Testing and Debugging

# Section 1. Issues found in Card.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| toString() method | When printing the value of the card the program prints the suit and vice versa. | Swapped the suit and value variables in the toString() method. |
| isSameSuit() method | When checking if the suit is the same, getValue() method is called instead of getSuit(). | Instead of calling getValue() method, changed it to getSuit() method. |
| hashCode() method | Wrong return value. | Changed the returning value of the hashCode() method from this.hashCode() to result. |
| hashCode() method | Wrong value of the prime variable. | Changed the variable prime value from 15 to 17. |
| equals() method | Wrong comparison operator in the first if statement. | Changed the comparison operator in the first if statement from != to ==. |

# Section 2. Issues found in Player.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| getHandAsString() method | The method only prints the first card of the player. | Changed the assignment for the variable output inside the for loop from output = c.toString() + "\n" to output = output + c.toString() + "\n" |
| hasWon() method | Wrong return value. | Changed the return value from !hand.isEmpty() to hand.isEmpty(). |
| isAI() method | Wrong return value | Changed the return value from false to ai. |

# Section 3. Issues found in Switch.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| dealCards() method | shuffleStock() method is being called before the genPack() method. | Changed the order in which the methods are called: firstly the genPack() method then the shuffleStock() method. |
| resetFlags() method | Wrong value of draw4 variable. | Changed the value of draw4 from true to false. |
| runPlayer() method | The pickUpCard() method is not being called enough times when variable draw4 is true. | Added two more pickUpCard() method calls. |
| runPlayer() method | Wrong second if statement. | Changed the second if statement from draw2 || draw4 to draw2. |
| discardCard() method | Wrong discarded variable value after discarding the card. | Changed the discarded variable value inside the if statement from false to true. |
| setFlags() method | Swapped variables in the KING and QUEEN cases. | Changed the variable in the case KING from draw4 to reverse and in the case QUEEN from reverse to draw4. |
| runGame() method | Unnecessary runRound() method call in the default case of the switch statement. | Commented out the runRound() method call in the default case of the switch statement. |

# Section 4. Issues found in UserInterface.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| getIntInput() method | When calling getIntInput() method inside the runGame() method, throws java.lang.NullPointerException, because no Scanner class object is being created. | The private Scanner type variable sc is initialised as a new Scanner object in the getIntInput() method. |
| getPlayerInformation() method | The first for cycle asking for the human player names is not being executed. | In the first for cycle changed the > to < (i < noOfPlayers instead of i > noOfPlayers). |
| getPlayerInformation() method | The name of the AI is not being printed. | Added a System.out.println() method to print the name of the AI player. |
| selectPlayer() method | Wrong argument when calling players.get() method inside the for loop. | Changed the argument when calling players.get() method inside the for loop from i + 1 to i. |
| printWinnerOfGame() method | The name of the winner is not being printed. | Added the name of the player as an argument when calling System.out.println() method. |

# Section 5. Issues found in Constants.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| FOR SWITCH field declarations. | Wrong HAND\_SIZE variable value. | Changed HAND\_SIZE value from 16 to 7 as written in original program specification. |

# Section 6. Testing and debugging description

Firstly, I have started testing the program by checking if it runs at all. The program did not run properly and the NullPointerException was being thrown. Looking at the exception description I have identified that the problem was in the getIntInput() method and no Scanner class object was being created. I have fixed this problem and also provided an alternative implementation of this method to handle any type of input from the user.

Then I have tried running the program again and provided various incorrect inputs to check if the user can try entering the values until the input is accepted by the program. After making sure that this works, I have moved on to trying to enter other inputs asked by the program (number of players). However, looking at the getPlayerInformation() method, the program should have asked the user to enter the names of the players. Unfortunately, the program was not asking to enter any names. It was quite easy to notice that the wrong relational operator was used in the for loop inside the getPlayerInformation() method and it was not being executed. After fixing this, I have noticed that the comment in the program asks for the name of the AI player as well, however printing the name of the AI was not implemented, so I have added System.out.println() method to fix this.

By running the program again, I have identified that the next problem was with shuffling the cards. The error message was saying that the source for the deck of cards was not found. It meant that there were no cards to shuffle. Firstly, the deck of cards had to be generated and only then shuffled, I have swapped the method call order of genPack() and shuffleStock() to fix this. After that, I have printed the values of the Card objects and noticed that value and suit should be swapped. To fix this I have swapped them in the toString() method in the Card class. I have also checked if shuffling works by printing the Card objects again. After shuffling, I have also printed the value of the top card in the getTopCard() method to see if it matches with the first card in the shuffled deck.

After running the program further, it was throwing java.lang.IndexOutOfBoundsException when cards were dealt between the players. To see where the mistake was, I have printed the values of the cards of each player and noticed that all of the cards in the stock were being distributed between the players and the for loop in the dealCards() method was being executed more times than necessary (checked by printing the value of i at the start of the loop) until there were no more cards to deal. To fix this, I have changed the value of HAND\_SIZE constant in the Constants class as the original program specification requires that each player should have 7 cards in his hand at the start of the game.

Then, I have continued to run the program and got to the point where it was printing the information about the player. I have noticed that the number of cards in the player’s hand was still higher than it should be. I have followed the program from top to bottom and noticed that the cause of this was the wrong value of draw4 variable in the resetFlags() method inside the Switch class. It was the reason why the player had 4 more cards. When investigating this mistake, I have also encountered another one related with drawing 4 cards. The method pickUpCard() was being called only 2 times instead of 4 when the player had to draw 4 cards.

After this, I have noticed that the program was asking the user to take actions for AI players as well. AI players should not ask the user to take any actions for them. To fix this I have changed the return value in the isAI() method in the Player class.

I have tried running the program again and after making sure that the player has the correct number of cards, I have tried to discard one of the cards in the hand that had the same suit as the top card. However, the program did not allow me to do so. Again, I have tried following the program from top to bottom and found out that the mistake was in the isSameSuit() method in the Card class. It was calling getValue() method instead of getSuit(). After fixing this mistake, it was clearly visible that when the correct card was discarded the message in the program said that it was unable to discard the card. By following the program’s code I found that the problem was in the discardCard() method inside the Switch class and the discarded variable value after discarding the card was set to false instead of true.

The next problem was that the program discarded not the card that was selected but the first one in the player’s hand. After following the program step by step I still could not find the mistake. I have done a bit of research about why this might be happening on the internet and found out that the problem was the wrong comparison operator in the equals() method inside the Card class. While fixing this mistake I have also realised that the return value in the hashCode() method was wrong as well, so I fixed it. Furthermore, the prime variable value was not a prime number, so I have changed it from 15 to 17.

After fixing the previous problem, I have tried selecting the option where the program only prints the hand of the player. However, the program was printing only the first card in the hand. I have searched through the program and encountered the problem in the getHandAsString() method in the Player class. The variable output was being assigned a wrong value. I have changed it so that it would print all of the cards in the player’s hand.

After testing the program even further and printing out the size of the hand after each player’s turn, it was visible that the function of the card with value KING was wrong as it added more cards to the other player’s hand. I have noticed that in the setFlags() method variables in the KING and QUEEN cases were swapped, so I changed them according to the program’s specification. I have also noticed that when QUEEN card was played the next player had to pick up more than 4 cards. I have missed this mistake in runPlayer() method when I was fixing the number of pickUpCard() method calls, so I fixed it now (changed the second if statement from draw2 || draw4 to draw2), when investigating the KING and QUEEN cases.

Then I have tried inspecting if the special cards do what they are supposed to. I was running the program multiple times to get the needed cards and while testing the functions of these cards I have noticed that when the card with the JACK value was played the program was throwing IndexOutOfBound exception. To fix this, I have investigated the line where the exception occurred and found out that the argument was wrong when calling the get() method in the selectPlayer() method.

Finally, I have tested the program until there was a winner. However, when one of the players discarded all of his cards the game continued. The problem was with the hasWon() method in the Player class. It was returning the wrong value. After fixing this, I have noticed that the name of the winner was not being printed by the program. It was not difficult to find that the mistake was inside the printWinnerOfGame() method, so I have added the name of the player as an argument when calling System.out.println() method.

After making sure that the game runs correctly, I tried to figure out why the goodbye message was not being printed after selecting exit. It was quite easy to find that the cause of this was calling the runRound() method if the exit option was selected, so I have commented out this line of code.

One more problem that I found but not included in the issues section is handling the case when none of the players discard any cards and only pick them up from the stock. In this case the program was throwing OutOfBounds exception when the stock was empty and there were no cards to pick from in the discard pile. I have added some comments and my way of dealing with this situation. Also, it might be a good idea to implement a way for AI to automatically choose the player with the least cards when the card with the value of JACK is played and swap the cards with this player.

Section 7. Conclusion

Overall, I believe that the testing and debugging of this card game program was successful. I have gained valuable experience on finding and fixing mistakes in a code that is not written by me. My method of testing and debugging mostly consisted of running the program multiple times, seeing the results, printing lines of code to the console and fixing each mistake one by one. I have not used Eclipse tools to test this program as I believe that I still need more experience working with them. However, I think that if I would have made use of these tools, the debugging process would have been a bit easier. I will try to get more familiar with the available tools that might help to test the code and use them in the future.