Table of Contents

[Introduction 5](#_Toc464565553)

[Task -1 (Application) 6](#_Toc464565554)

[Overview of all Project Files 6](#_Toc464565555)

[Main package files 6](#_Toc464565556)

[Test package files 7](#_Toc464565557)

[Class: NotationReader 8](#_Toc464565558)

[Class: Dice 10](#_Toc464565559)

[Class: HistoryDice 13](#_Toc464565560)

[Class: Card 14](#_Toc464565561)

[Class: CardStack 16](#_Toc464565562)

[Class: CardStackRemovedCards 18](#_Toc464565563)

[Class: CollectionCardStacks 19](#_Toc464565564)

[Class: CardStackDealtCards 22](#_Toc464565565)

[Class: UserInterface 23](#_Toc464565566)

[Inner Class: MyButton 29](#_Toc464565567)

[Interface: NotificationListener 29](#_Toc464565568)

[Interface: DrawActionListener 30](#_Toc464565569)

[Class: Canvas 31](#_Toc464565570)

[Task-2 (Testing Data) 37](#_Toc464565571)

[Test Suite No.1 37](#_Toc464565572)

[Test Execution 37](#_Toc464565573)

[Test Result 41](#_Toc464565574)

[Test Summary 41](#_Toc464565575)

[Test Suite No.2 42](#_Toc464565576)

[Test Execution 43](#_Toc464565577)

[Test Summary 44](#_Toc464565578)

[Test Suite No.3 45](#_Toc464565579)

[Test Execution 45](#_Toc464565580)

[Test Result 49](#_Toc464565581)

[Test Summary 49](#_Toc464565582)

[Test Suite No.4 50](#_Toc464565583)

[Test Execution 50](#_Toc464565584)

[Test Summary 54](#_Toc464565585)

[Test Suite No.5 55](#_Toc464565586)

[Test Execution 55](#_Toc464565587)

[Test Result 57](#_Toc464565588)

[Test Summary 57](#_Toc464565589)

[Test Suite No.6 58](#_Toc464565590)

[Test Execution 58](#_Toc464565591)

[Test Summary 59](#_Toc464565592)

[Test Suite No.7 60](#_Toc464565593)

[Test Execution 60](#_Toc464565594)

[Test Result 64](#_Toc464565595)

[Test Summary 64](#_Toc464565596)

[Test Suite No.8 65](#_Toc464565597)

[Test Execution 65](#_Toc464565598)

[Test Summary 68](#_Toc464565599)

[Test Suite No.9 69](#_Toc464565600)

[Test Execution 69](#_Toc464565601)

[Test Result 71](#_Toc464565602)

[Test Summary 71](#_Toc464565603)

[Test Suite No.10 72](#_Toc464565604)

[Test Execution 72](#_Toc464565605)

[Test Summary 74](#_Toc464565606)

[Test Suite No. 11 75](#_Toc464565607)

[Test Execution 75](#_Toc464565608)

[Test Result 77](#_Toc464565609)

[Test Summary 77](#_Toc464565610)

[Test Suite No.12 78](#_Toc464565611)

[Test Execution 78](#_Toc464565612)

[Test Summary 79](#_Toc464565613)

[Test Suite No. 13 80](#_Toc464565614)

[Test Execution 80](#_Toc464565615)

[Test Result 81](#_Toc464565616)

[Test Summary 81](#_Toc464565617)

[Test Suite No. 14 82](#_Toc464565618)

[Test Execution 82](#_Toc464565619)

[Test Summary 83](#_Toc464565620)

[Test Suite No.15 84](#_Toc464565621)

[Test Execution 84](#_Toc464565622)

[Test Result 86](#_Toc464565623)

[Test Summary 86](#_Toc464565624)

[Test Suite No.16 87](#_Toc464565625)

[Test Execution 87](#_Toc464565626)

[Test Summary 90](#_Toc464565627)

[Test Suite No.17 91](#_Toc464565628)

[Test Execution 91](#_Toc464565629)

[Test Result 94](#_Toc464565630)

[Test Summary 94](#_Toc464565631)

[Test Suite No.18 95](#_Toc464565632)

[Test Execution 95](#_Toc464565633)

[Test Summary 97](#_Toc464565634)

[Test Suite No.19 98](#_Toc464565635)

[Test Execution 98](#_Toc464565636)

[Test Result 100](#_Toc464565637)

[Test Summary 100](#_Toc464565638)

[Test Suite No.20 101](#_Toc464565639)

[Test Execution 101](#_Toc464565640)

[Test Summary 103](#_Toc464565641)

[Test Suite No.21 104](#_Toc464565642)

[Test Execution 104](#_Toc464565643)

[Test Result 109](#_Toc464565644)

[Test Summary 109](#_Toc464565645)

[Test Suite No.22 110](#_Toc464565646)

[Test Execution 110](#_Toc464565647)

[Test Result 116](#_Toc464565648)

[Test Summary 116](#_Toc464565649)

[Test Suite No.23 117](#_Toc464565650)

[Test Execution 117](#_Toc464565651)

[Test Result 122](#_Toc464565652)

[Test Summary 122](#_Toc464565653)

[Test Suite No.24 123](#_Toc464565654)

[Test Execution 123](#_Toc464565655)

[Test Result 128](#_Toc464565656)

[Test Summary 128](#_Toc464565657)

[Final Test Suite No.25 129](#_Toc464565658)

[Test Execution 129](#_Toc464565659)

[Test Summary 135](#_Toc464565660)

[Task 3 – Class Diagram 136](#_Toc464565661)

[Conclusion 137](#_Toc464565662)

[Referencing 140](#_Toc464565663)

# Introduction

In games that use the element of randomness to the gameplay, often times, regular faced dices are used. However, for the dice to produce an ideal distribution, the time taken is relatively longer than the time period of game itself. One of the ways of mitigating this situation, is the use of card stack, where the card stack stores the card that corresponds to the combination generated by the dice roll, if in case the dice was rolled.

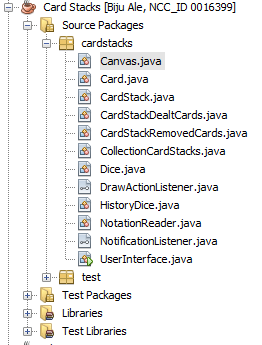
The java based application presented in this documentation, is specifically, and designed to solve the issue with regular dice rolls. In addition to storing the combination in the form of a card, it also promotes a fair randomness by optimizing the occurrence of each combination towards statistically favorable distribution.Moreover, it has many advantages over real life card stack adaptation. It can generate virtually any kind of dices which means it is able to simulate the rolling of multi-dimensional dices that is not possible in real world. It can independently track multiple type of dice’s rolling’s& maintain graphs of frequencies

This report incorporates mainly 3 parts – the source code of the application, the testing carried out full coverage of verification & validation of all classes, and finally the UML based class diagram to provide the overview of the application’s design.

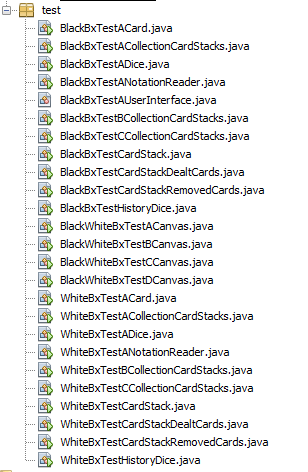
# Task -1 (Application)

## Overview of all Project Files

### Main package files

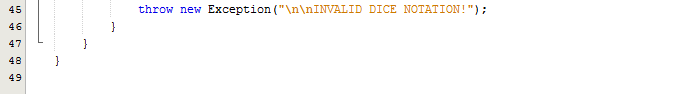


### Test package files



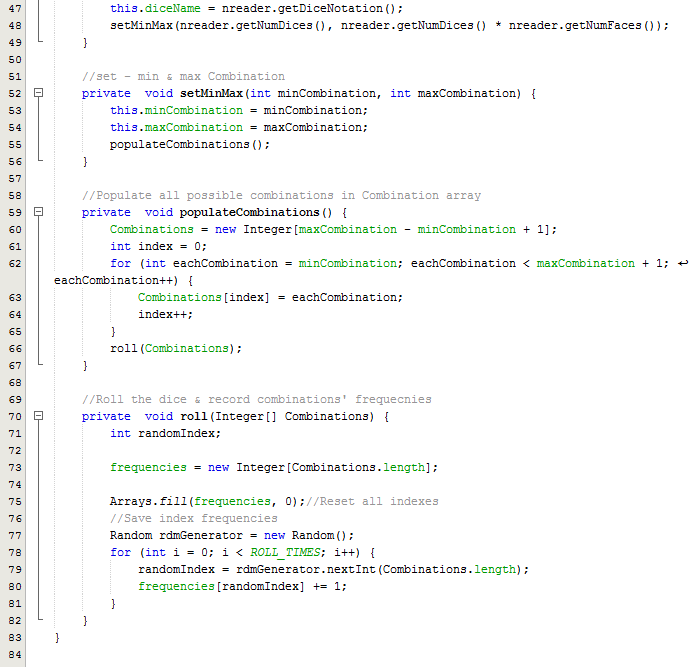
## Class: NotationReader

[PLEASE TURN OVER]

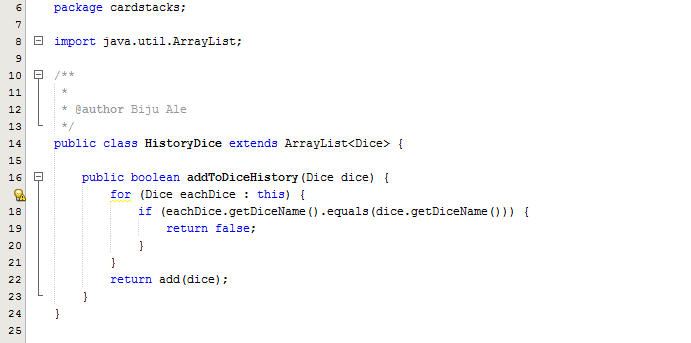


## Class: Dice

[PLEASE TURN OVER]

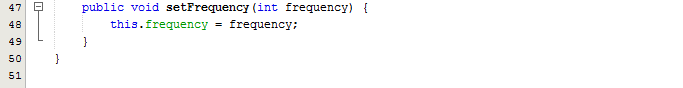
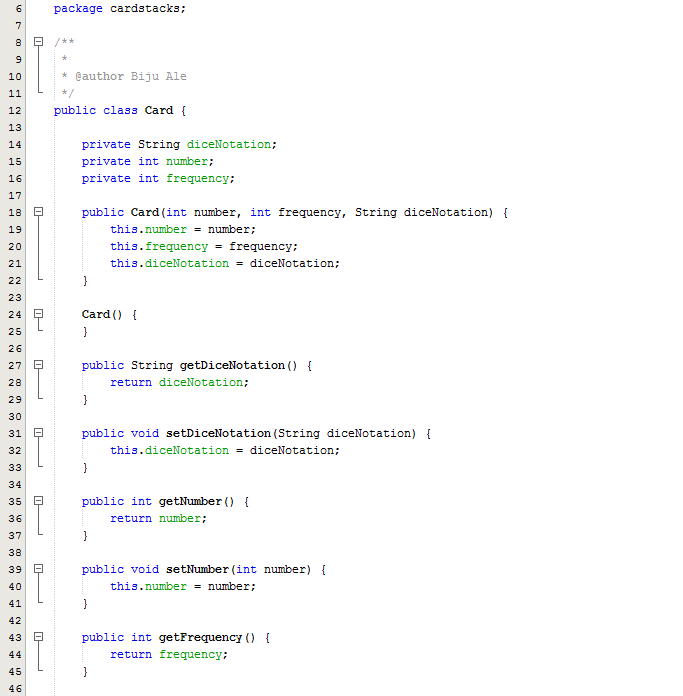


## Class: HistoryDice



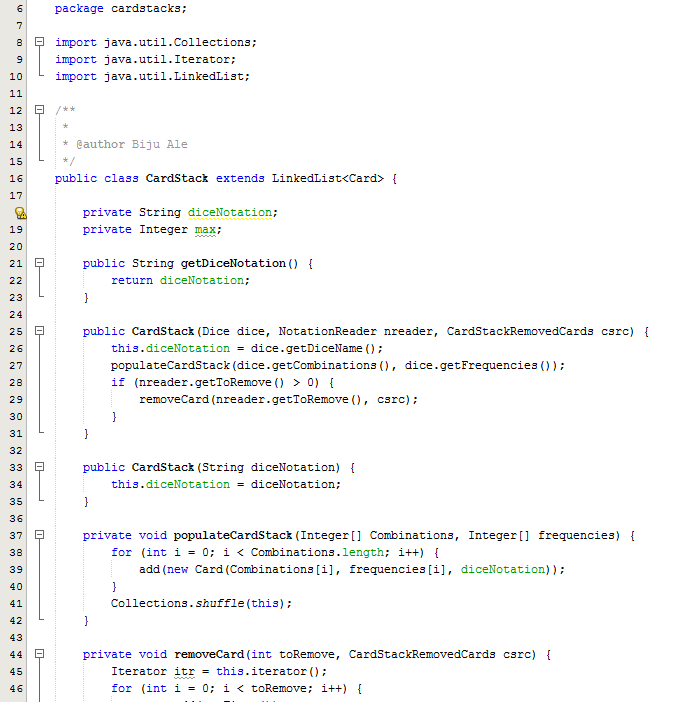
## Class: Card

[PLEASE TURN OVER]

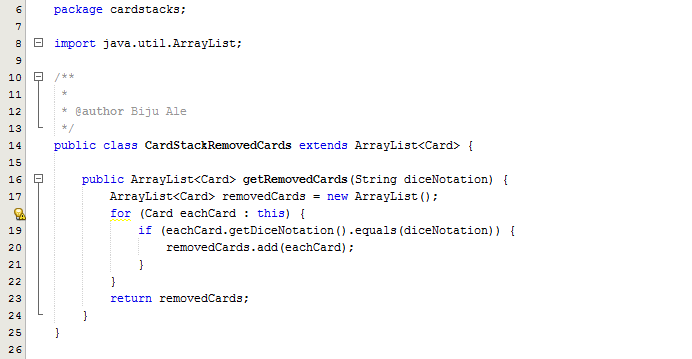


## Class: CardStack

[PLEASE TURN OVER]

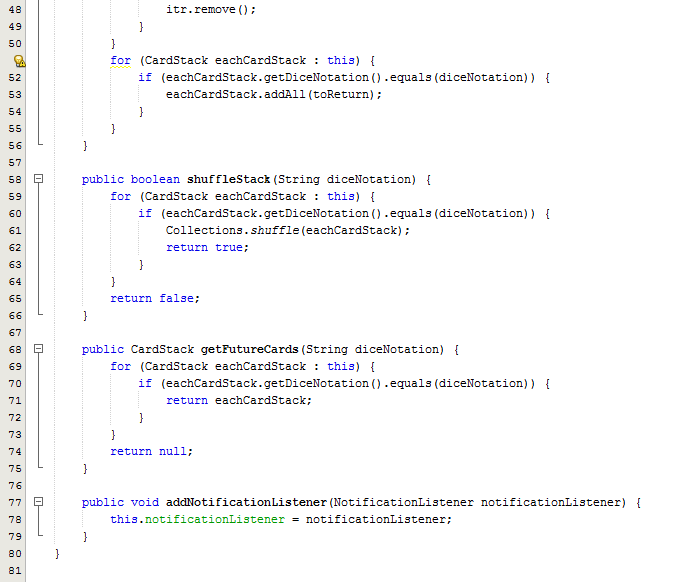
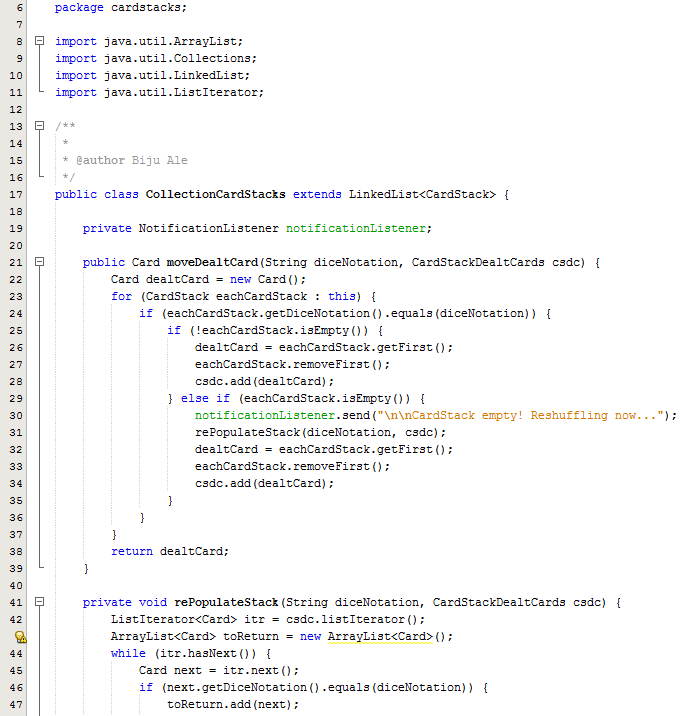


## Class: CardStackRemovedCards

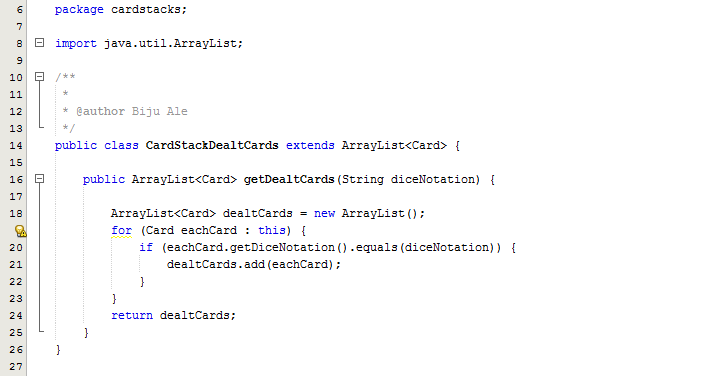


## Class: CollectionCardStacks

[PLEASE TURN OVER]

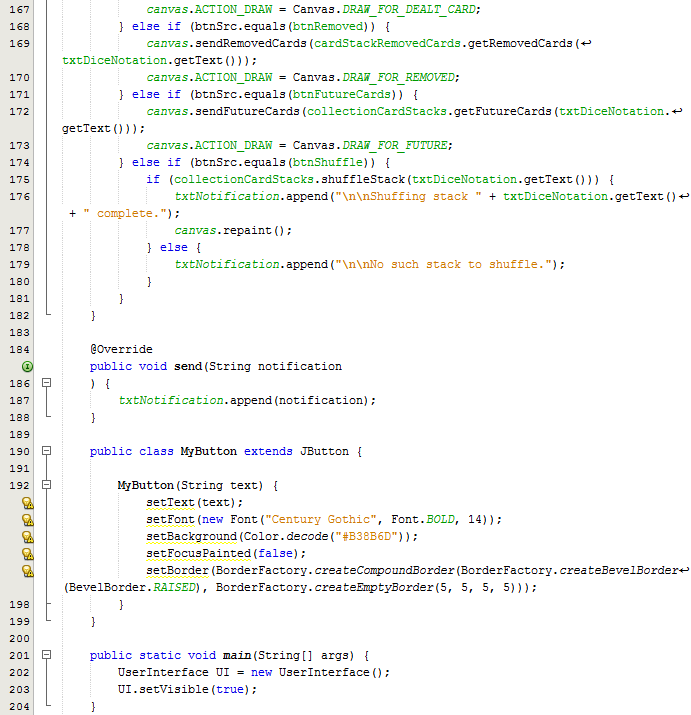
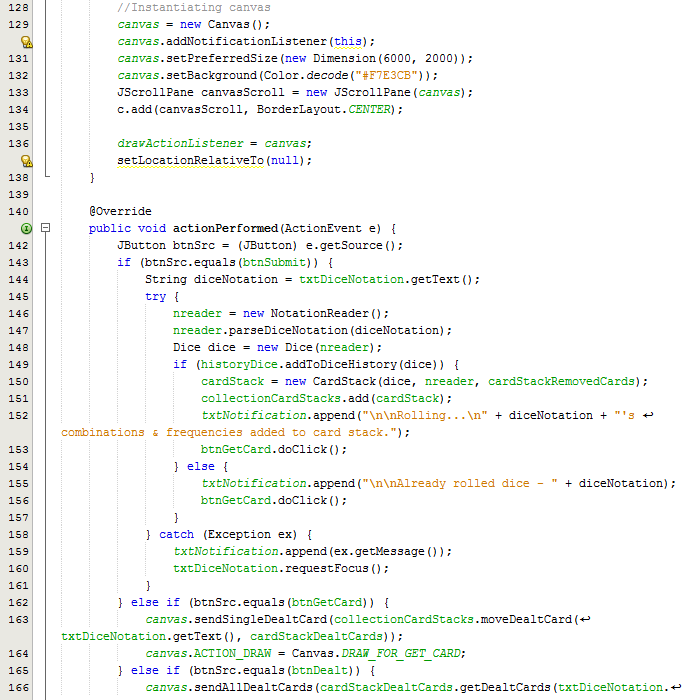
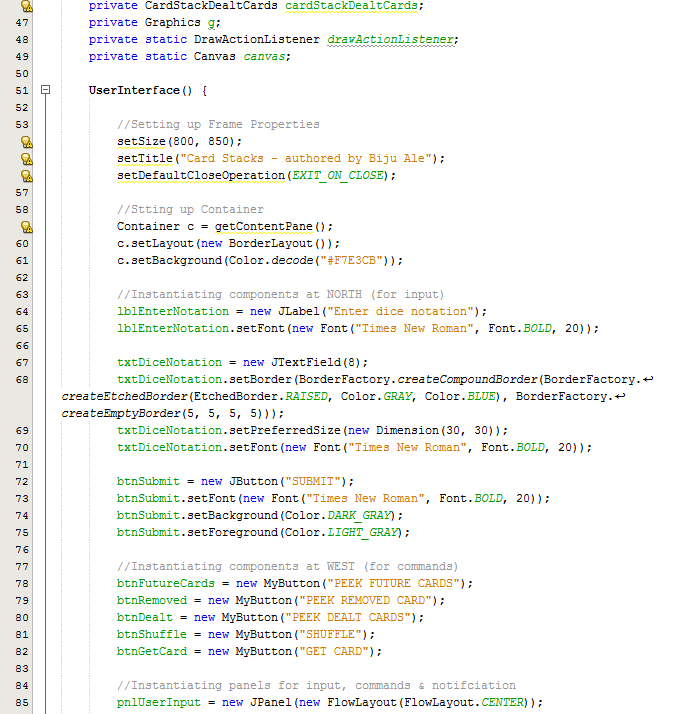
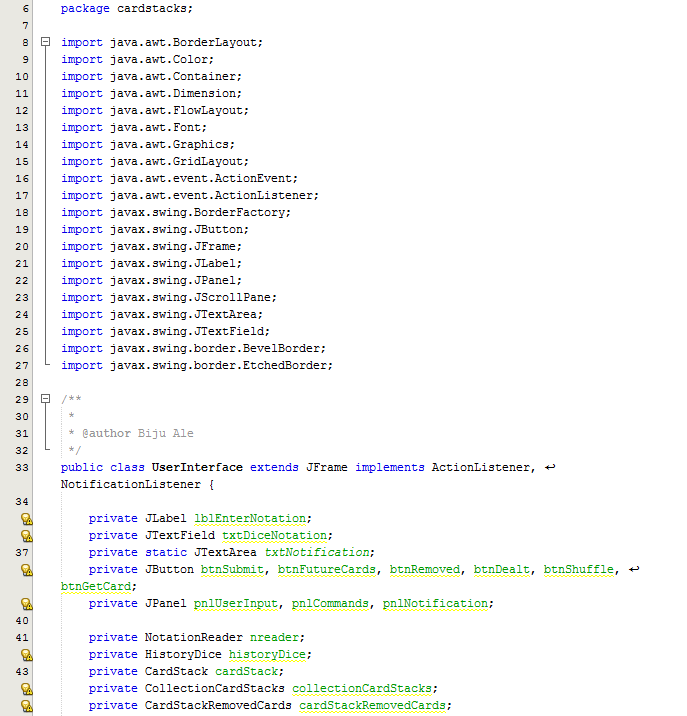


## Class: CardStackDealtCards

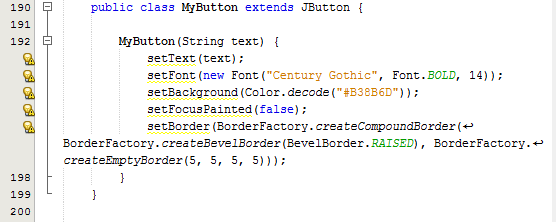


## Class: UserInterface

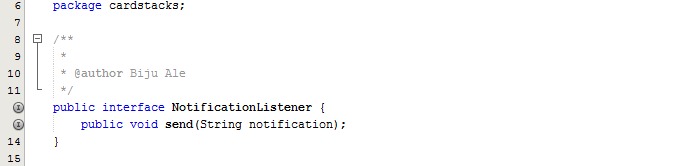
[PLEASE TURN OVER]



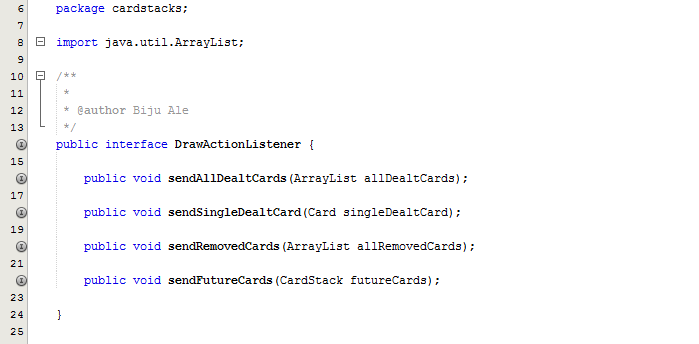
## Inner Class: MyButton

  
*Nested inside Class: UserInterface.*

## Interface: NotificationListener

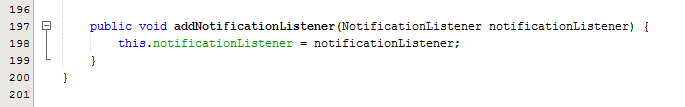
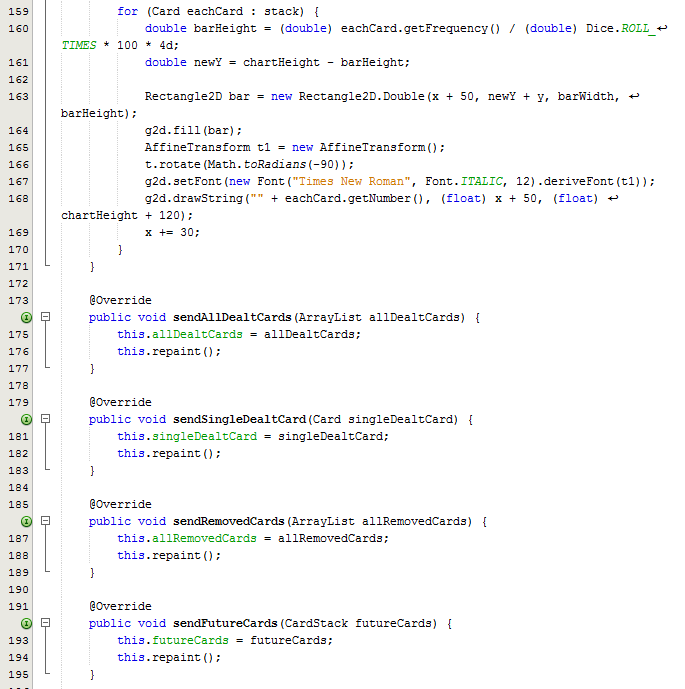
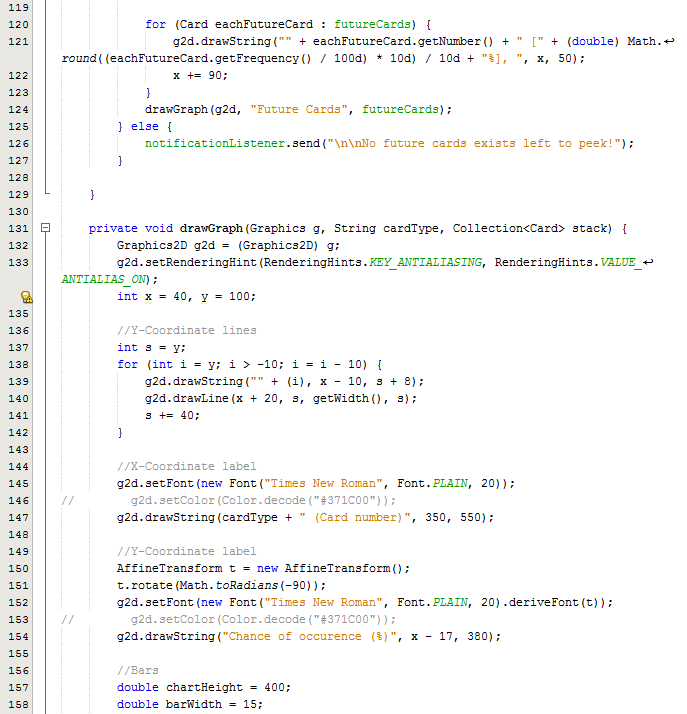
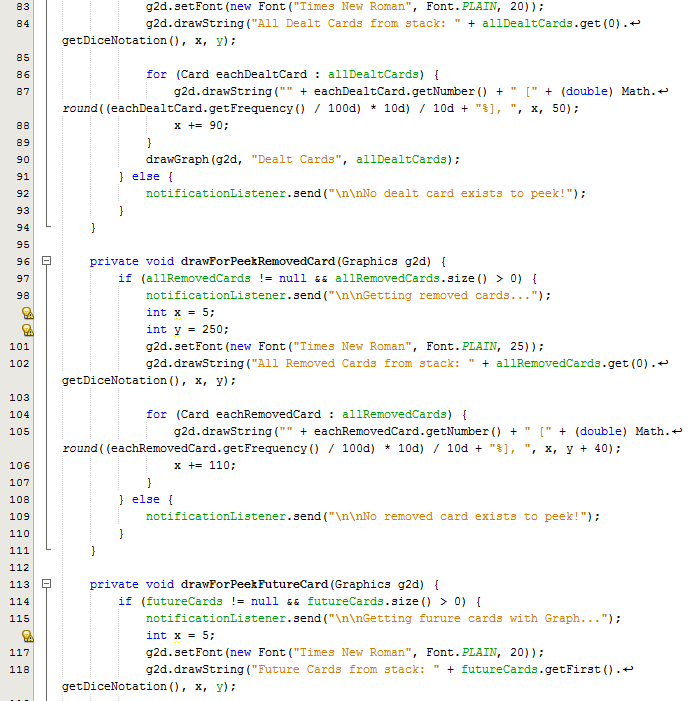
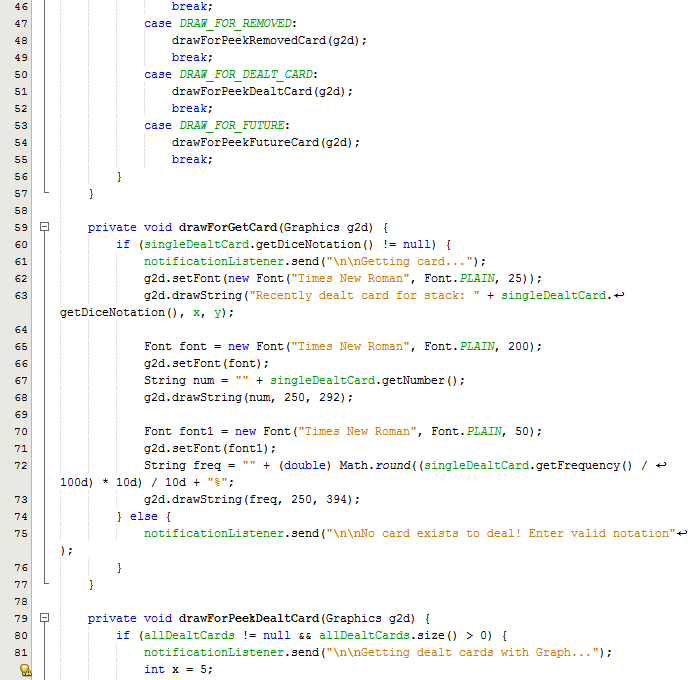
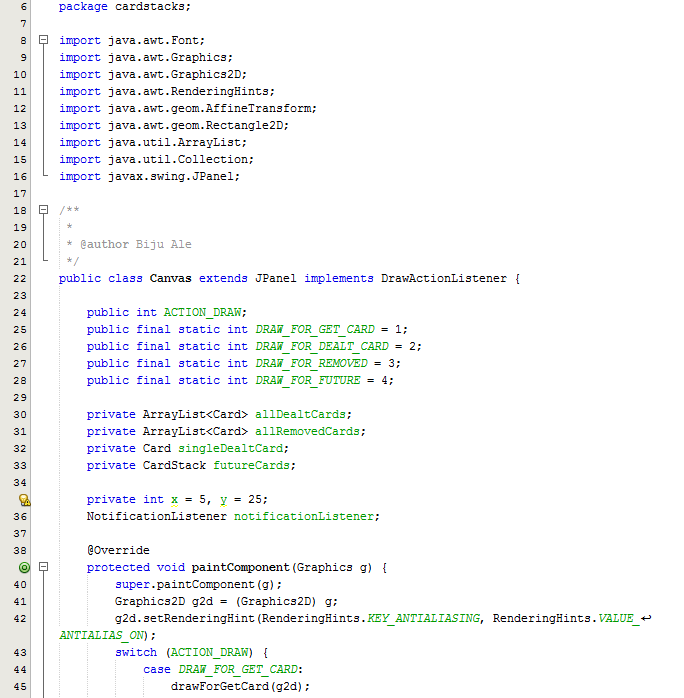


## Interface: DrawActionListener



## Class: Canvas

[PLEASE TURN OVER]



# Task-2 (Testing Data)

## Test Suite No.1

**Testingclass**: cardstacks.NotationReader  
**Testingtype**: 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!" |

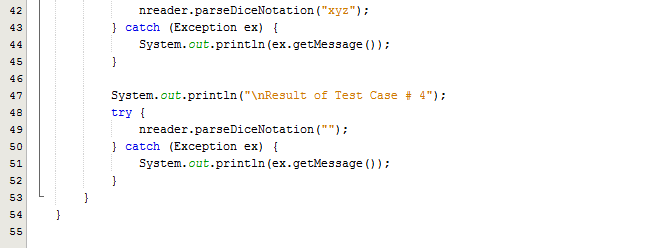
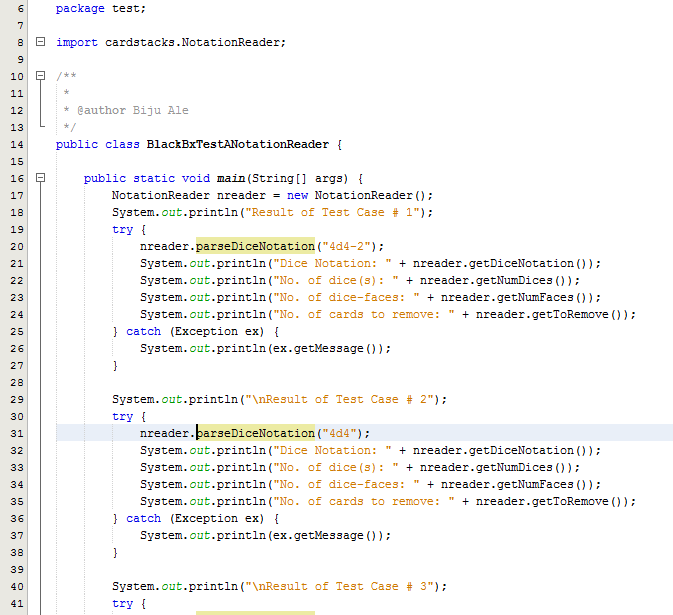
**Method of Equivalence partitioning:**

* 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.

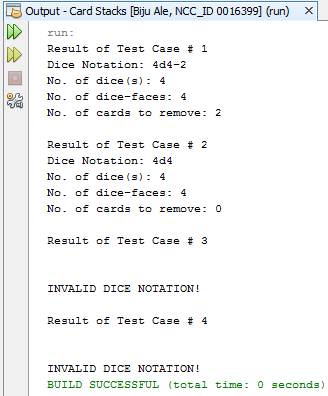
### Test Execution

#### 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

**Testingclass**: cardstacks.NotationReader  
**Testingtype**: White Box / Unit Testing

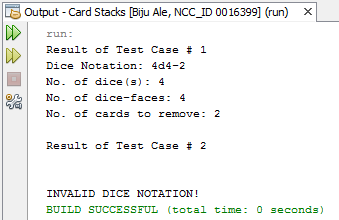
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void parseDiceNotation(String diceNotation) | 4d4-2 | Valid (2 delimiters –  ‘d’ & ‘-’) | Sets values of instance variables as following:   * diceNotation = 4d4-2 * numDices = 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

**Testingclass**: cardstacks.Dice  
**Testingtype**: 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" |

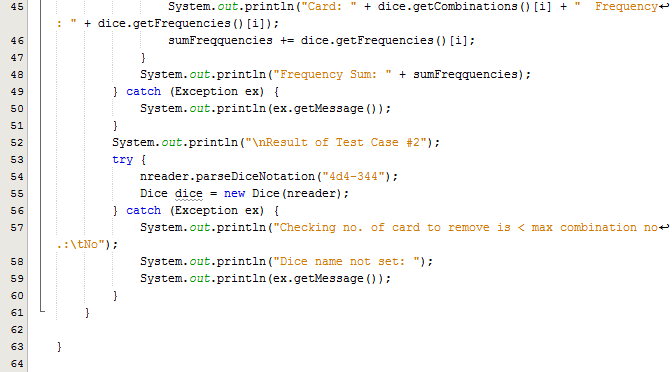
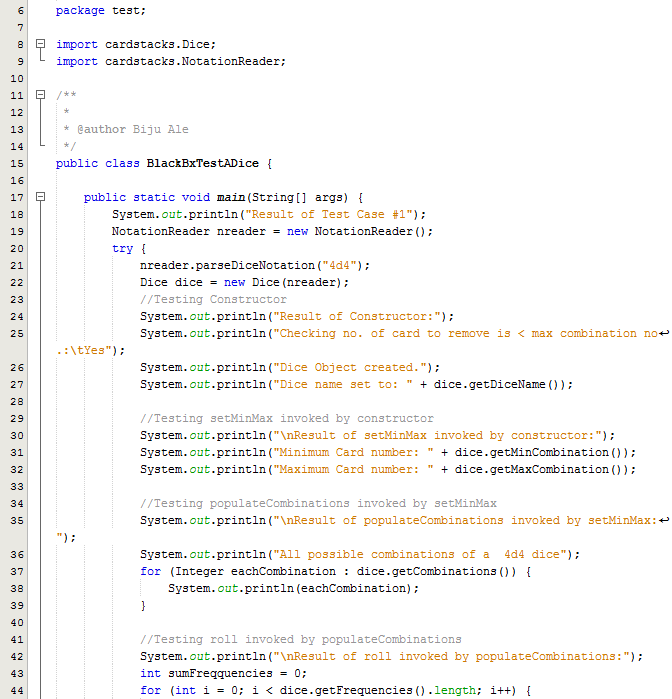
**Method of test data selection:**

* \*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.

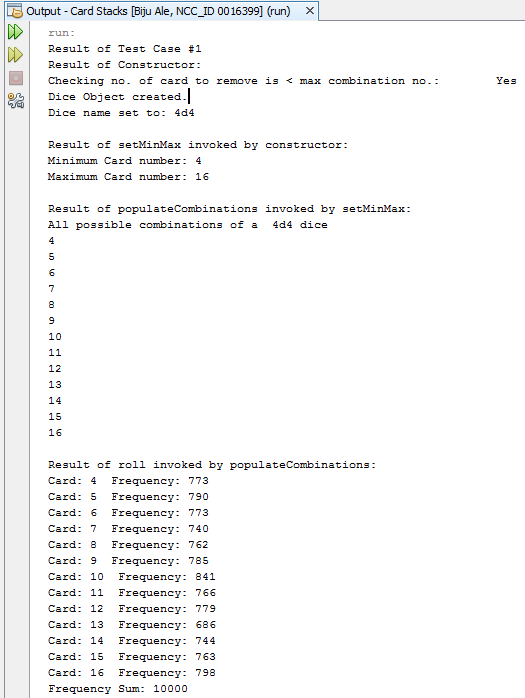
### Test Execution

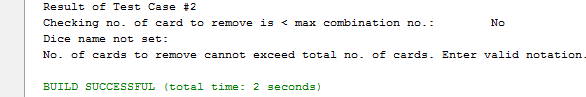
#### Source Code

**[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

**Testingclass**: cardstacks.Dice  
**Testingtype**: 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" |

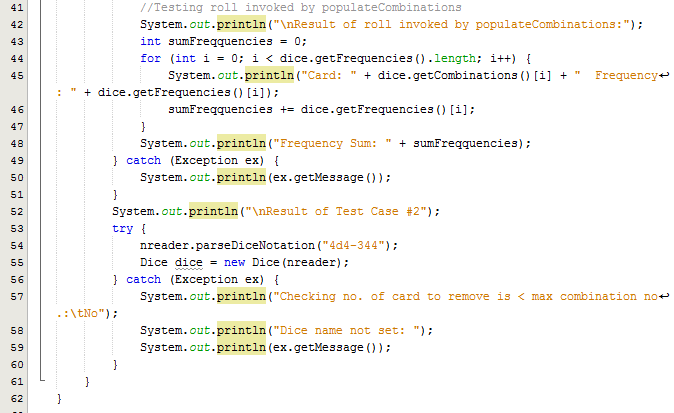
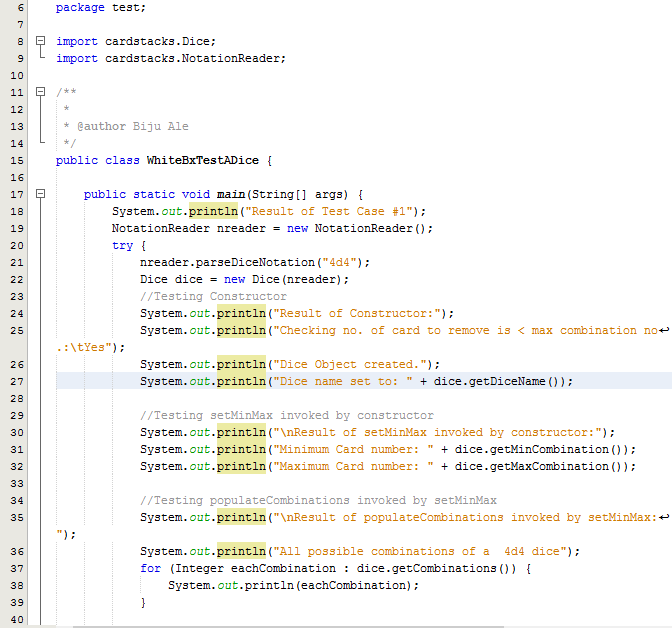
**Method of test data selection:**

* \*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

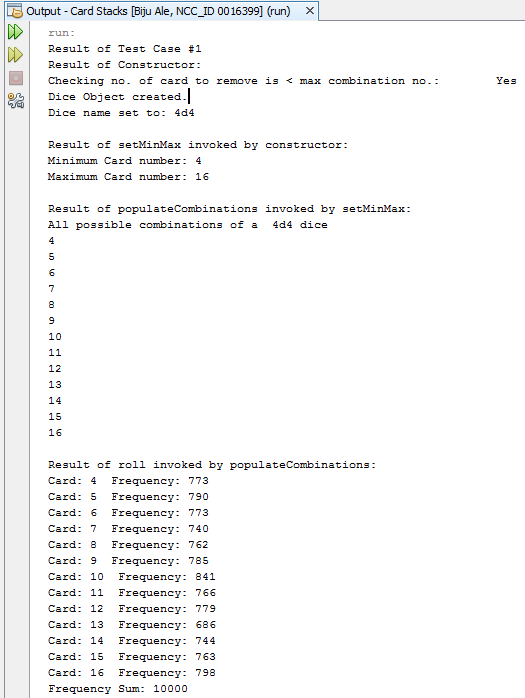
#### Source Code

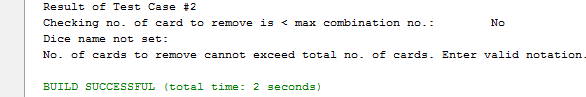
**[PLEASE TURN OVER]**



#### 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

**Testingclass**: cardstacks.Dice  
**Testingtype**: Black Box / Unit Testing

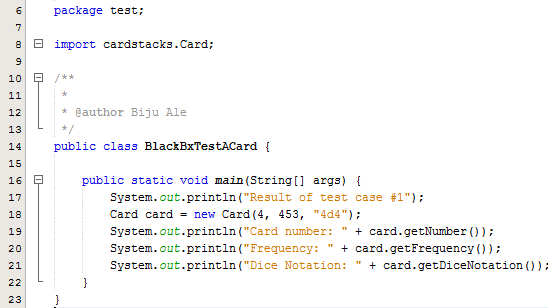
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | Card(int number, int frequency, String diceNotation) | (4,453,”4d4”) | Valid | Constructor should set the values of instance variables as following:   * number = 4 * frequency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

**Method of test data selection:**

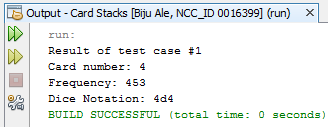
This constructor always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### 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

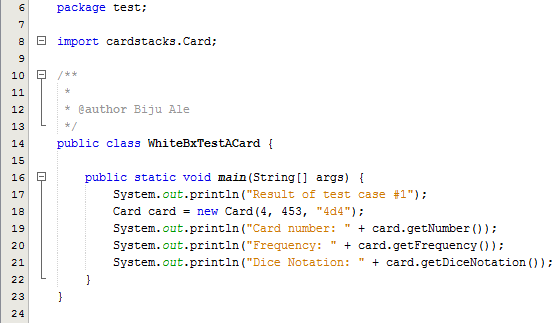
**Testingclass**: cardstacks.Dice  
**Testingtype**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **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 |

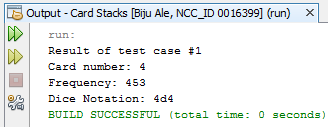
**Method of test data selection:**

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

**Testingclass**: cardstacks.CardStack  
**Testingtype**: 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. |

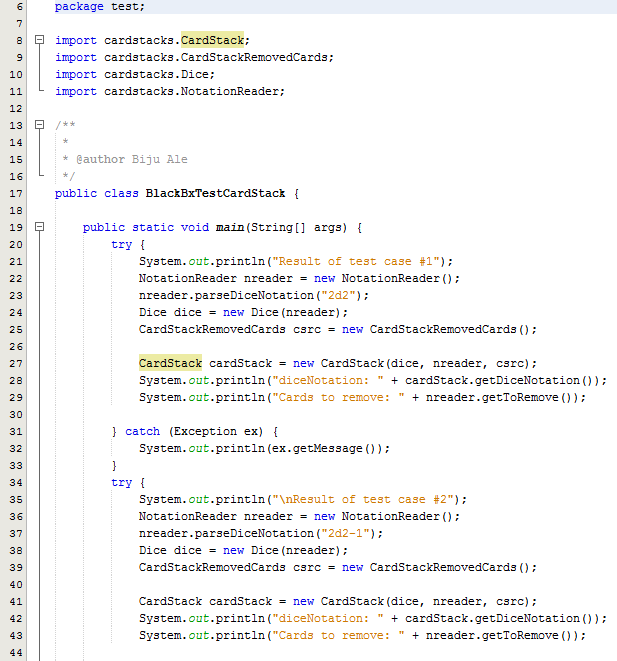
**Method of test data selection:**

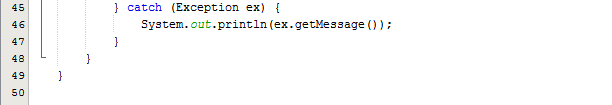
* \*Object of CardStackRemovedCards csrc is instantiated in GUI using default constructor. It does not contain any data member/instance variables. No. of card to remove here, is 0.
* \*\*’nreader’ contains data member, where card to remove is greater than 0.
* It never receives null as input, because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

### Test Execution

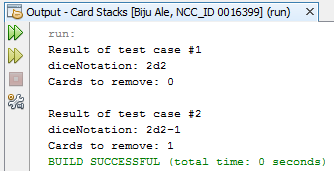
#### Source Code

**[PLEASE TURN OVER]**



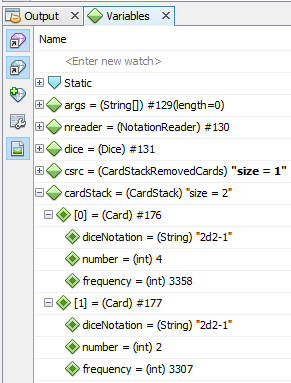
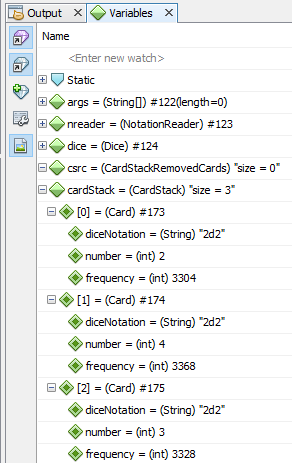


#### Output



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

**Testingclass**: cardstacks.CardStack  
**Testingtype**: 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. |

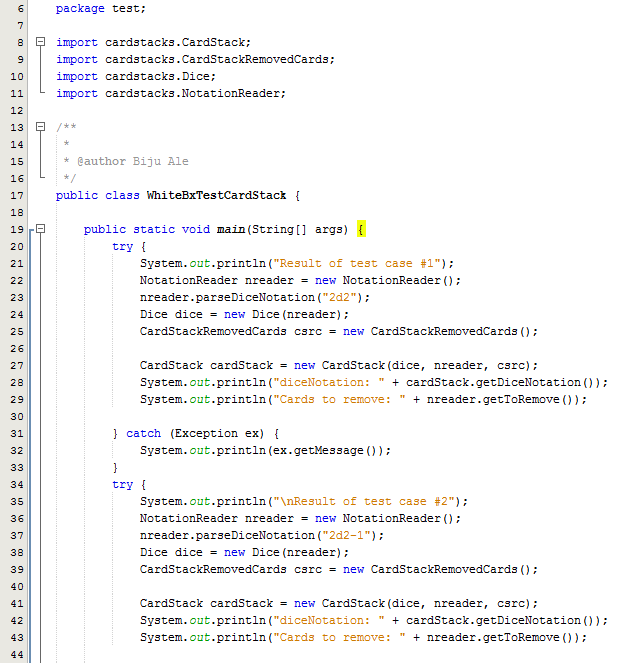
**Method of test data selection:**

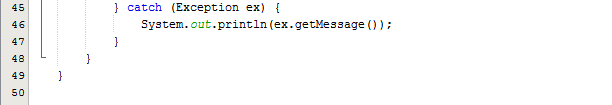
* \*Object of CardStackRemovedCards csrc is instantiated in GUI using default constructor. It does not contain any data member/instance variables. No. of card to remove here, is 0.
* \*\*’nreader’ contains data member, where card to remove is greater than 0.
* It never receives null as input, because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

### Test Execution

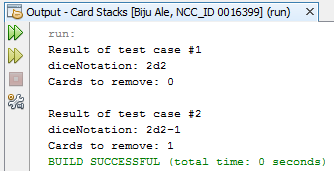
#### Source Code

**[PLEASE TURN OVER]**



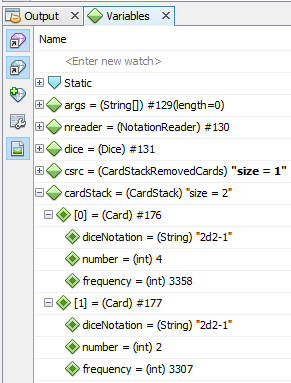
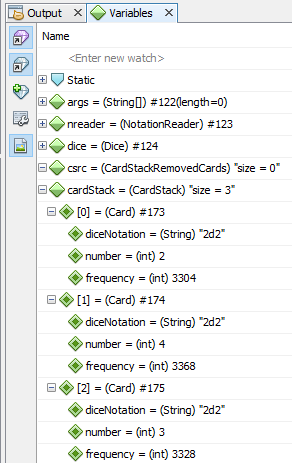


#### Output



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

**Testingclass**: cardstacks.CardStackRemovedCards  
**Testingtype**: 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. |

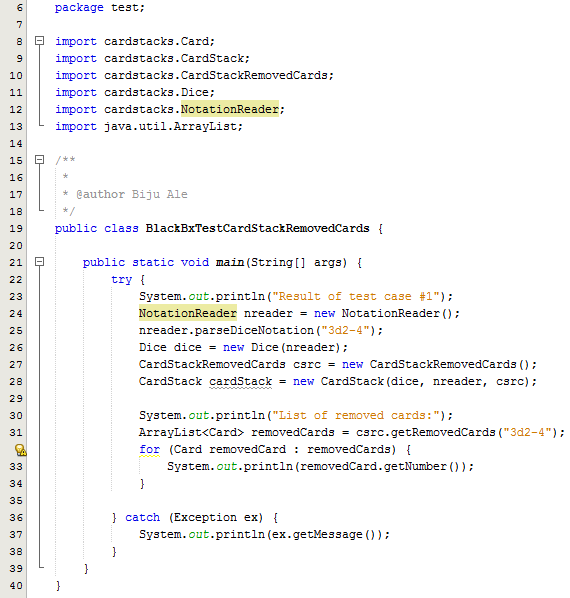
**Method of test data selection:**

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.

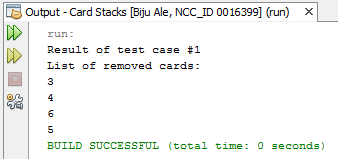
### Test Execution

#### 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

**Testingclass**: cardstacks.CardStackRemovedCards  
**Testingtype**: 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. |

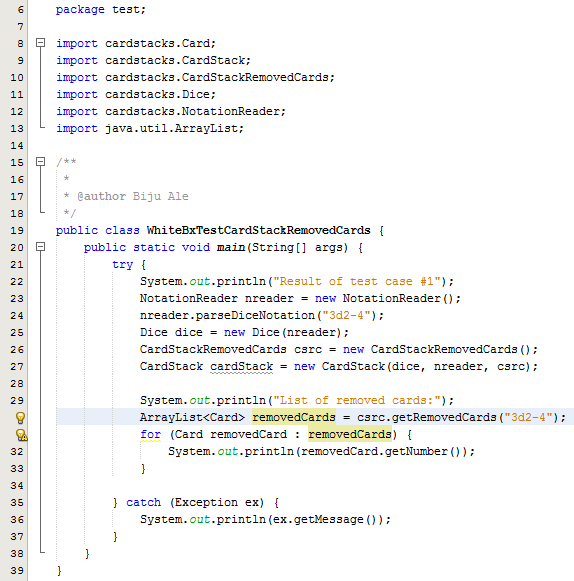
**Method of test data selection:**

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.

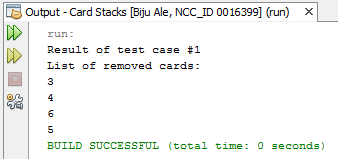
### Test Execution

#### 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

**Testingclass**: cardstacks.HistoryDice  
**Testingtype**: Black Box / Unit Testing

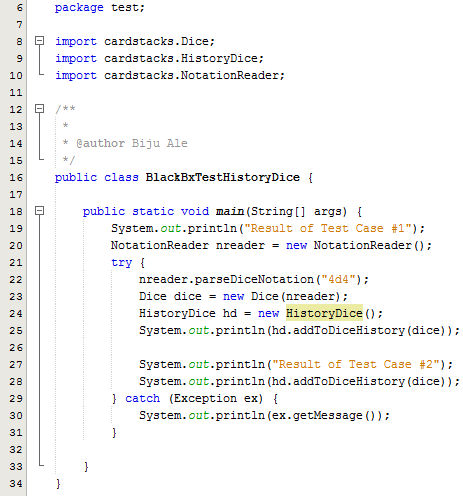
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean addToDiceHistory(Dice dice) | dice | Valid object | Returns TRUE after adding the dice to history. |
| 2 | boolean addToDiceHistory(Dice) | dice | Invalid object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

**Method of test data selection:**

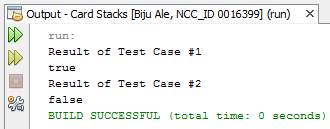
This method does not receive null, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the null input from GUI.

### 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

**Testingclass**: cardstacks.HistoryDice  
**Testingtype**: White Box / Unit Testing

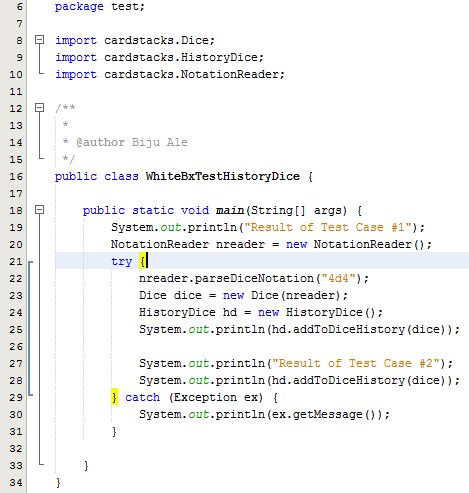
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean addToDiceHistory(Dice dice) | dice | Valid object | Returns TRUE after adding the dice to history. |
| 2 | boolean addToDiceHistory(Dice) | dice | Invalid object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

**Method of test data selection:**

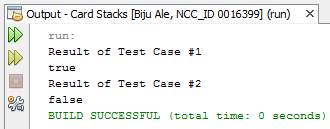
This method does not receive null input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates null input from GUI.

### 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

**Testingclass**: cardstacks.CardStackDealtCards  
**Testingtype**: Black Box / Unit Testing

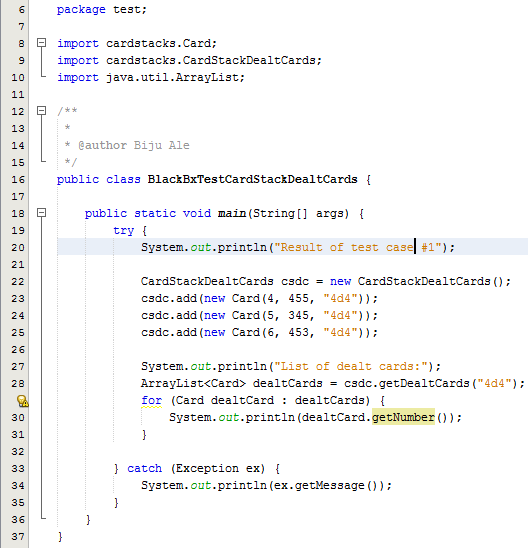
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | ArrayList<Card> getDealtCards(String diceNotation) | “4d4” | Valid string | Returns correct no. i.e. of dealt cards. i.e. 4 dealt cards. |

**Method of test data selection:**

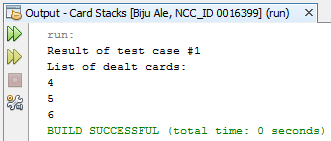
This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### 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

**Testingclass**: cardstacks.CardStackDealtCards  
**Testingtype**: 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:**

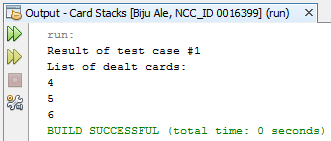
This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.

### 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

**Testingclass**: cardstacks.CollectionCardStacks   
**Testingtype**: 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. |

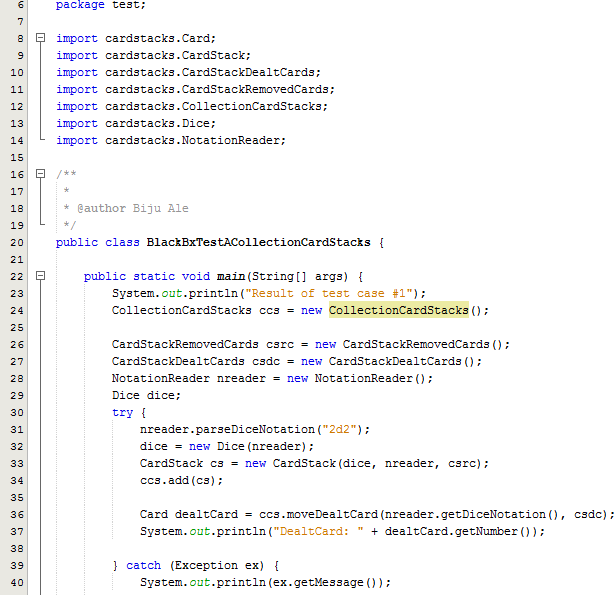
**Method of test data selection:**

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.

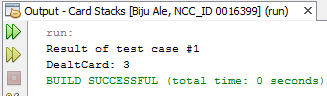
### Test Execution

#### 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

**Testingclass**: cardstacks.CollectionCardStacks  
**Testingtype**: 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. |

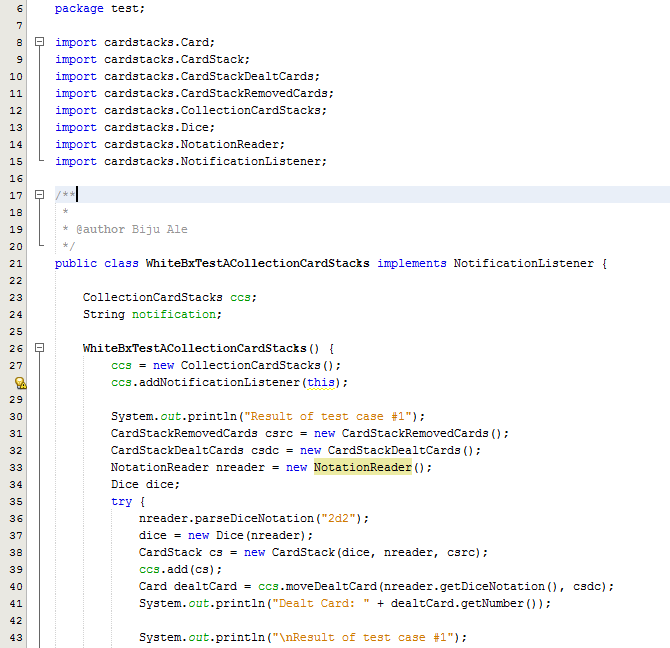
**Method of test data selection:**

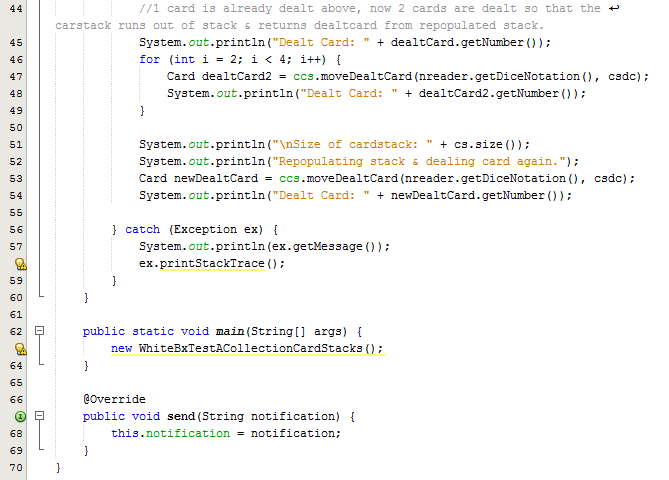
* This method always receives valid input only, because it is only invoked after NotationReader class that was checked in Test Suite No.1 & 2, validates the input from GUI.
* Both test data are same, but in the second case, the test data is passed 3 times so that the 2d2 cardstack runs out of cards.

### Test Execution

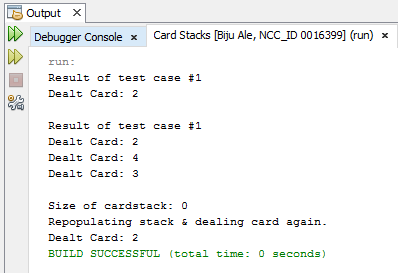
#### Source Code

**[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

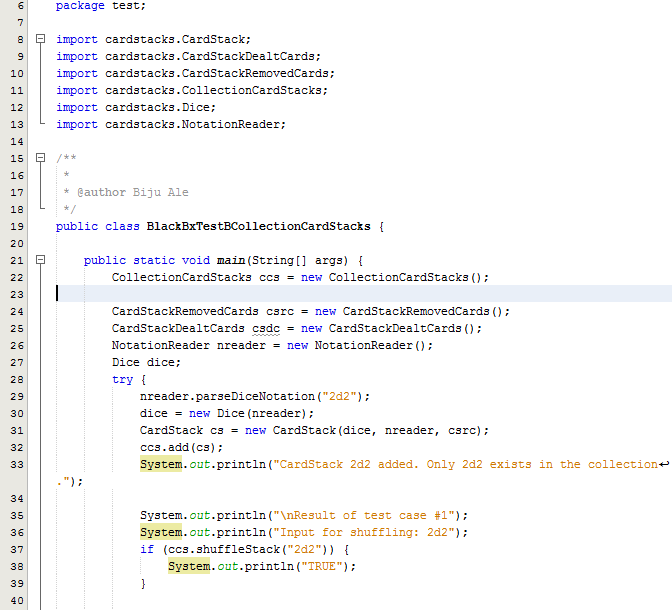
**Testingclass**: cardstacks.NotationReader  
**Testingtype**: Black Box / Unit Testing

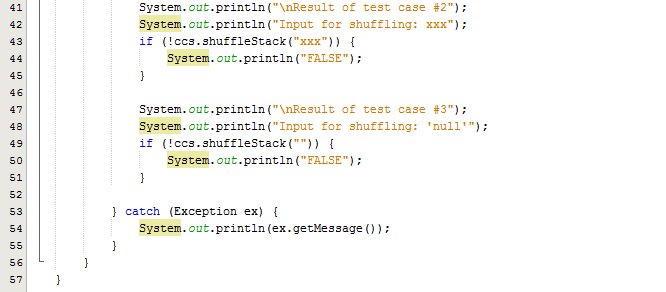
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | boolean shuffleStack(String diceNotation) | “4d4” | Valid string | Returns true after changing order of 4d4 cardstack. |
| 2 | boolean shuffleStack(String diceNotation)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

### Test Execution

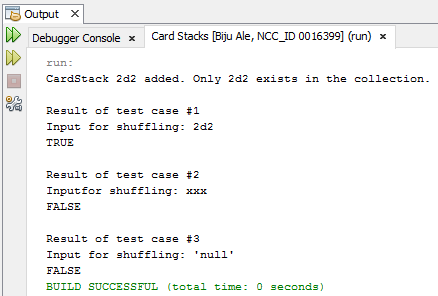
#### Source Code

**[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.

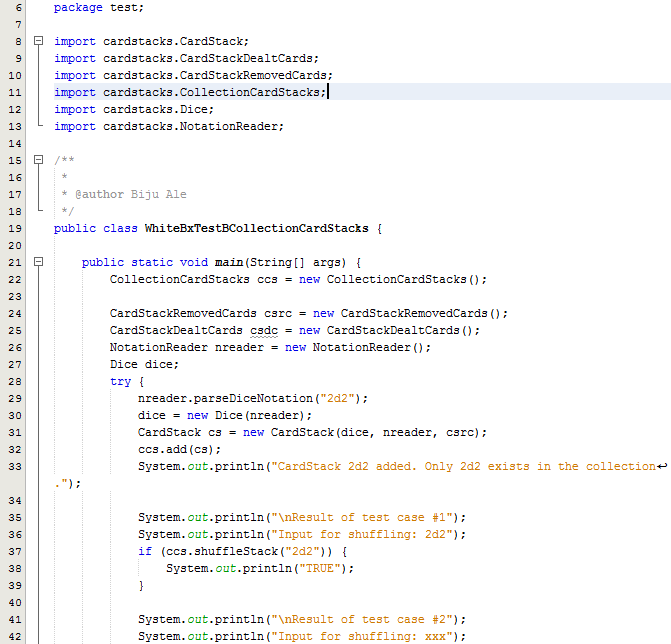
## Test Suite No.18

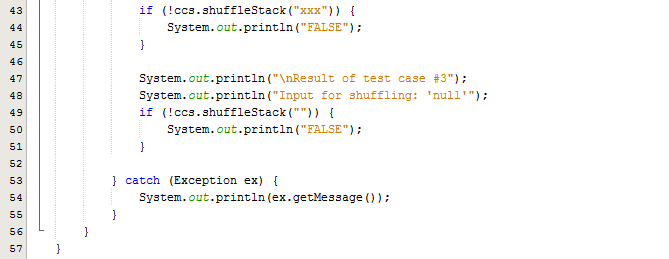
**Testingclass**: cardstacks.NotationReader  
**Testingtype**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **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. |

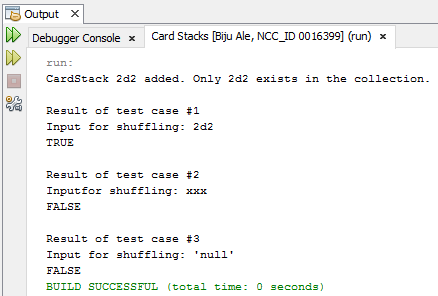
### Test Execution

#### 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

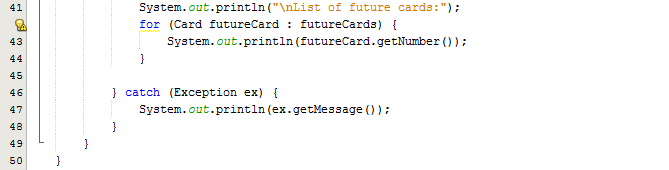
**Testingclass**: cardstacks.NotationReader  
**Testingtype**: Black Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **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. |

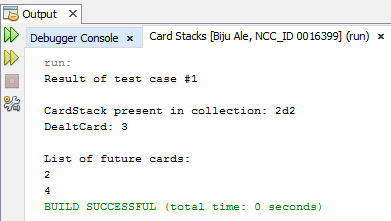
### Test Execution

#### 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

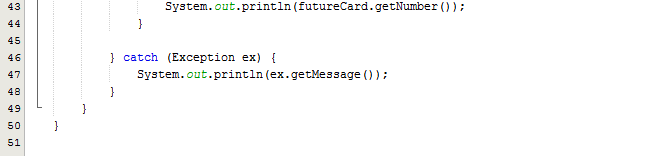
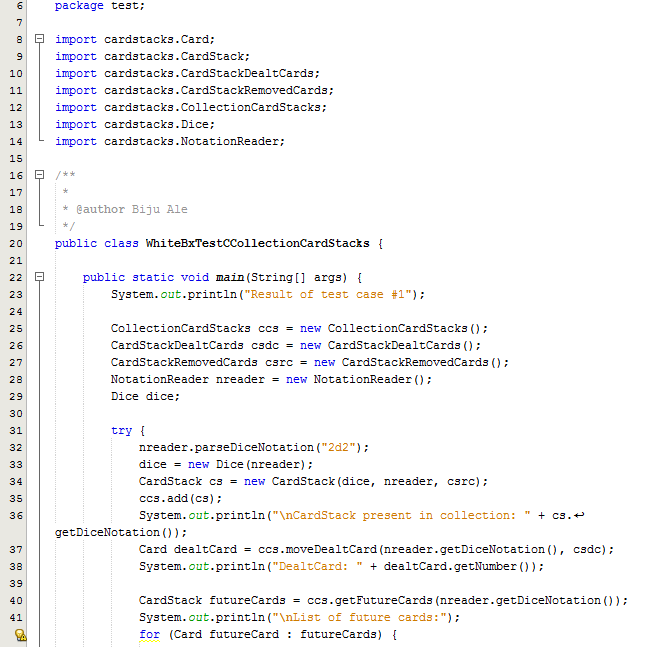
**Testingclass**: cardstacks.NotationReader  
**Testingtype**: White Box / Unit Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **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. |

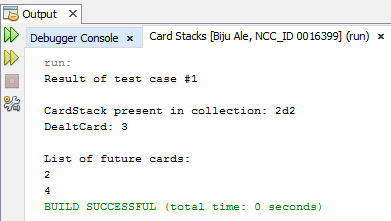
### Test Execution

#### 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.

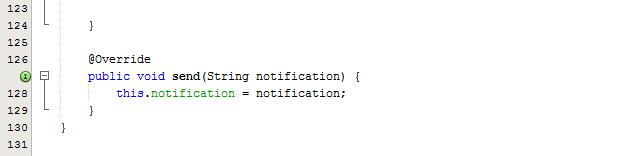
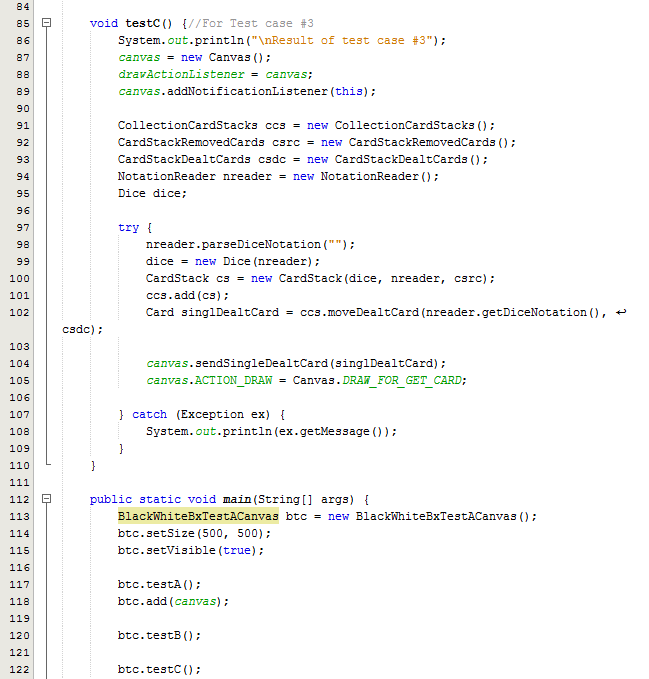
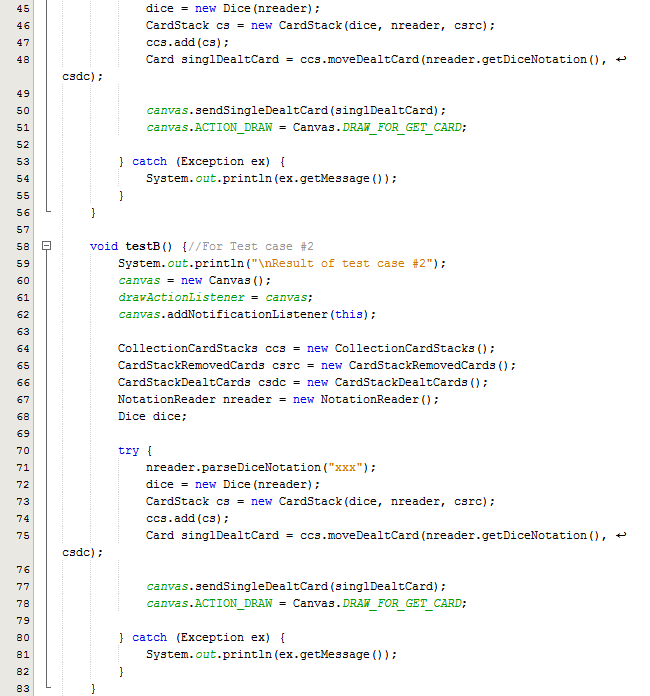
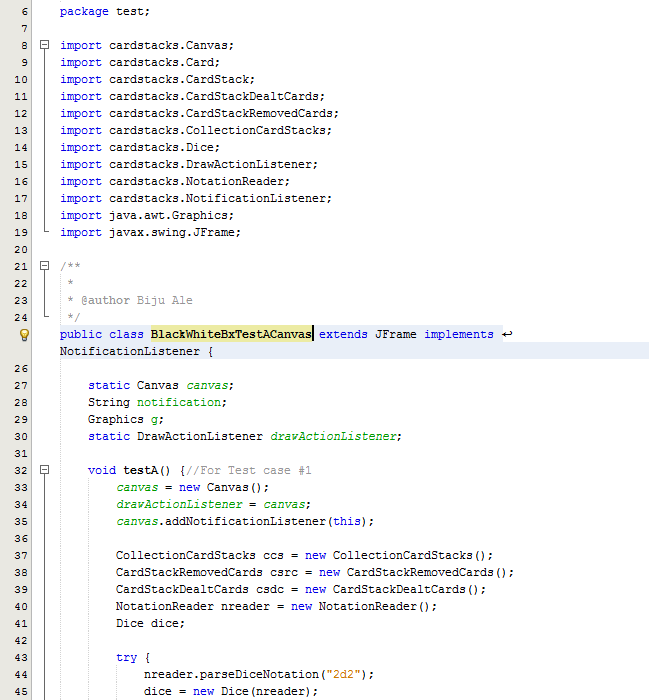
## Test Suite No.21

**Testingclass**: cardstacks.Canvas  
**Testingtype**: Black Box & White Box / Unit Testing

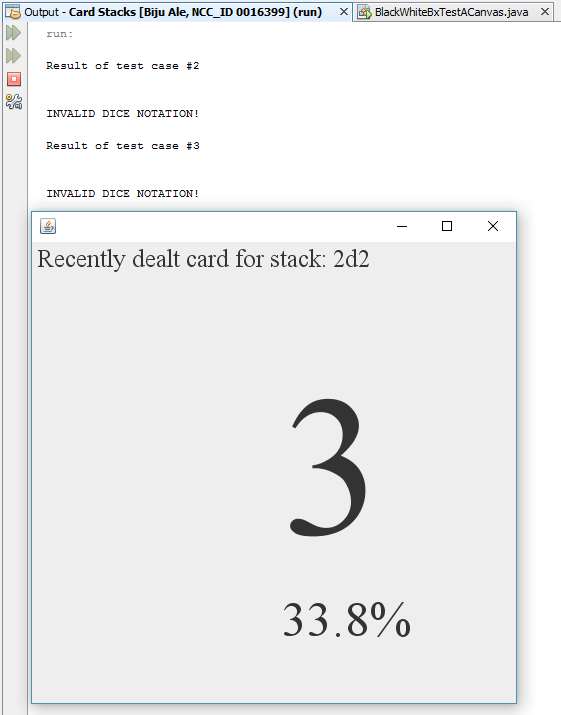
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | void sendSingleDealtCard(Card singleDealtCard) | singleDealtCard | Valid object | Draws single dealt card on JPanel |
| 2 | void sendSingleDealtCard(Card singleDealtCard) | 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.

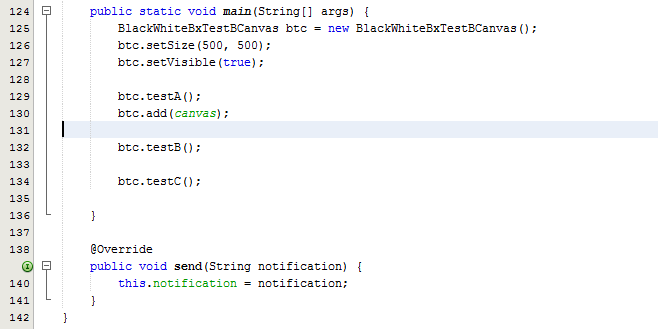
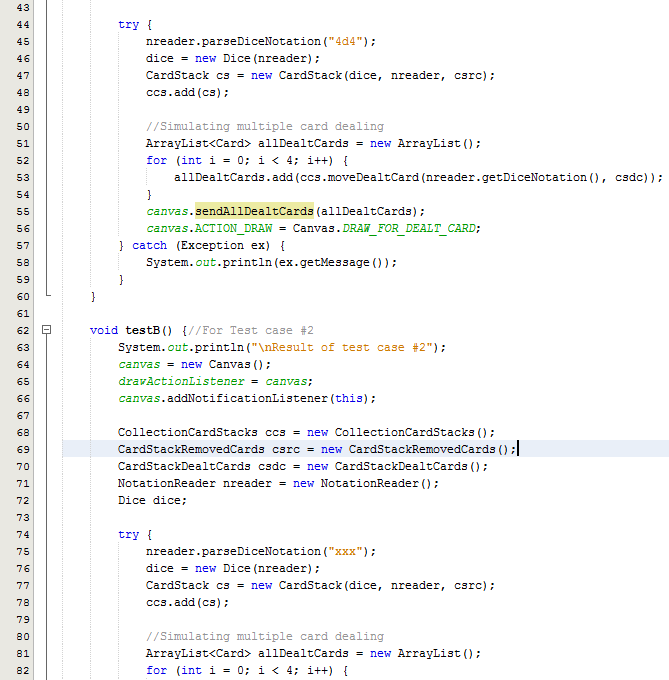
## Test Suite No.22

**Testingclass**: cardstacks.Canvas  
**Testingtype**: Black Box & White Box / Unit Testing

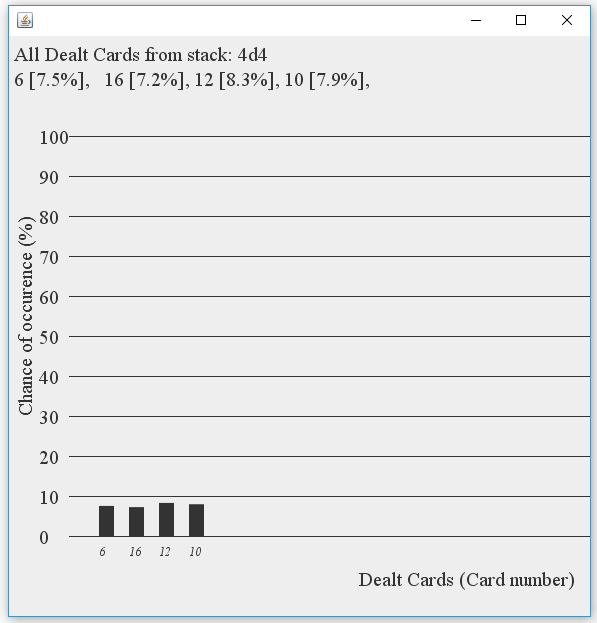
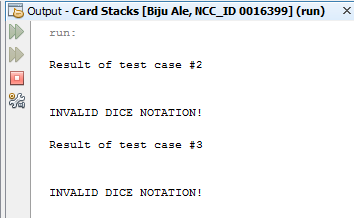
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendAllDealtCards(ArrayList allDealtCards) | allDealtCards | Valid object | Draws Graph of all dealt cards on JPanel |
| 2 | sendAllDealtCards(ArrayList allDealtCards) | 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.

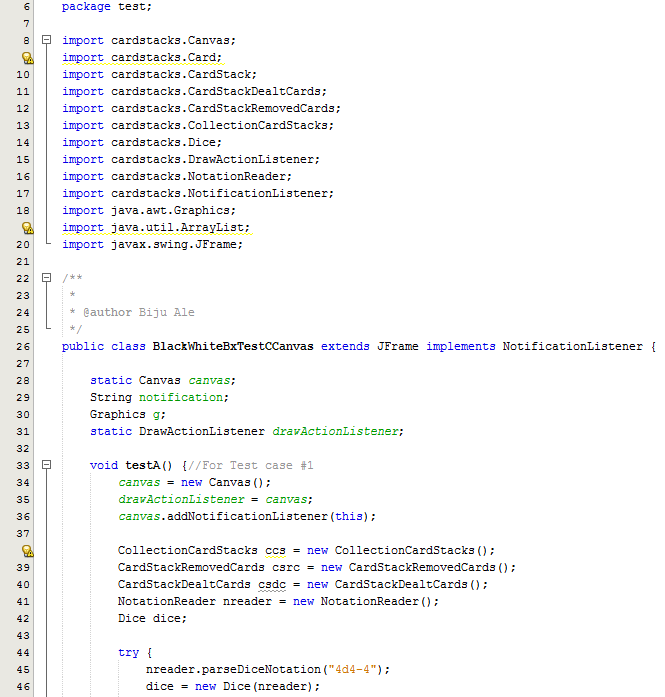
## Test Suite No.23

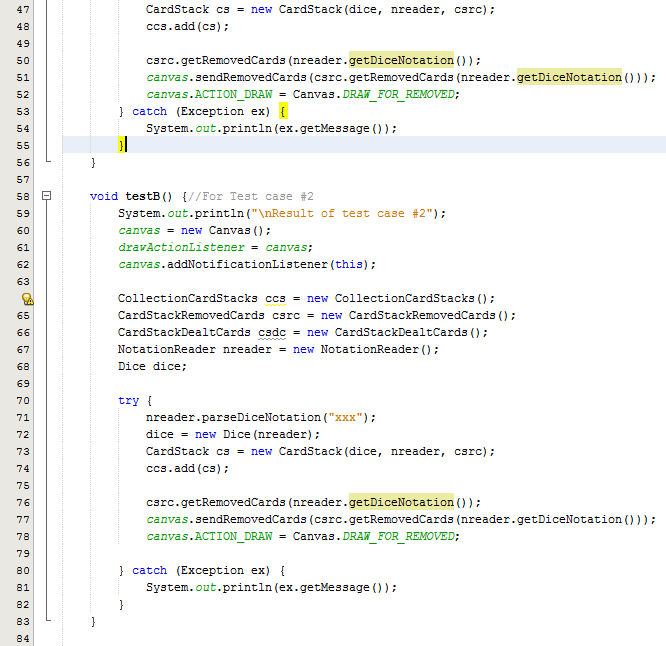
**Testingclass**: cardstacks.Canvas  
**Testingtype**: Black Box & White Box / Unit Testing

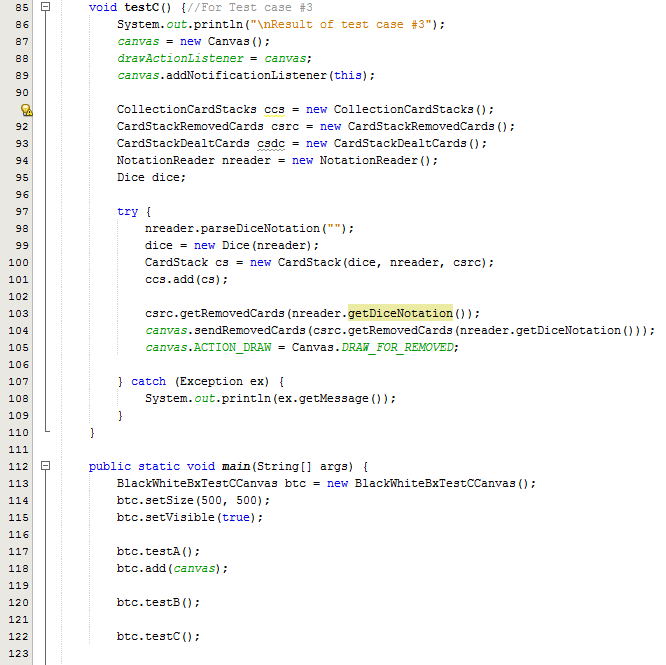
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendRemovedCards(ArrayList allRemovedCards) | allRemovedCards | Valid object | Draws all removed cards on JPanel |
| 2 | sendRemovedCards(ArrayList allRemovedCards) | 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

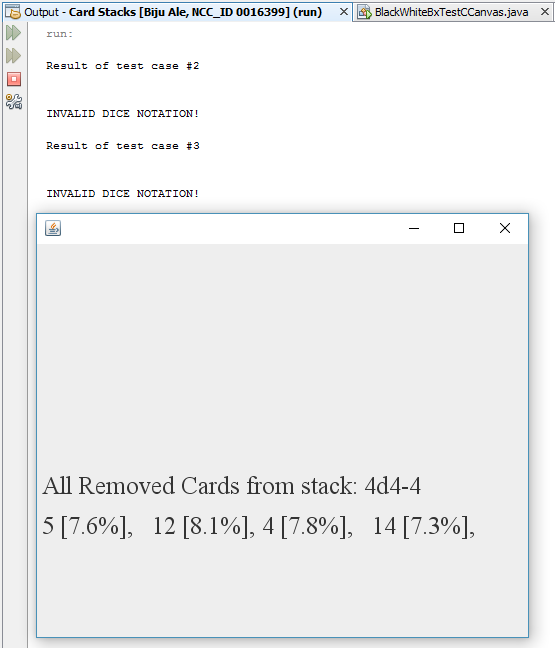
#### Source Code







#### 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.

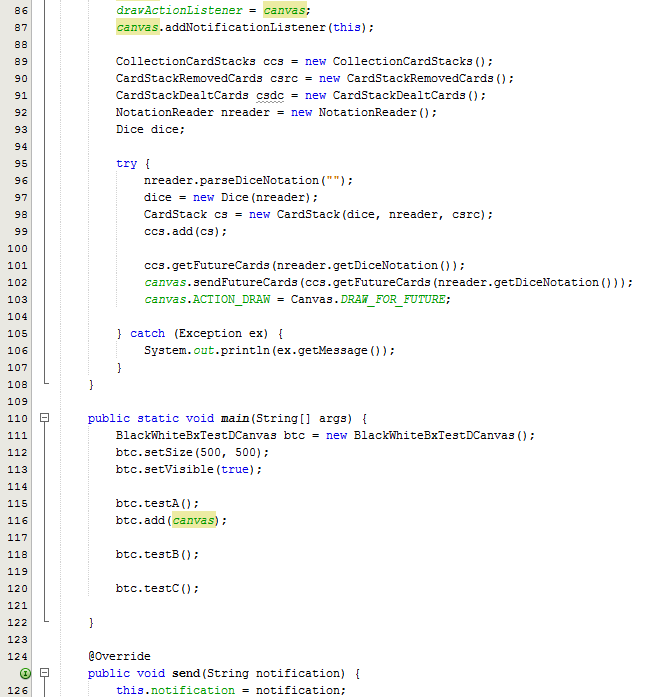
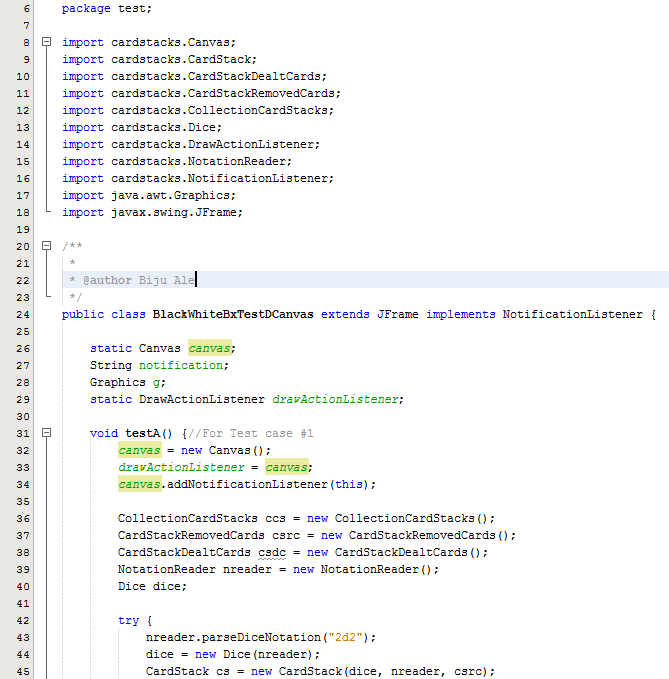
## Test Suite No.24

**Testingclass**: cardstacks.Canvas  
**Testingtype**: Black Box & White Box / Unit Testing

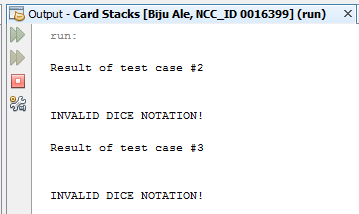
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Method** | **Test Data** | **Input type** | **Expected Outcome** |
| 1 | sendFutureCards(CardStack futureCards) | futureCards | Valid object | Draws graph of all future cards on JPanel |
| 2 | sendFutureCards(CardStack futureCards) | futureCards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

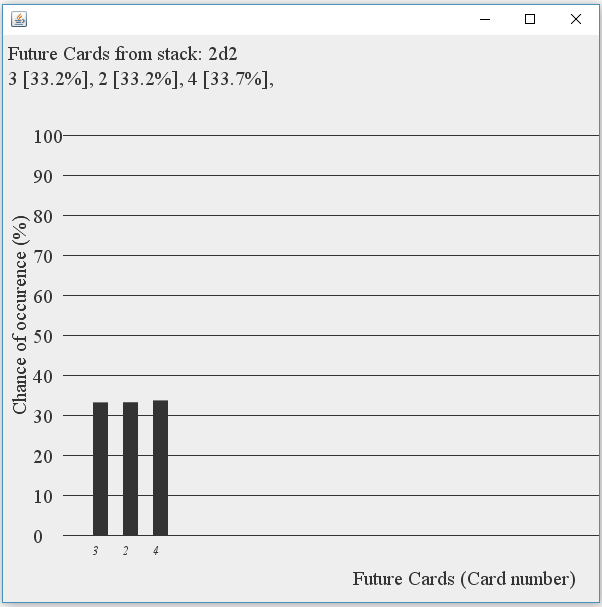
### Test Execution

#### 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 | sendFutureCards(CardStack futureCards) | futureCards | Valid object | Draws graph of all future cards on JPanel | Yes |
| 2 | sendFutureCards(CardStack futureCards) | futureCards | Null (object has null data member i.e. null dice notation) | Throws exception with message –  “INVALID DICE NOTATION” | Yes |
| 3 | sendFutureCards(CardStack futureCards) | futureCards | Invalid (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.

## Final Test Suite No.25

**Testingclass**: cardstacks.NotationReader  
**Testingtype**: Integration Testing

Test Suite No. 1 to 24 was designedtounit test all the methods of classes that make up the application. Now, integration testing is required to test the communication between objects. Main functionalities are shown executed as a final application, below.

### Test Execution

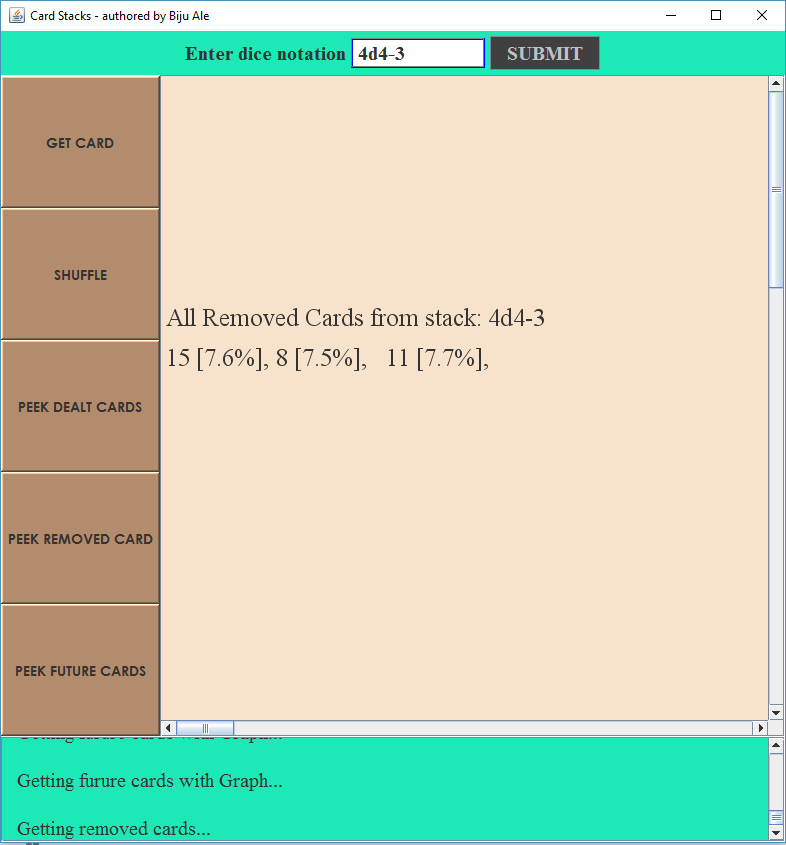
Please turn over….



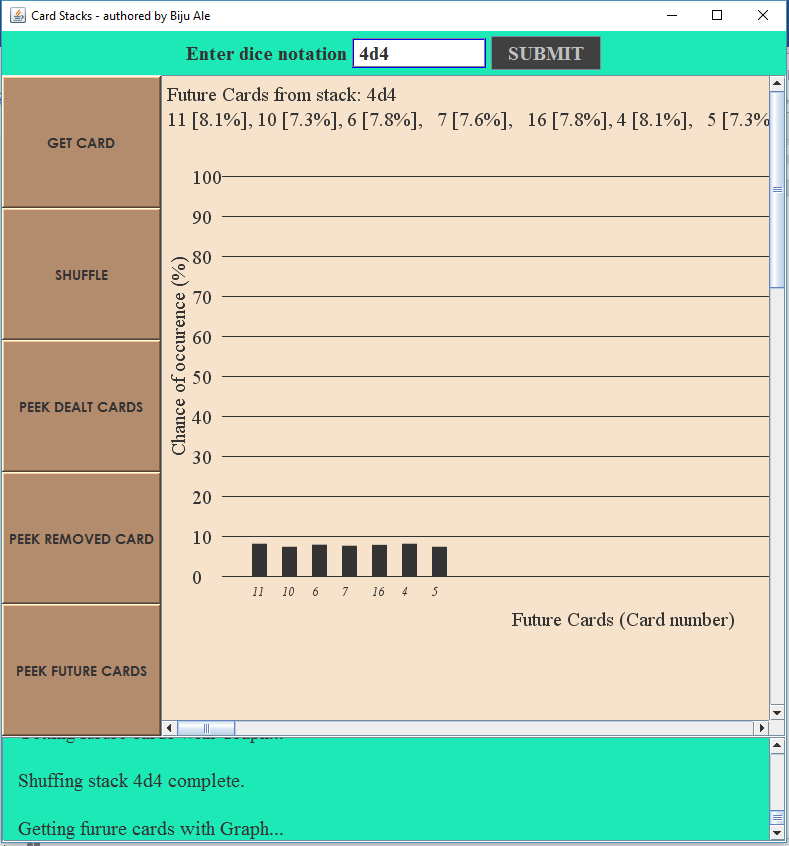
As shown in the above figure, invalid input dice notation triggered exception handling which threw the correct message.



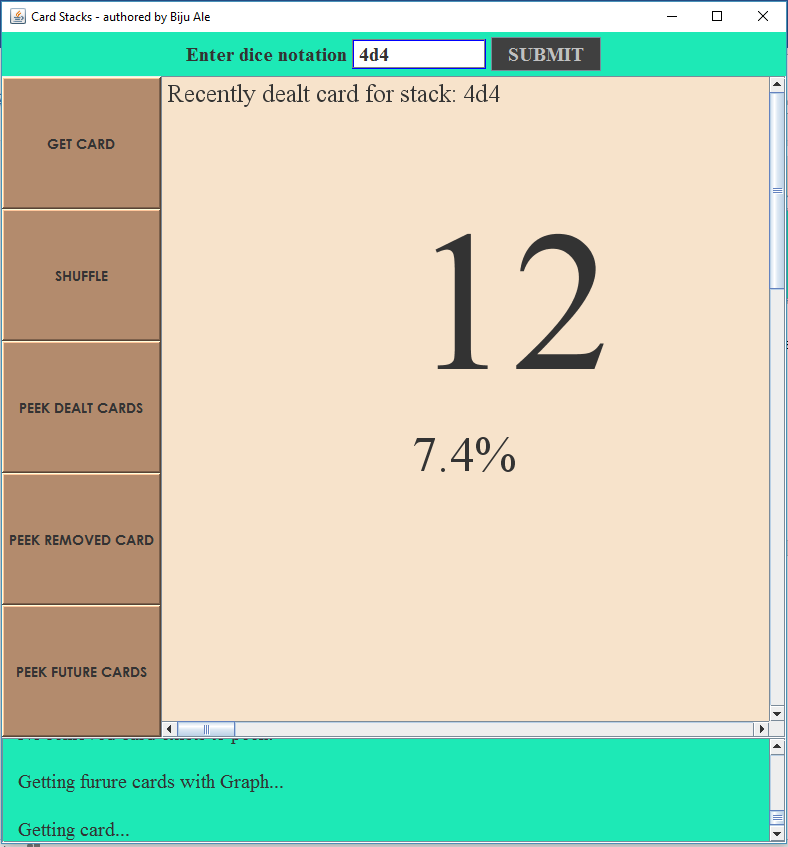
As shown in the above figure, null input for dice notation triggered exception handling which threw the correct message.



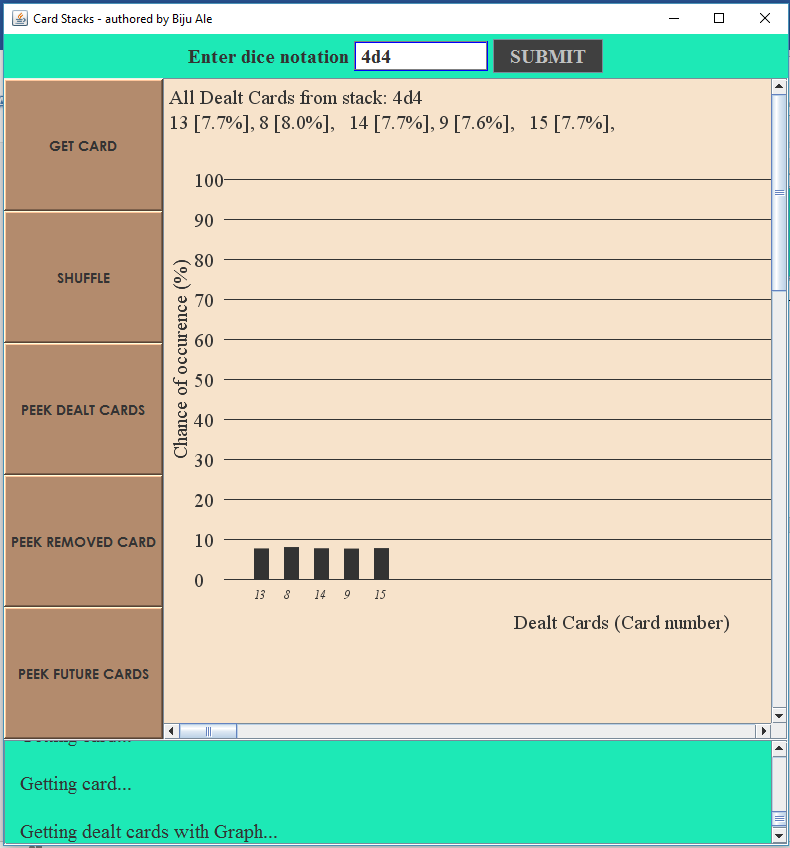
As shown in the above figure, application generated the correct number of cards to remove. Parsing of dual delimiter input – “d” and “-” was correctly executed.



As shown in the above figure, graph of future cards is correctly generated.



As shown in the above figure, a card is immediately dealt off once valid dice notation is passed.

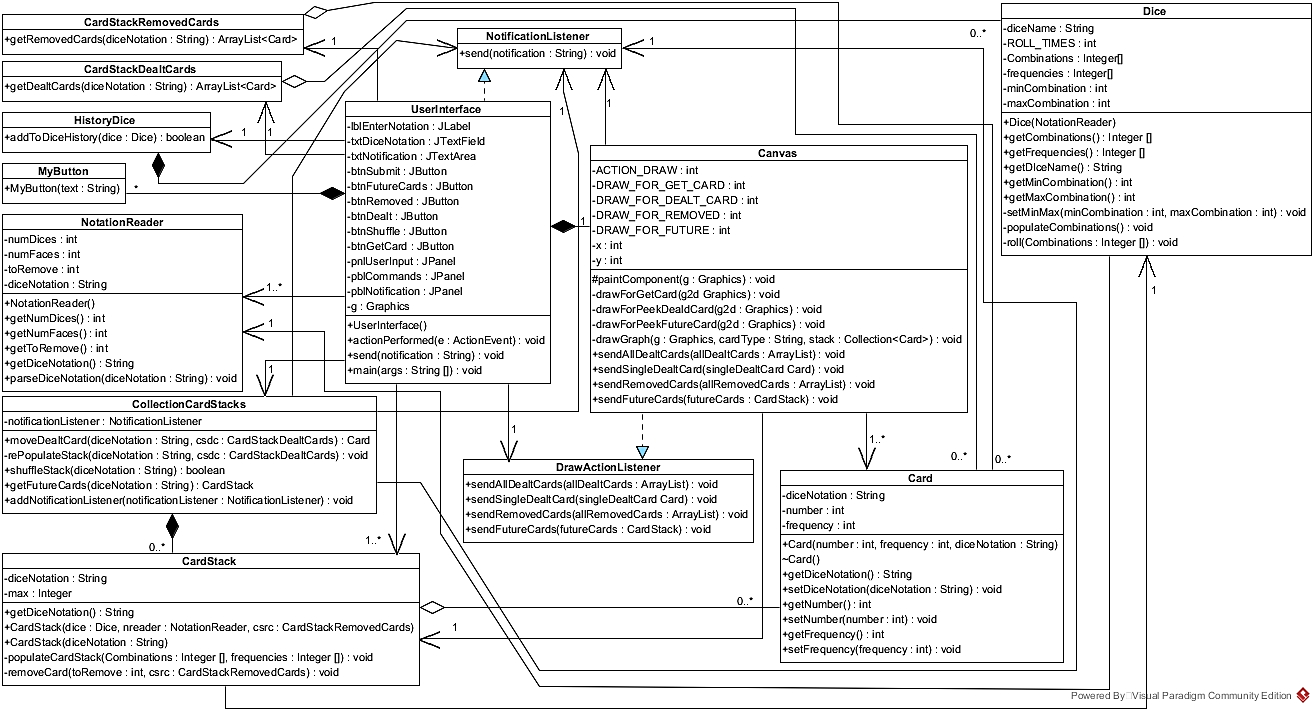


As shown in the above figure, graph of dealt cards were correctly generated.

### Test Summary

From the above test results, all tests were executed as expected. It is safe to say that the communication between objects has been validated. Integration testing was successfully accomplished without unexpected results.

# Task 3 – Class Diagram



Page **136** of **140**

# Conclusion

In a nutshell, the report has incorporated 3 aspects of application documentation - the source code of the application, the testing carried out full coverage of verification & validation of the application and finally the UML based class diagram to provide the design overview.

In case of technical implementations, the application has provided the ability to generate combinations from multi-dimensional dices, by simulating the virtual dice. The integrity of the dice notation was maintained by validating user input. Exceptions were handled where relevant in order to make the application robust enough against unexpected runtime scenarios. Cohesion was maintained by distributing workload over several dedicated classes. Tight coupling was loosened using interfaces in case of notification passing & canvas paintings for graph.

Similarly, in case of testing of application. Each methods of all classes were black-box and white-box tested using relevant equivalence partitioning for test data. The tests were distributed across test suites for common type of test & the method being tested. Lastly, integration testing was carried out to check proper functioning of communication between the objects.

The report is a documented evidence for demonstrating the requirements fulfillment as mandated by the board game company. The power of object-oriented programming style was harnessed to deliver a reliable solution.

Page **137** of **140**

# Referencing

Bell, D. & Parr, M., 2010. Exceptions. In: *Java for Students, 6th Ed..* London: Pearson Education Limited, pp. 301-314.

Deitel, P. & Deitel, H., 2015. Regular Expressions, Class Patterns and Class Matcher. In: *Java How To Program, 10th Ed..* New Jersey: Pearson, pp. 624-633.

Schildt, H., 2014. Painting in Swing. In: *Java The Complete Reference, 9th Edition.* New York: McGraw-Hill Education, pp. 1036-1040.

Vermeulen, A. et al., 2000. Documentation Conventions. In: *The Element of Java Style.* Cambridge: Cambridge University Press, pp. 31-52.

Page left intentionally blank.Page left intentionally blank.