|  |  |
| --- | --- |
| **Application/ Program name:** | PickTwoCards and Card |
| **Written by:** | Zachary Muerle |

|  |
| --- |
| **Purpose or problem definition:** |
| “Playing cards are used in many computer games, including versions of such classics  as solitaire, hearts, and poker. Design a Card class that contains a character data field  to hold a suit (s for spades, h for hearts, d for diamonds, or c for clubs) and an integer  data field for a value from 1 to 13. (When you learn more about string handling in  the chapter Characters, Strings, and the StringBuilder, you can modify the class to  hold words for the suits, such as spades or hearts, as well as words for some of the  values—for example, ace or king.) Include get and set methods for each field. Save  the class as Card.java.  Write an application that randomly selects two playing cards and displays their  values. Simply assign a suit to each of the cards, but generate a random number  for each card’s value. Save the application as PickTwoCards.java” |
|  |
| **Program Procedures:** |
| Card creates a data type of type Card that holds a char and an int. Inputs are sanitized, so as to make sure the class behavior is predictable. PickTwoCards creates a list of 4 characters, used to choose a random suit for the card, creates 2 null Cards, then generates 2 random numbers twice: one from 0 to 4 for the suit, the other 1-13 for the value of the card. These values are then used to create 2 new cards at random. The program then asks the Card class to put the cards into a descriptive string, and it packs those into a message dialog to show the user. |
|  |
| **Algorithm/Processing/Conditions:** |
| **Inputs: none** |
|  |
| **Processes: toString()** |
| **return "The " + this.getValueName() + " of " + this.getSuitName(); where:**  **public** String getValueName(){//translates the value into their names, if applicable  **if**(**this**.value==1){  **return** "Ace";  }  **else** **if**(**this**.value==11){  **return** "Jack";  }  **else** **if**(**this**.value==12){  **return** "Queen";  }  **else** **if**(**this**.value==13){  **return** "King";  }  **else**{  **return** Integer.*toString*(**this**.value);//send back the value as a string if it's not a special value  }  }  And:  **public** String getSuitName(){//returns the full names, if you just want the char, the previous method does that  **if**(**this**.suit == 's'){  **return** "Spades";  }  **else** **if**(**this**.suit == 'c'){  **return** "Clubs";  }  **else** **if**(**this**.suit == 'h'){  **return** "Hearts";  }  **else** **if**(**this**.suit == 'd'){  **return** "Diamonds";  }  **else**{  **throw** **new** IndexOutOfBoundsException("The card's suit wasn't s,c,h, or d. Somehow.");  //if it gets down here, the card was corrupted.  }  } |
| **Outputs: Message** |
| "I picked 2 cards: \n" + cardOne.toString() + "\nand \n" + cardTwo.toString() |
|  |
| **Notes & Restriction:** |
| Card only supports the 4 normal suits: Spades, Clubs, Hearts, and Diamonds. To input a suit, only lowercase letters are allowed, S,C,H, or D will error, just as if it were any character besides s,c,h, and d. Card also only allows values 1-13, and assumes 1 = ace, 11=jack, 12=queen, and 13=king |
|  |
| **Comments:** |
| The for loop isn’t really needed, but I decided to use it anyways, to re-use the random number code. It simplifies debugging if there is less places for the code to break, and if it does not break one time, it should not break any time. |