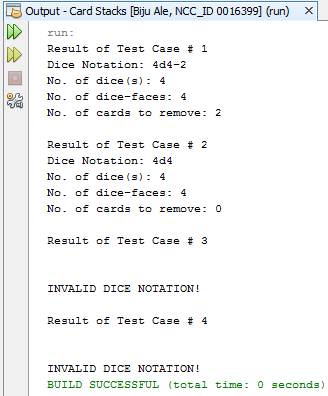
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**



#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | void parseDiceNotation(String diceNotation) | 4d4-2 | Valid (2 delimiters –  ‘d’ & ‘-’) | Sets values of instance variables as following:   * diceNotation = 4d4-2 * numDices = 4 * numFaces = 4 * toRemove = 2 | Yes |
| 2 | void parseDiceNotation(String diceNotation) | 4d4 | Valid (1 delimiter – ‘d’) | Sets values instance variables as following:   * diceNotation = 4d4 * numDices = 4 * numFaces = 4 * toRemove = 0 | Yes |
| 3 | void parseDiceNotation(String diceNotation) | xyz | Invalid | Throws NumberFormatException with message - "\n\nINVALID DICE NOTATION!" | Yes |
| 4 | void parseDiceNotation(String diceNotation) | “” | Null | Throws NumberFormatException with message - "\n\nINVALID DICE NOTATION!" | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

Test Suite No.1 also implicitly covered white box & black box tests for getter methods, which returned the respective values of instance variables. Hence, it too executed as expected without any errors.

## Test Suite **No**.2

**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: White Box / Unit Testing

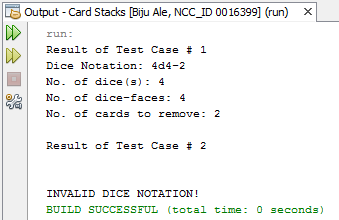
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void parseDiceNotation(String diceNotation) | 4d4-2 | Valid (2 delimiters –  ‘d’ & ‘-’) | Sets values of instance variables as following:   * diceNotation = 4d4-2 * numDices = 4 * numFaces = 4 * toRemove = 2 |
| 2 | void parseDiceNotation(String diceNotation) | xyz | Invalid | Throws NumberFormatException with message - "INVALID DICE NOTATION!" |

### Test Execution

#### Source Code



#### Output



### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.3

**Testing** **class**: cardstacks.Dice  
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Dice(NotitficationReader nreader) | nreader | Valid object | Constructor should set the instance variable value i.e. value of dice name.  It should also invoke complementary private methods: setMinMax method which invokes populateCombinations method which invokes roll method. |
| 2 | Dice (NotificationReader nreader) | nreader | \*Invalid object (due to invalid data member) | Constructor should not set the dice name. Exception should be thrown with message –  "No. of cards to remove cannot exceed total no. of cards. Enter valid notation.\n" |

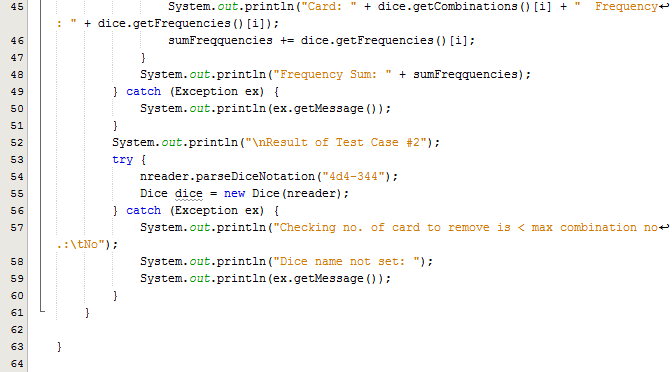
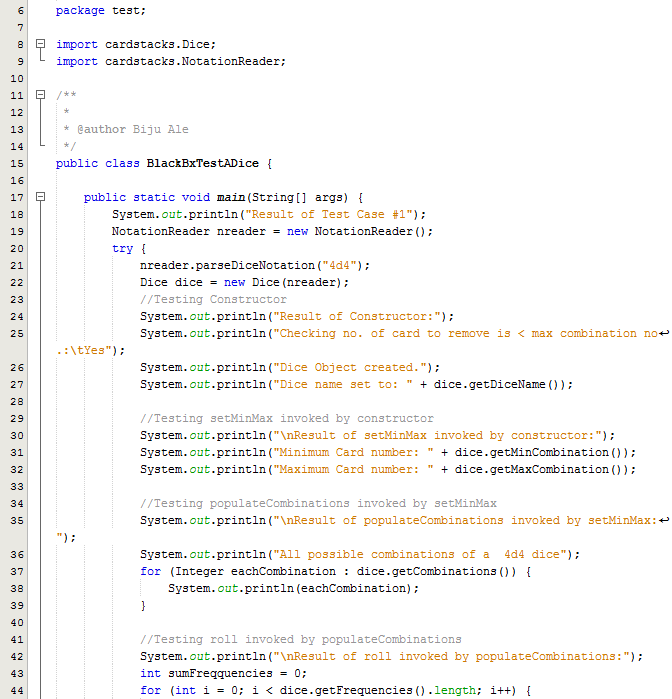
**Method of test data selection:**

* \*Here, invalid data member of ‘nreader’ means the parsing was correct (Test Suite No.1) but the number of card to remove exceeded the maximum combination number (total no. of cards). Correct parsing is checked in Test Suite No.1 whereas the valid no. of card to remove is checked in Test Suite No.3’s constructor.
* It never receives null as input, because before the constructor is called, null is already validated by NotationReader that was checked in test suite no.1 and 2.

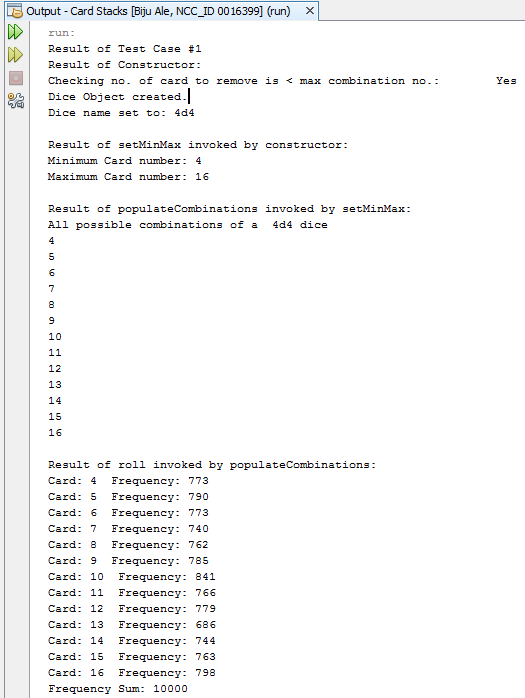
### Test Execution

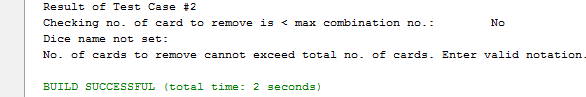
#### Source Code

**[PLEASE TURN OVER]**



#### Output





### Test Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** | |
| 1 | Dice(NotitficationReader nreader) | nreader | Valid object | Constructor should set the instance variable value i.e. value of dice name.  .  It should also invoke complementary private methods: setMinMax method which invokes populateCombinations method which invokes roll method. | Yes |
| 2 | Dice (NotificationReader nreader) | nreader | \*Valid object (with invalid data member) | Constructor should not set the dice name. Exception should be thrown with message –  "No. of cards to remove cannot exceed total no. of cards. Enter valid notation.\n" | Yes |

### Test Summary

Since, testing constructor invoked 3 other complementary private methods. It is safe to say that the 3 additional methods were implicitly black-box tested.

From the above test results, all tests were executed as expected.

## Test Suite No.4

**Testing** **class**: cardstacks.Dice  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Dice(NotitficationReader nreader) | nreader | Valid object | Constructor should set the instance variable value i.e. value of dice name.  It should also invoke complementary private methods: setMinMax method which invokes populateCombinations method which invokes roll method. |
| 2 | Dice (NotificationReader nreader) | nreader | \*Invalid object (due to invalid data member) | Constructor should not set the dice name. Exception should be thrown with message –  "No. of cards to remove cannot exceed total no. of cards. Enter valid notation.\n" |

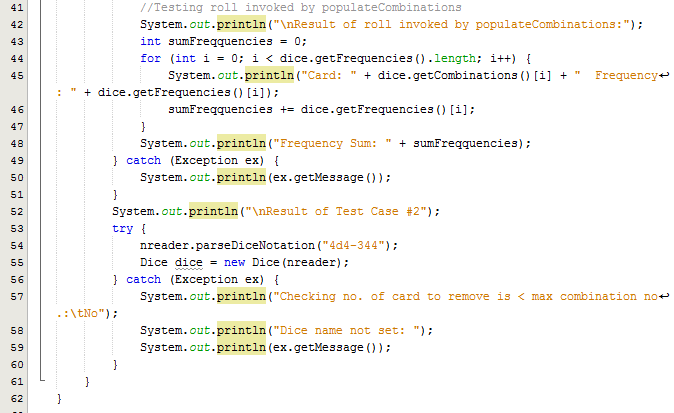
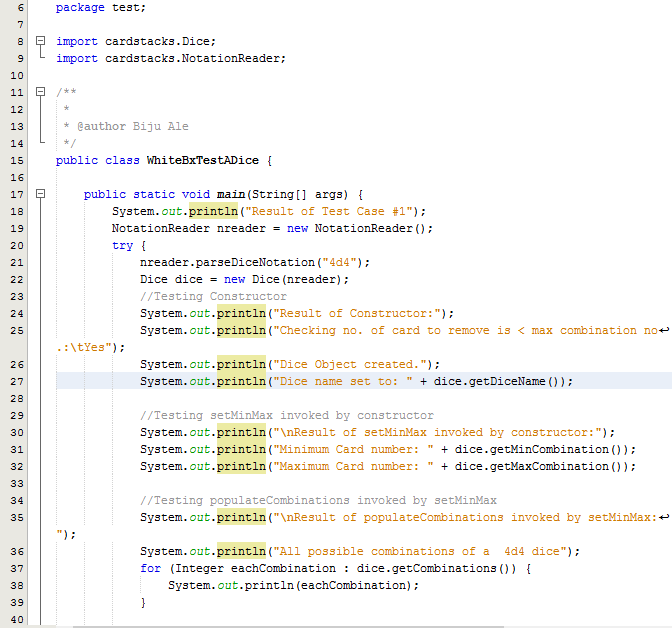
**Method of test data selection:**

* \*Here, invalid data member of ‘nreader’ the parsing was correct correct (Test Suite No.1) but the number of card to remove exceeded the maximum combination number (total no. of cards). Correct parsing is checked in Test Suite No.1 whereas the valid no. of card to remove is checked in Test Suite No.3’s constructor.
* It never receives null as input, because before the constructor is called, null is already validated by NotationReader that was checked in test suite no.1 and 2.

### Test Execution

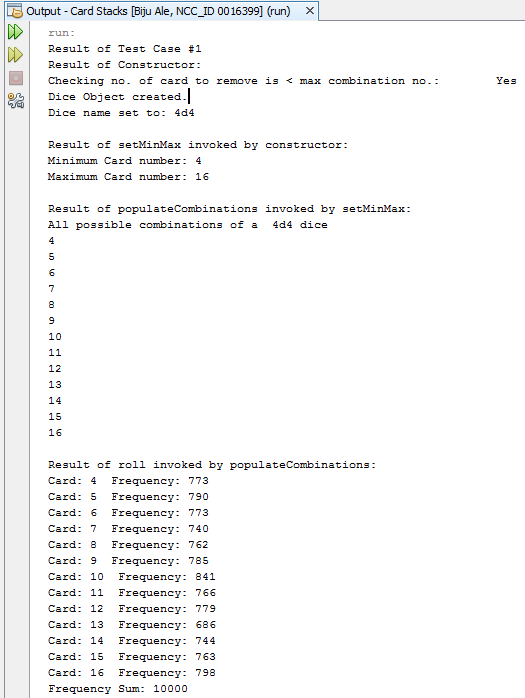
#### Source Code

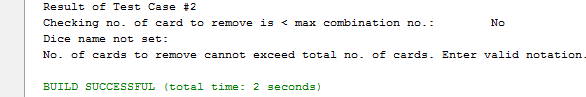
**[PLEASE TURN OVER]**



#### Output:

**[PLEASE TURN OVER]**





### Test Summary

Since, testing constructor invoked 3 other complementary private methods. It is safe to say that the 3 additional methods were implicitly white-box tested.

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.5

**Testing** **class**: cardstacks.Dice  
**Testing** **type**: Black Box / Unit Testing

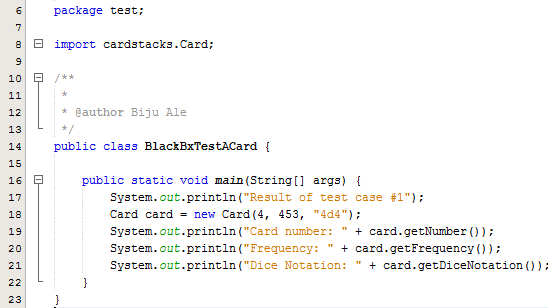
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Card(int number, int frequency, String diceNotation) | (4,453,”4d4”) | Valid | Constructor should set the values of instance variables as following:   * number = 4 * frequency = 453 * diceNotation = 4d4 |

**Method of test data selection:**

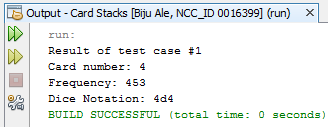
This constructor always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### Test Execution

#### Source Code



#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | Card(int number, int frequency, String diceNotation) | (4,453,”4d4”) | Valid | Constructor should set the values of instance variables as following:   * number = 4 * frequency = 453 * diceNotation = 4d4 | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite No.6

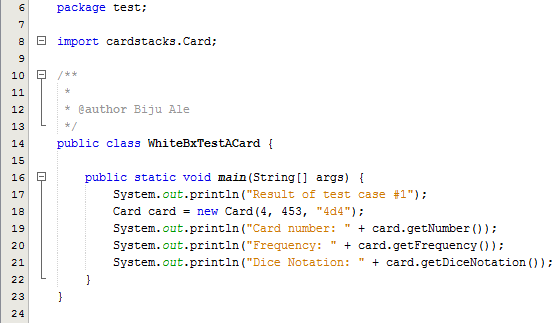
**Testing** **class**: cardstacks.Dice  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Card(int number, int frequency, String diceNotation) | (4,453,”4d4”) | Valid | Constructor should set the values of instance variables as following:   * number = 4 * frequency = 453 * diceNotation = 4d4 |

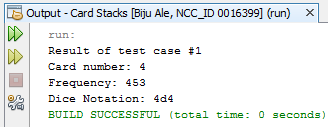
**Method of test data selection:**

This constructor always receives valid input only, because it is only invoked after NotationReader calss that was checked in Test Suite No.1 & 2, validates the input from GUI.

### Test Execution



#### Output



### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.7

**Testing** **class**: cardstacks.CardStack  
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | \*(dice, nreader, csrc) | Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack method which will add all shuffled cards to CardStack. |
| 2 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | (dice,nreader, csrc) | \*\*Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack method which will add all shuffled cards to CardStack.  It should then remove the correct no. of cards. |

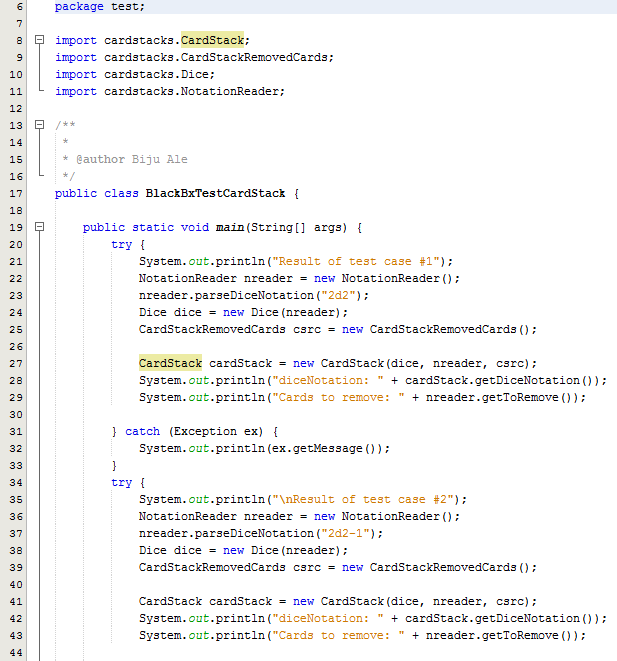
**Method of test data selection:**

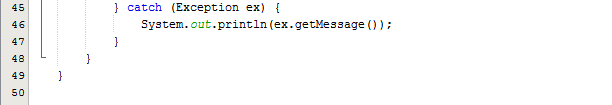
* \*Object of CardStackRemovedCards csrc is instantiated in GUI using default constructor. It does not contain any data member/instance variables. No. of card to remove here, is 0.
* \*\*’nreader’ contains data member, where card to remove is greater than 0.
* It never receives null as input, because before the constructor is called, null is already validated by NotationReader that was checked in test suite no.1 and 2.

### Test Execution

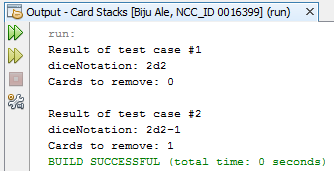
#### Source Code

**[PLEASE TURN OVER]**



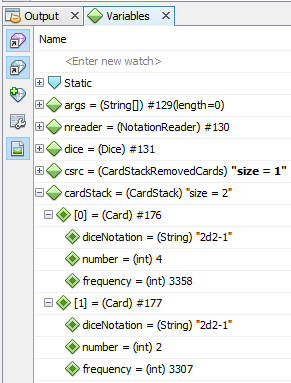
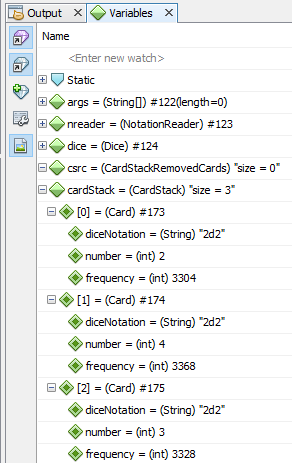


#### Output



After constructor invoked populateCardStack & removeCard methods, it added shuffled cards to CardStack.

Since, there is no getter for CardStack’s elements that test package can access, debug mode was used to test if populateCardStack was invoked by constructor with correct results. Following are the results:

  
***Figure: Checking if populateCardStack invoked by constructor removed correct no. of cards. For test case #1 (left) & test case#2.***

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | \*(dice, nreader, csrc) | Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack method which will add all shuffled cards to CardStack. | Yes |
| 2 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | (dice,nreader, csrc) | \*\*Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack method which will add all shuffled cards to CardStack.  It should then remove the correct no. of cards. | Yes |

### Test Summary

Since, testing constructor invoked 2 other complementary private methods. It is safe to say that the 2 additional methods were implicitly black-box tested.

From the above test results, all tests were executed as expected.

## Test Suite No.8

**Testing** **class**: cardstacks.CardStack  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | \*(dice, nreader, csrc) | Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack & removeCard private methods which will add all shuffled cards to CardStack. |
| 2 | CardStack(Dice dice, NotationReader nreader, CardStackRemovedCards csrc) | (dice,nreader, csrc) | \*\*Valid objects | Constructor should set the instance variable value i.e. value of dice notation.  It should also invoke populateCardStack & removeCard private methods which will add all shuffled cards to CardStack.  It should then remove the correct no. of cards. |

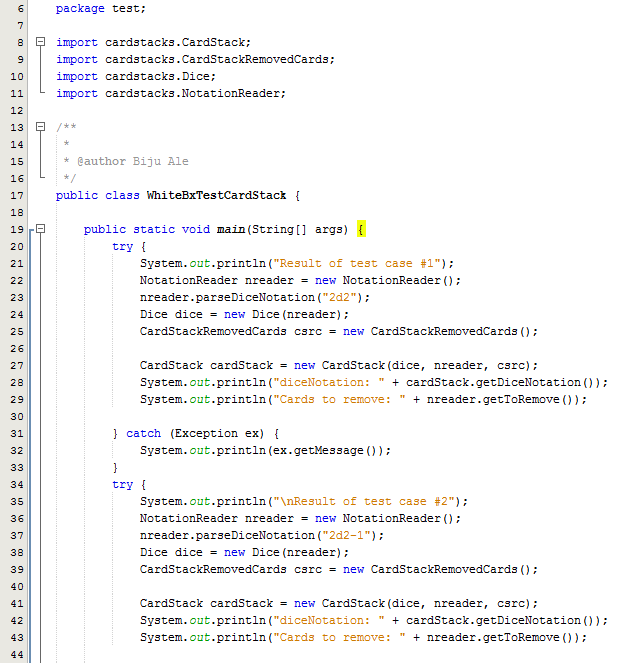
**Method of test data selection:**

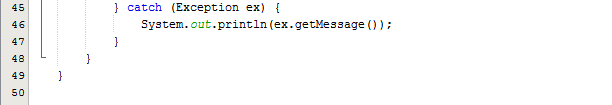
* \*Object of CardStackRemovedCards csrc is instantiated in GUI using default constructor. It does not contain any data member/instance variables. No. of card to remove here, is 0.
* \*\*’nreader’ contains data member, where card to remove is greater than 0.
* It never receives null as input, because before the constructor is called, null is already validated by NotationReader that was checked in test suite no.1 and 2.

### Test Execution

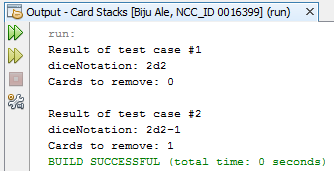
#### Source Code

**[PLEASE TURN OVER]**



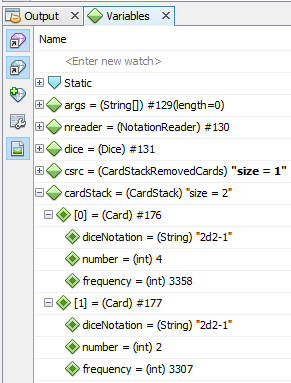
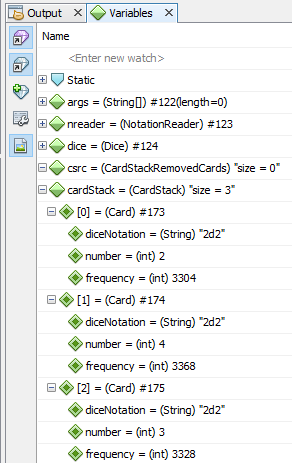


#### Output



After constructor invoked populateCardStack & removeCard methods, it added shuffled cards to CardStack.

Since, there is no getter for CardStack’s elements that test package can access, debug mode was used to test if populateCardStack was invoked by constructor with correct results. Following are the results:

  
***Figure: Checking if populateCardStack invoked by constructor removed correct no. of cards. For test case #1 (left) & test case#2.***

### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.9

**Testing** **class**: cardstacks.CardStackRemovedCards  
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | ArrayList<Card> getRemovedCards(String diceNotation) | “3d2-4” | Valid string | Returns correct no. i.e. of removed cards. i.e. 4 cards. |

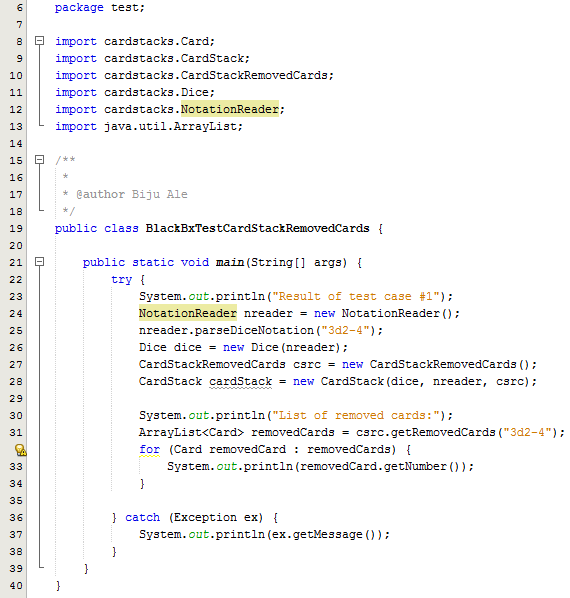
**Method of test data selection:**

This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

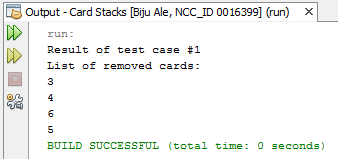
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**

****

#### Output

****

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | ArrayList<Card> getRemovedCards(String diceNotation) | “3d2-4” | Valid string | Returns correct no. i.e. of removed cards. i.e. 4 cards. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite No.10

**Testing** **class**: cardstacks.CardStackRemovedCards  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | ArrayList<Card> getRemovedCards(String diceNotation) | “3d2-4” | Valid string | Returns correct no. i.e. of removed cards. i.e. 4 removed cards. |

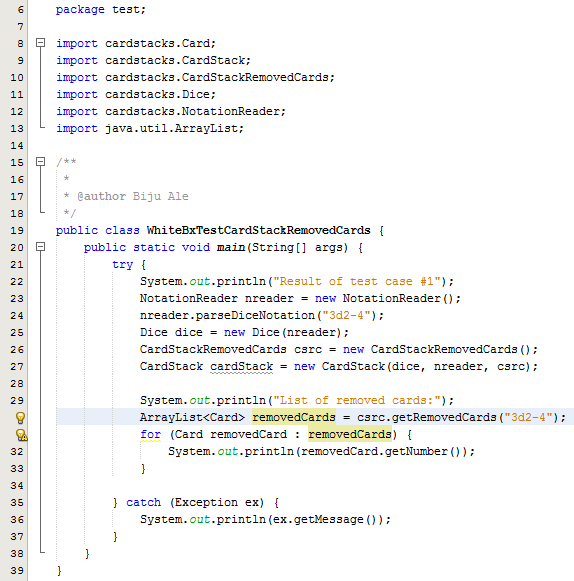
**Method of test data selection:**

This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

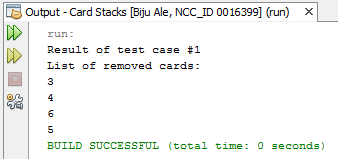
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**



#### Output

****

### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No. 11

**Testing** **class**: cardstacks.HistoryDice  
**Testing** **type**: Black Box / Unit Testing

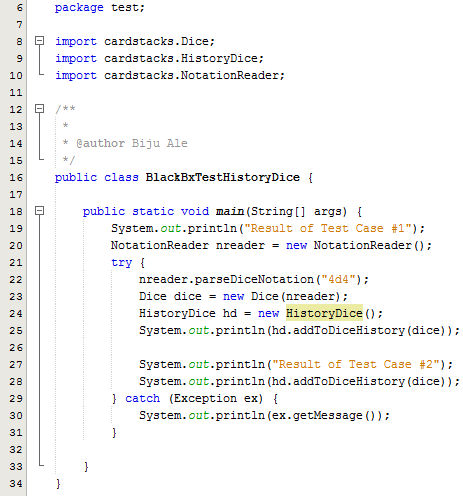
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean addToDiceHistory(Dice dice) | dice | Valid object | Returns TRUE after adding the dice to history. |
| 2 | boolean addToDiceHistory(Dice) | dice | Invalid object (duplicate dice) | Returns FALSE after matching dice to its history. |

**Method of test data selection:**

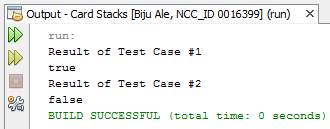
This method does not receive null, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the null input from GUI.

### Test Execution

#### Source Code

****

#### Output

****

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual Outcome as expected?** |
| 1 | addToDiceHistory(Dice dice) | dice | Valid object | Returns TRUE after adding the dice to history. | Yes |
| 2 | addToDiceHistory(Dice) | dice | Invalid object (duplicate dice) | Returns FALSE after matching dice to its history. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite No.12

**Testing** **class**: cardstacks.HistoryDice  
**Testing** **type**: White Box / Unit Testing

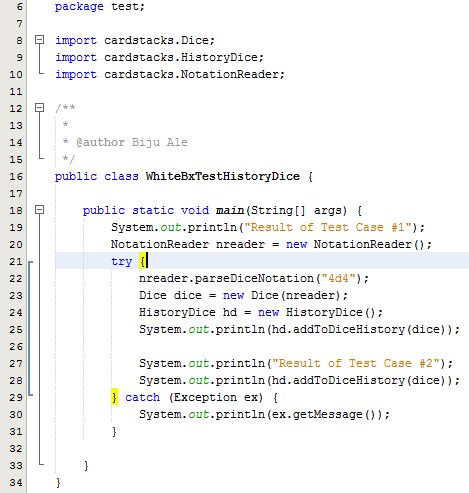
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean addToDiceHistory(Dice dice) | dice | Valid object | Returns TRUE after adding the dice to history. |
| 2 | boolean addToDiceHistory(Dice) | dice | Invalid object (duplicate dice) | Returns FALSE after matching dice to its history. |

**Method of test data selection:**

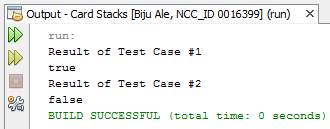
This method does not receive null input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates null input from GUI.

### Test Execution

#### Source Code



**Output**

****

### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No. 13

**Testing** **class**: cardstacks.CardStackDealtCards  
**Testing** **type**: Black Box / Unit Testing

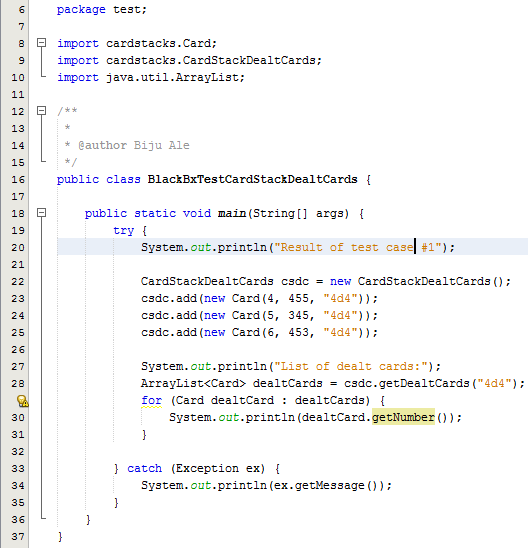
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | ArrayList<Card> getDealtCards(String diceNotation) | “4d4” | Valid string | Returns correct no. i.e. of dealt cards. i.e. 4 dealt cards. |

**Method of test data selection:**

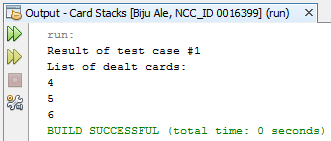
This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### Test Execution

#### Source Code



#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | ArrayList<Card> getDealtCards(String diceNotation) | “4d4” | Valid string | Returns correct no. i.e. of dealt cards. i.e. 4 dealt cards. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite No. 14

**Testing** **class**: cardstacks.CardStackDealtCards  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | ArrayList<Card> getDealtCards(String diceNotation) | “4d4” | Valid string | Returns correct no. i.e. of dealt cards. i.e. 4 dealt cards. |

**Method of test data selection:**

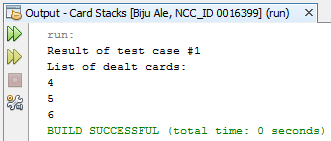
This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### Test Execution

#### Source Code



#### Output



### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.15

**Testing** **class**: cardstacks.CollectionCardStacks   
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Card moveDealtCard(String diceNotation, CardStackDealtCards csdc) | (“2d2”, csdc) | Valid objects | Returns dealt card by removing first card from the cardstack. |

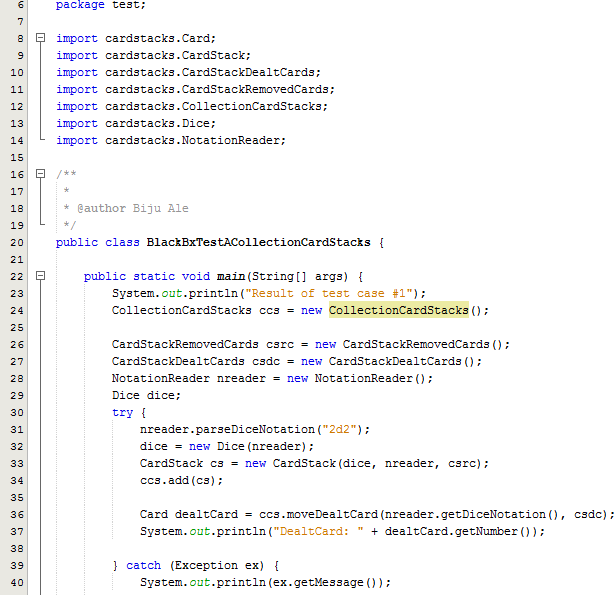
**Method of test data selection:**

This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

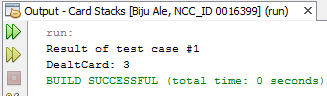
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**



#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | moveDealtCard(String diceNotation, CardStackDealtCards csdc) | (“2d2”, csdc) | Valid objects | Returns dealt card by removing first card from the cardstack. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite **No**.16

**Testing** **class**: cardstacks.CollectionCardStacks  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Card moveDealtCard(String diceNotation, CardStackDealtCards csdc) | (“2d2”, csdc) | Valid objects | Returns dealt card by removing first card from the cardstack. |
| 2 | Card moveDealtCard(String diceNotation, CardStackDealtCards csdc) | (“2d2”, csdc) | Valid objects | Repopulates stack when it is empty & returns dealt card by removing first card from the cardstack. |

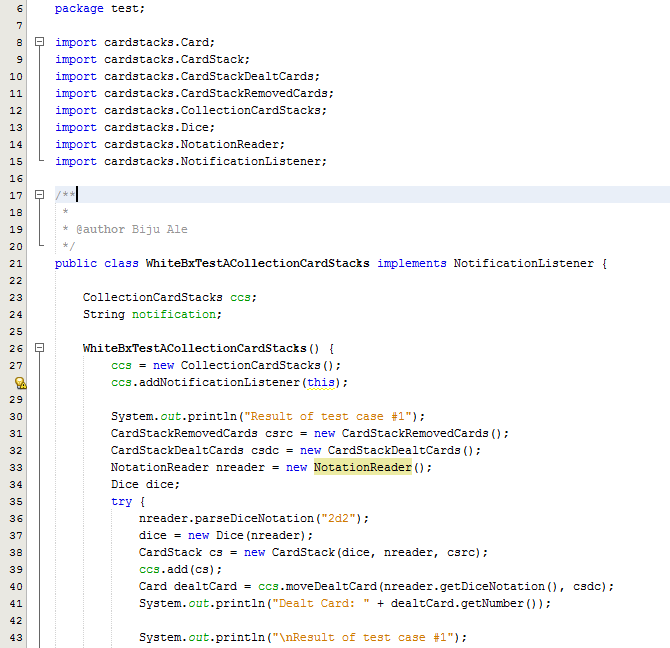
**Method of test data selection:**

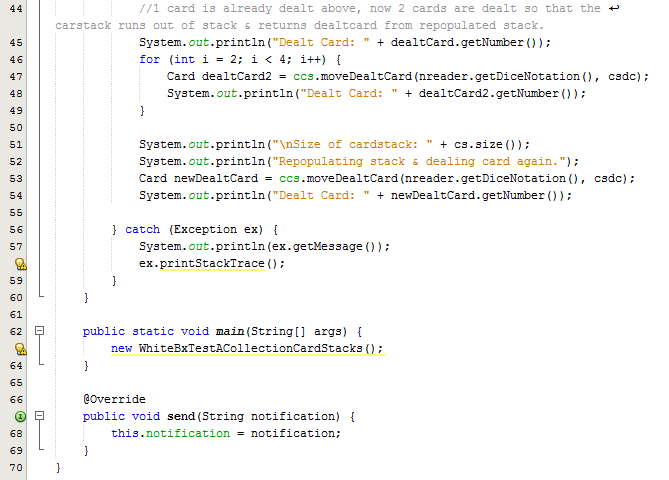
* This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.
* Both test data are same, but in the second case, the test data is passed 3 times so that the 2d2 cardstack runs out of cards.

### Test Execution

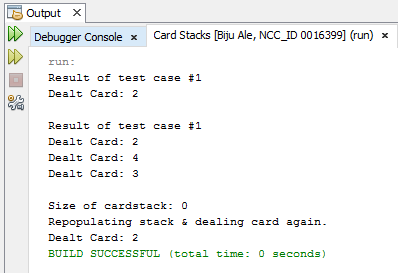
#### Source Code

**[PLEASE TURN OVER]**





#### Output



### Test Summary

While testing moveDealtCard method, implicitly rePopulateStack also got white-box & black-box tested.

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.17

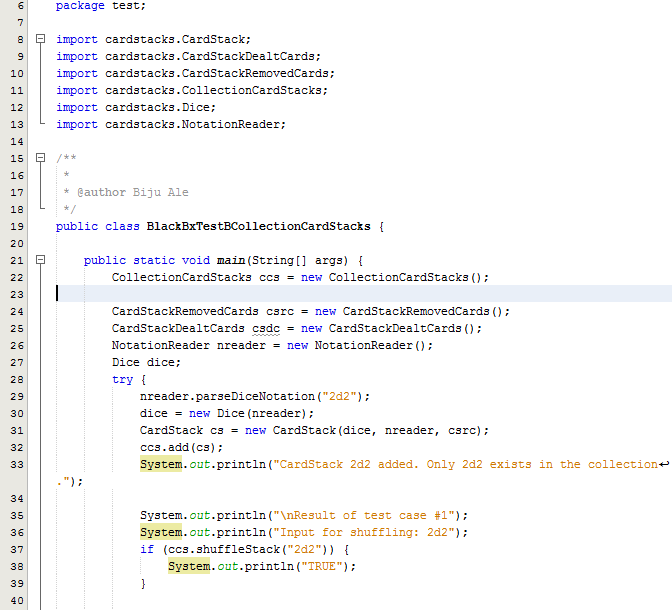
**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: Black Box / Unit Testing

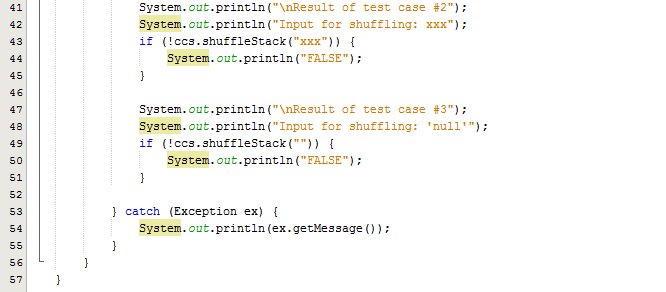
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean shuffleStack(String diceNotation) | “4d4” | Valid string | Returns true after changing order of 4d4 cardstack. |
| 2 | boolean shuffleStack(String diceNotation) | “xxx” | Valid string | Returns false. |
| 3 | boolean shuffleStack(String diceNotation) | “” | Null | Returns false. |

### Test Execution

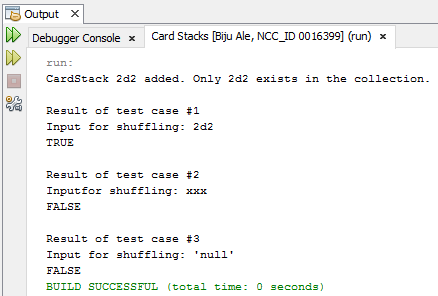
#### Source Code

**[PLEASE TURN OVER]**





#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | boolean shuffleStack(String diceNotation) | “4d4” | Valid string | Returns true after changing order of 4d4 cardstack. | Yes |
| 2 | boolean shuffleStack(String diceNotation) | “xxx” | Valid string | Returns false. | Yes |
| 3 | boolean shuffleStack(String diceNotation) | “” | Null | Returns false. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

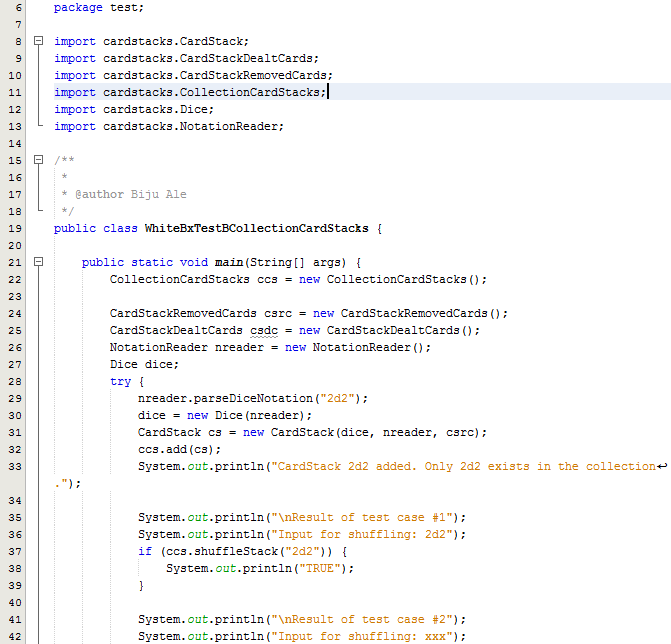
## Test Suite No.18

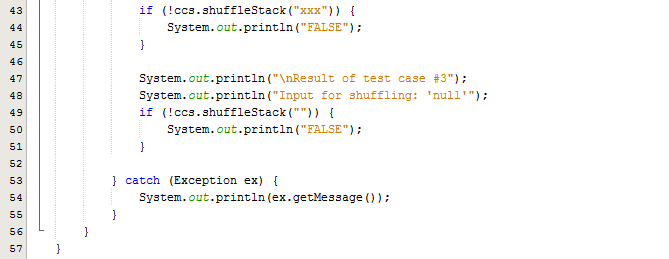
**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean shuffleStack(String diceNotation) | “4d4” | Valid string | Returns true after changing order of 4d4 cardstack. |
| 2 | boolean shuffleStack(String diceNotation) | “xxx” | Valid string | Returns false. |
| 3 | boolean shuffleStack(String diceNotation) | “” | Null | Returns false. |

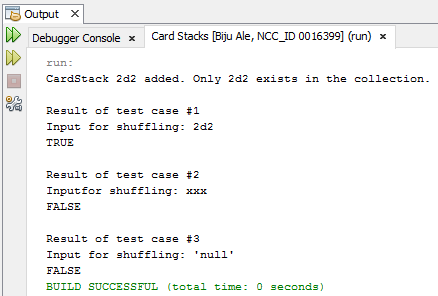
### Test Execution

#### Source Code





#### Output



### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

## Test Suite No.19

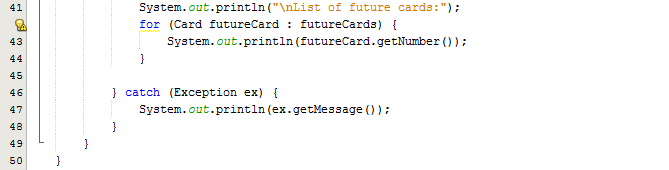
**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | CardStack getFutureCards(String diceNotation) | “2d2” | Valid string | Returns CardStack containing future cards. |
| 2 | CardStack getFutureCards(String diceNotation) | “xxx” | Invalid string | Returns null. |
| 3 | CardStack getFutureCards(String diceNotation) | null | Invalid string | Returns null. |

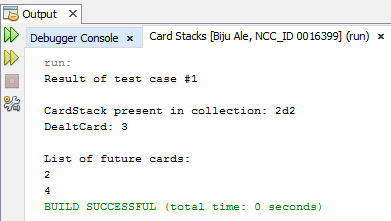
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**



#### Output



### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | CardStack getFutureCards(String diceNotation) | “2d2” | Valid string | Returns CardStack containing future cards. | Yes |
| 2 | CardStack getFutureCards(String diceNotation) | “xxx” | Invalid string | Returns null. | Yes |
| 3 | CardStack getFutureCards(String diceNotation) | null | Invalid string | Returns null. | Yes |

### Test Summary

From the above test results, all tests were executed as expected.

## Test Suite No.20

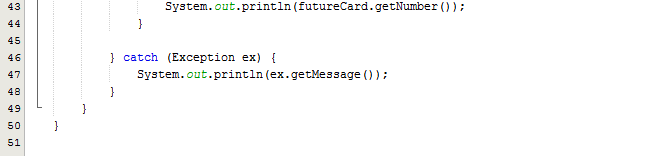
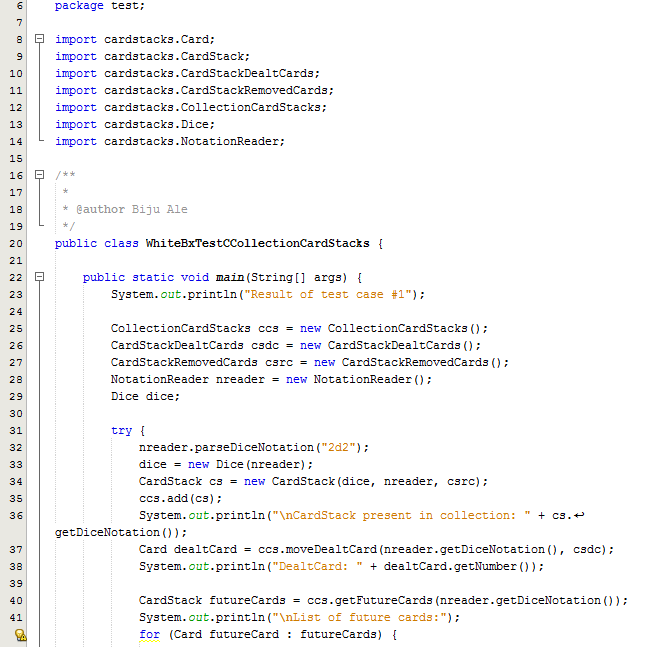
**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | CardStack getFutureCards(String diceNotation) | “2d2” | Valid string | Returns CardStack containing future cards. |
| 2 | CardStack getFutureCards(String diceNotation) | “xxx” | Invalid string | Returns null. |
| 3 | CardStack getFutureCards(String diceNotation) | null | Invalid string | Returns null. |

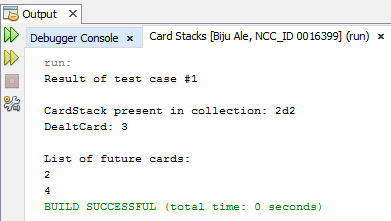
### Test Execution

#### Source Code

**[PLEASE TURN OVER]**



#### Output



### Test Summary

As from the above output, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution.

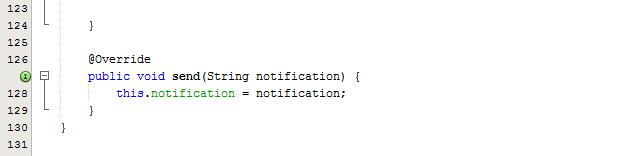
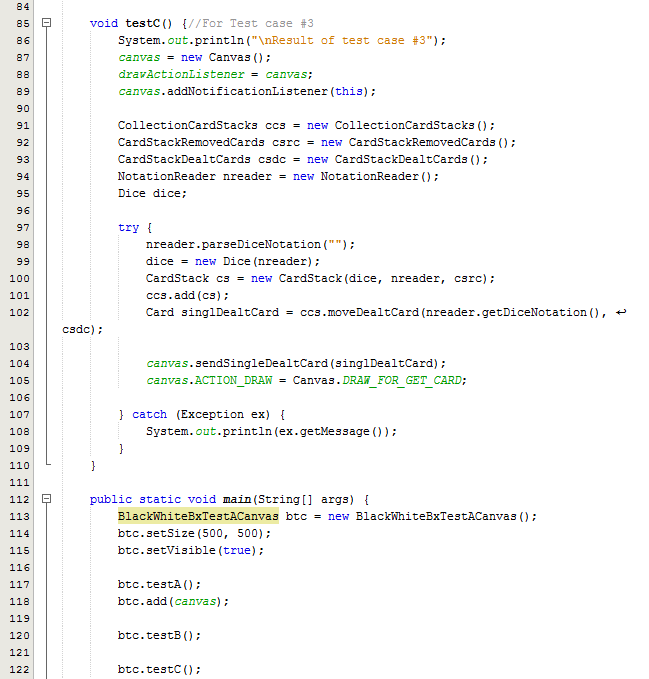
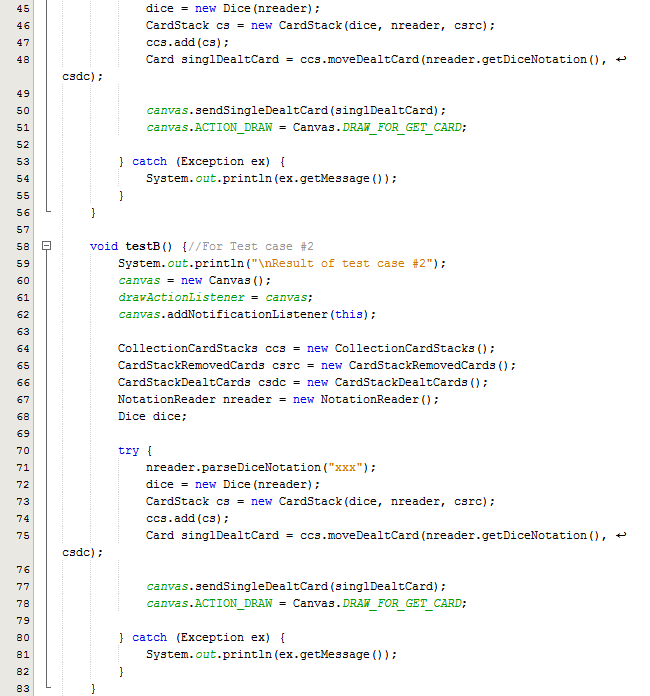
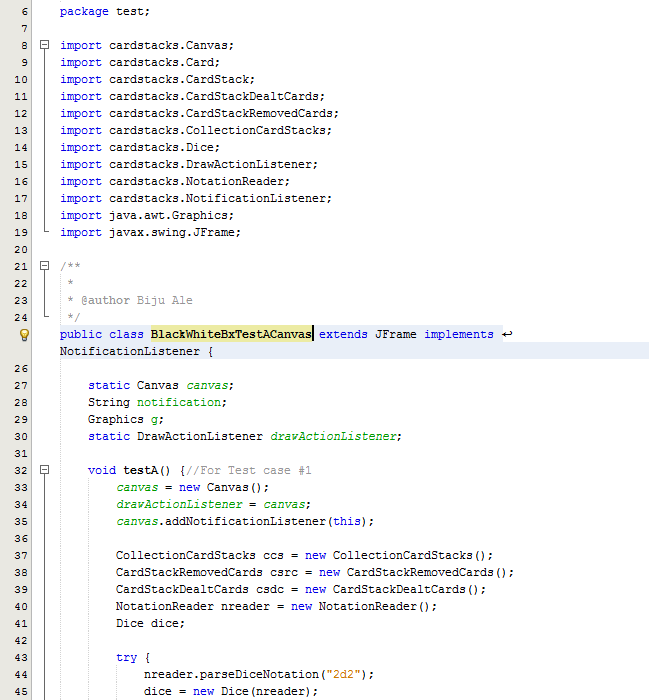
## Test Suite No.21

**Testing** **class**: cardstacks.Canvas  
**Testing** **type**: Black Box & White Box / Unit Testing

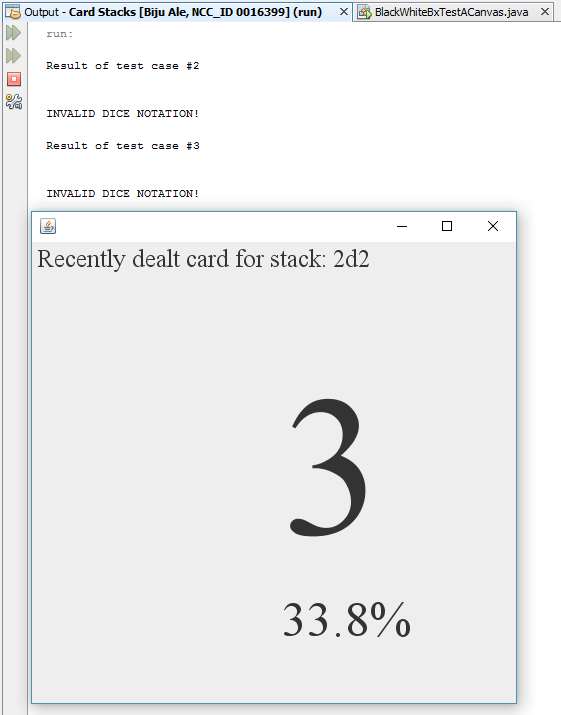
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Valid object | Draws single dealt card on JPanel |
| 2 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |
| 3 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |

### Test Execution

#### Source Code



#### Output

  
***Figure: The output window shows result for test case #2 and test case #3 whereas the JFrame shows the result of test case #1***

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Valid object | Draws single dealt card on JPanel | Yes |
| 2 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |
| 3 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |

### Test Summary

While black box test was performed, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution. Therefore, it is safe to say that both black box and white box was performed simultaneously.

From the above test results, all tests were executed as expected.

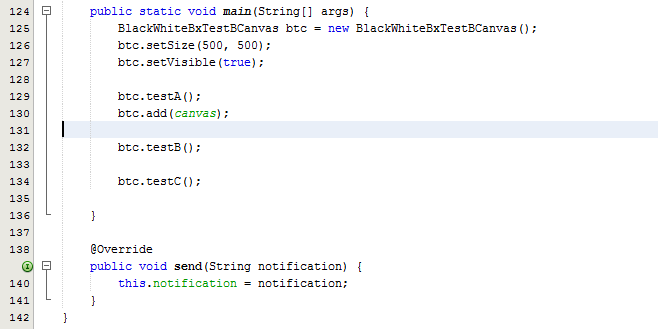
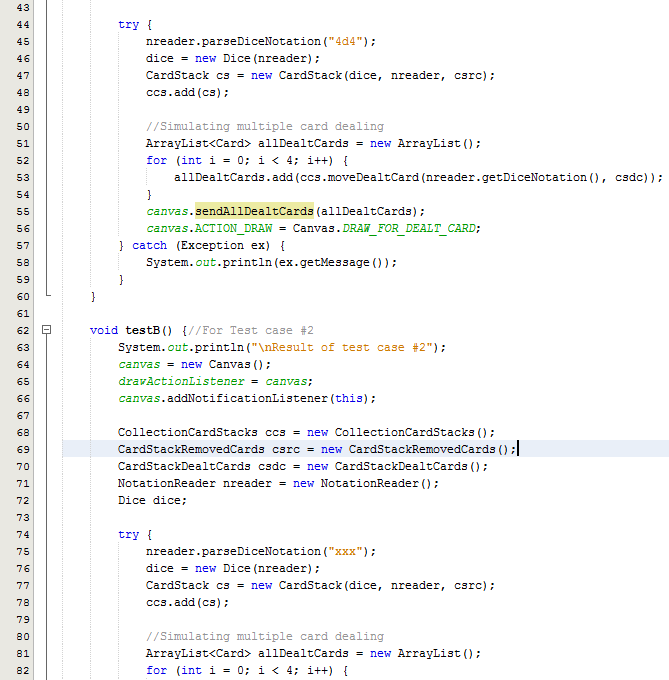
## Test Suite No.22

**Testing** **class**: cardstacks.Canvas  
**Testing** **type**: Black Box & White Box / Unit Testing

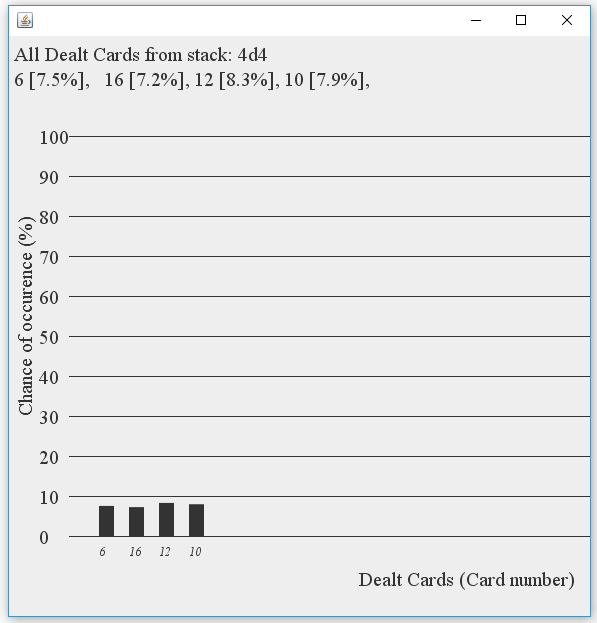
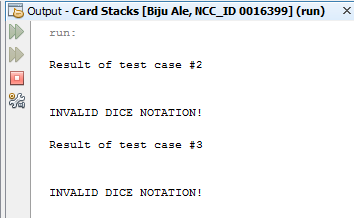
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Valid object | Draws Graph of all dealt cards on JPanel |
| 2 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |
| 3 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |

### Test Execution

#### Source Code



#### Output

  
***Figure: The output window shows result for test case #2 and test case #3 whereas the JFrame shows the result of test case #1***

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Valid object | Draws Graph of all dealt cards on JPanel | Yes |
| 2 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |
| 3 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |

### Test Summary

While black box test was performed, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution. Therefore, it is safe to say that both black box and white box was performed simultaneously.

From the above test results, all tests were executed as expected.

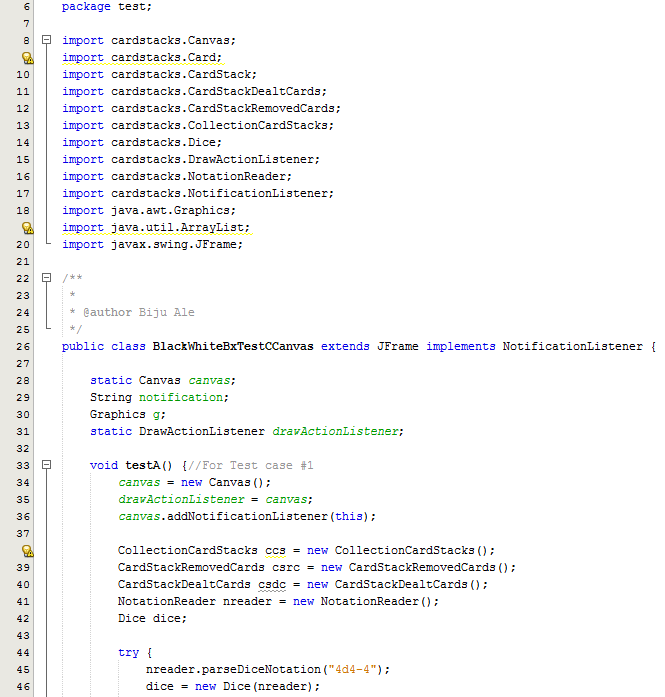
## Test Suite No.23

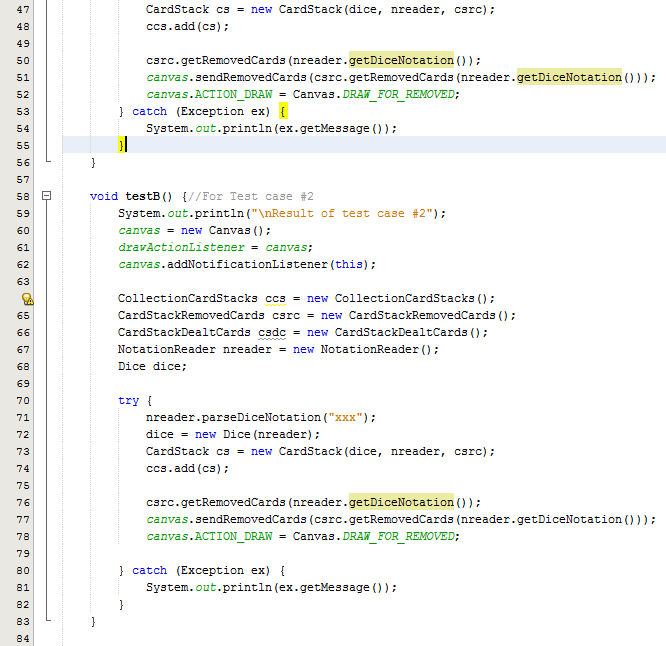
**Testing** **class**: cardstacks.Canvas  
**Testing** **type**: Black Box & White Box / Unit Testing

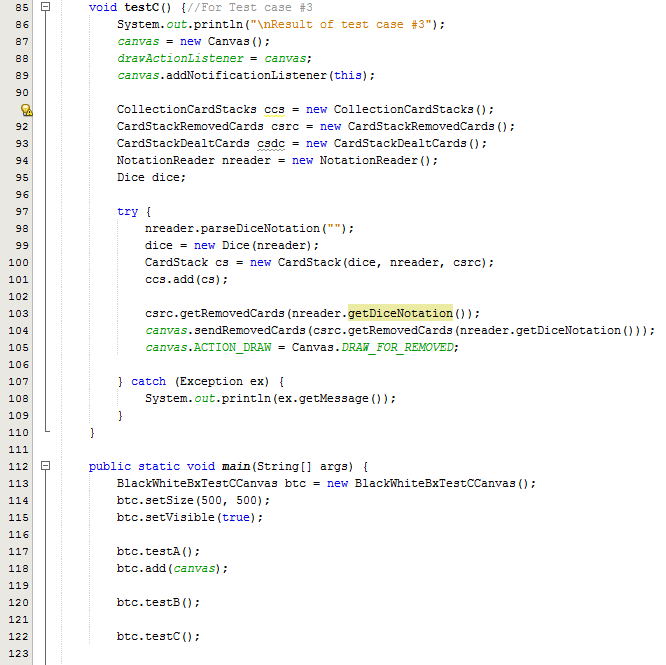
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Valid object | Draws all removed cards on JPanel |
| 2 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |
| 3 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |

### Test Execution

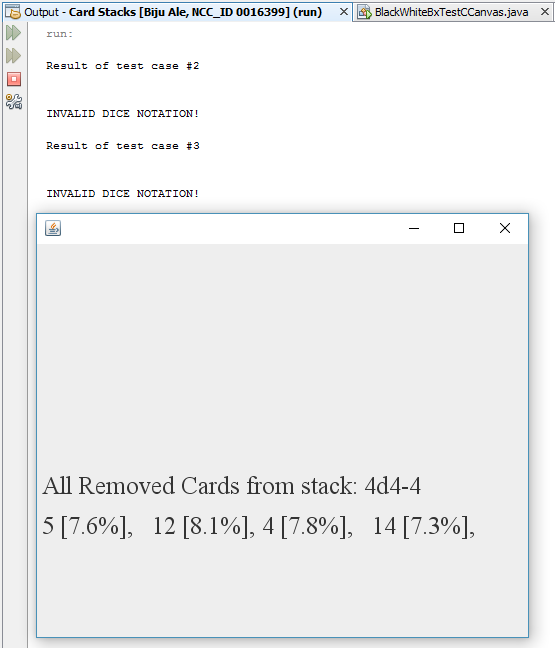
#### Source Code







#### Output

  
***Figure: The output window shows result for test case #2 and test case #3 whereas the JFrame shows the result of test case #1***

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Valid object | Draws all removed cards on JPanel | Yes |
| 2 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |
| 3 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |

### Test Summary

While black box test was performed, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution. Therefore, it is safe to say that both black box and white box was performed simultaneously.

From the above test results, all tests were executed as expected.

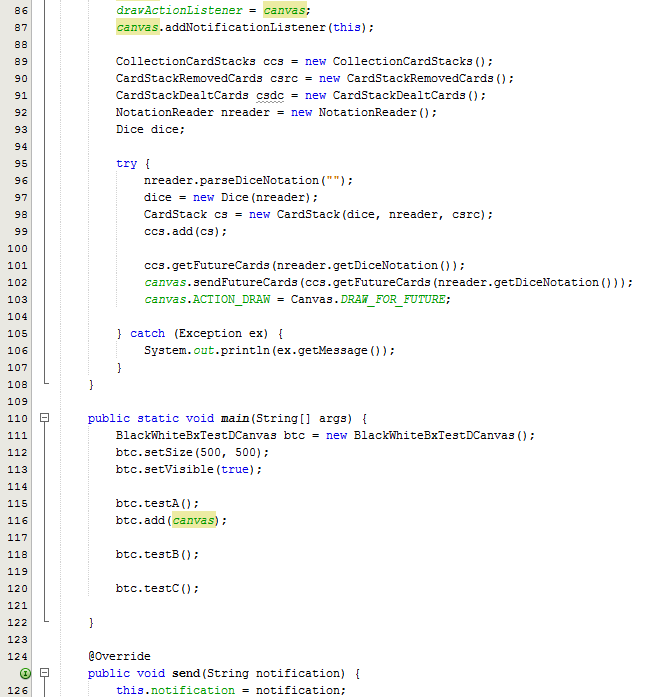
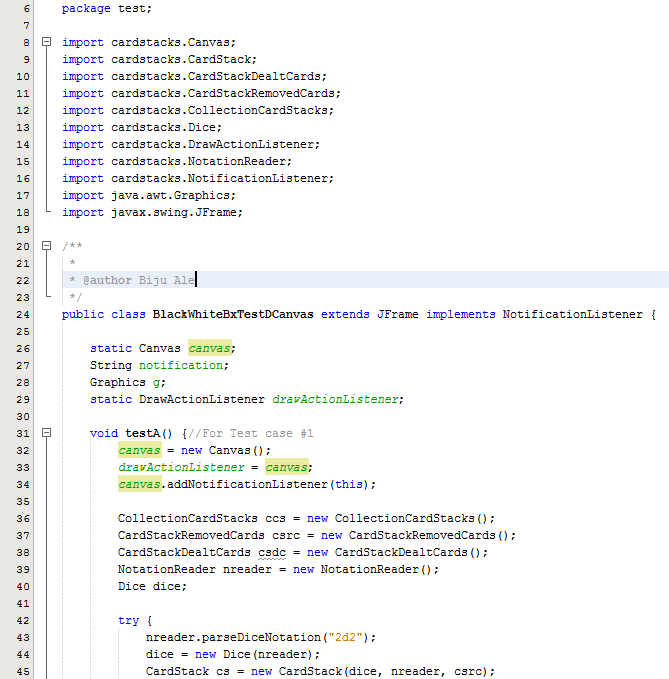
## Test Suite No.24

**Testing** **class**: cardstacks.Canvas  
**Testing** **type**: Black Box & White Box / Unit Testing

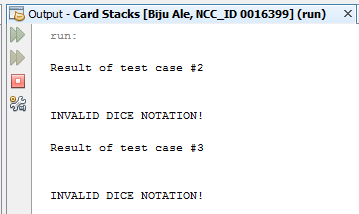
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendFutureCards(CardStack futureCards) | futureCards | Valid object | Draws graph of all future cards on JPanel |
| 2 | sendFutureCards(CardStack futureCards) | futureCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |
| 3 | sendFutureCards(CardStack futureCards) | futureCards | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” |

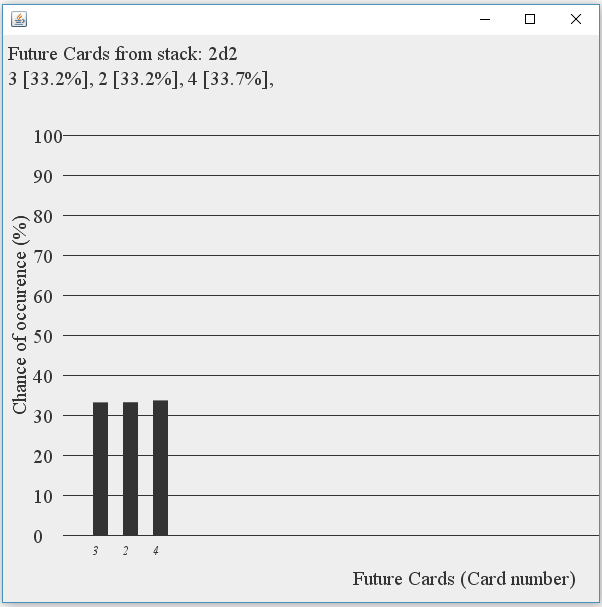
### Test Execution

#### Source Code



#### Output



  
***Figure: The output window shows result for test case #2 and test case #3 whereas the JFrame shows the result of test case #1***

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
| 1 |  |  | Valid object | Draws graph of all future cards on JPanel | Yes |
| 2 |  |  | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |
| 3 |  |  | Null (object has invalid data member i.e. invalid dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |

### Test Summary

While black box test was performed, all statements, conditions and branches were executed at least once, and no errors were discovered during the execution. Therefore, it is safe to say that both black box and white box was performed simultaneously.

From the above test results, all tests were executed as expected.

## Test Suite No.25

**Testing** **class**: cardstacks.NotationReader  
**Testing** **type**: White & Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void actionPerformed(ActionEvent e) | btnSubmit.doClick() | Valid | Saves the dice notation & deals single card. |
| 2 | void actionPerformed(ActionEvent e) | btnGetCard.doClick() | Valid | Deals single card. |
| 3 | void actionPerformed(ActionEvent e) | btnDealt.doClick() | Valid | Generates graph of all dealt cards. |
| 4 | void actionPerformed(ActionEvent e) | btnRemoved.doClick() | Valid | Draws string of all removed cards. |
| 5 | void actionPerformed(ActionEvent e) | btnFutureCards.doClick() | Valid | Genrates graph of all future cards. |
| 6 | void actionPerformed(ActionEvent e) | btnShuffle.doClick() | Valid | Changes the order of cards in the stack. |

### Test Execution

#### Source Code

#### Output

### Test Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** | **Actual outcome as expected?** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Test Summary

From the above test results, all tests were executed as expected.