Sortable Poker Hand

WEEKLY DEMO

ARNOLD BARNA

2017-09-15

Sortable Poker Hand

French card: Described by rank and suit

- Ranks: 2, 3, 4, 5, 6, 7, 8, 9, T(en), J(ack), Q(ueen), K(ing), A(ce)
- Suits: S(pades), H(earts), D(iamonds), C(lubs)

Hand: Five valid cards you have in your hand

Valid inputs

Valid card: „<Rank><Suit>”

- Acceptable: *“2H” “td”*
- Not acceptable: *“c9”* “c99” “” etc

Valid hand: ”<card#1> <card#2> … <card#5>”

- Acceptable: *“KS 2H 5C JD TD”*
- Not acceptable: *„KS 2H 5C JD” „KS2H5CJDTD” „KS,2H,5C,JD:TD” “KS KS 5C JD TD”*

How do I know it works?

```
CardValidatorTest("2H", true, Ranks, Suits);
CardValidatorTest("td", true, Ranks, Suits);
CardValidatorTest("c9", false, Ranks, Suits);
CardValidatorTest("c99", false, Ranks, Suits);
CardValidatorTest("", false, Ranks, Suits);
```

Test the card validator

```
public static void CardValidatorTest(string sampleCard, bool expectedResult, char[] Ranks, char[] Suits)
{
    bool actualResult = PokerHand.CardIsValid(sampleCard, Ranks, Suits);
    Console.WriteLine("Testing card validator, input string: {0}, expected to be {1}, result: {2}, succeded: {3}",
        sampleCard,
        expectedResult,
        actualResult,
        actualResult == expectedResult);
}
```

```
Testing card validator, input string: 2H, expected to be True, result: True, succeded: True
Testing card validator, input string: td, expected to be True, result: True, succeded: True
Testing card validator, input string: c9, expected to be False, result: False, succeded: True
Testing card validator, input string: c99, expected to be False, result: False, succeded: True
Testing card validator, input string: , expected to be False, result: False, succeded: True
```

Hand validator

```
HandValidatorTest("KS 2H 5C JD TD", true, Ranks, Suits, validNumberOfCards, Separator);
HandValidatorTest("KS 2H 5C JD", false, Ranks, Suits, validNumberOfCards, Separator);
HandValidatorTest("KS2H5CJDTD", false, Ranks, Suits, validNumberOfCards, Separator);
HandValidatorTest("KS,2H,5C,JD:TD", false, Ranks, Suits, validNumberOfCards, Separator);
HandValidatorTest("KS KS 5C JD TD", false, Ranks, Suits, validNumberOfCards, Separator);
```

```
Testing hand validator, input string: KS 2H 5C JD TD, expected to be True, result: True, succeded: True
Testing hand validator, input string: KS 2H 5C JD, expected to be False, result: False, succeded: True
Testing hand validator, input string: KS2H5CJDTD, expected to be False, result: False, succeded: True
Testing hand validator, input string: KS,2H,5C,JD:TD, expected to be False, result: False, succeded: True
Testing hand validator, input string: KS KS 5C JD TD, expected to be False, result: False, succeded: True
```

Sorting validator

```
SortingTest("KS 2H", "KS 2H", Ranks, Suits, 2, Separator);
SortingTest("2H KS", "KS 2H", Ranks, Suits, 2, Separator);
SortingTest("KS 2H 5C", "KS 5C 2H", Ranks, Suits, 3, Separator);
SortingTest("AS AH AD AC", "AS AH AD AC", Ranks, Suits, 4, Separator);
```

```
Testing hand sorting, input string: KS 2H, expected to be KS 2H, result: KS 2H, succeded: True
Testing hand sorting, input string: 2H KS, expected to be KS 2H, result: KS 2H, succeded: True
Testing hand sorting, input string: KS 2H 5C, expected to be KS 5C 2H, result: KS 5C 2H, succeded: True
Testing hand sorting, input string: AS AH AD AC, expected to be AS AH AD AC, result: AS AH AD AC, succeded: True
```

How to validate the hand?

```
public static bool IsValid(string handString, char[] Ranks, char[] Suits, int validNumberOfCards, char Separator)
{
    HashSet<string> cardsToCheckSet = new HashSet<string>(handString.ToUpper().Split(Separator)); // remove duplicates

    if (cardsToCheckSet.Count != validNumberOfCards)
    {
        return false;
    }

    foreach (string card in cardsToCheckSet)
    {
        if (!CardIsValid(card, Ranks, Suits))
        {
            return false;
        }
    }
    return true;
}
```

How to validate the card? Long way …

```
public static bool CardIsValid_VeryLongMethod(string card, char[] Ranks, char[] Suits)
{
    card = card.ToUpper();
    if (card.Length != 2)
    {
        return false;
    }
    if (!Ranks.Contains(card[0]))
    {
        return false;
    }
    if (!Suits.Contains(card[1]))
    {
        return false;
    }
    return true;
}
```

How to validate the card? Shorter way …

```
public static bool CardIsValid_LongMethod(string card, char[] Ranks, char[] Suits)
{
    card = card.ToUpper();
    if (card.Length != 2 || !Ranks.Contains(card[0]) || !Suits.Contains(card[1]))
    {
        return false;
    }
    return true;
}
```

How to validate the card? Even shorter …

```
public static bool CardIsValid(string card, char[] Ranks, char[] Suits)
{
    card = card.ToUpper();
    return !((card.Length != 2 || !Ranks.Contains(card[0]) || !Suits.Contains(card[1])));
}
```

How to validate the card? Black belt …

```
public static bool CardIsValid_WithLambda(string card, char[] Ranks, char[] Suits) =>
    !((card.Length != 2 || !Ranks.Contains(card.ToUpper()[0]) || !Suits.Contains(card.ToUpper()[1])));
```

How to sort the hand recursively?

```
public void Sort(char[] Ranks, char[] Suits, int validNumberOfCards, int outerCounter = Int32.MaxValue)
{
    outerCounter = outerCounter == Int32.MaxValue ? validNumberOfCards - 1 : outerCounter;

    if (outerCounter <= 0)
    {
        return;
    }

    for (int innerCounter = 0; innerCounter < outerCounter; innerCounter++)
    {
        if (FirstIsSmaller(Hand[innerCounter], Hand[innerCounter + 1], Ranks, Suits))
        {
            SwapCardWithNext(innerCounter);
        }
    }
    Sort(Ranks, Suits, outerCounter - 1);
}
```

How to compare two cards?

```
public static bool FirstIsSmaller(string card1, string card2, char[] Ranks, char[] Suits)
{
    if (Array.IndexOf(Ranks, card1[0]) < Array.IndexOf(Ranks, card2[0]))
    {
        return true;
    }
    if (Array.IndexOf(Ranks, card1[0]) > Array.IndexOf(Ranks, card2[0]))
    {
        return false;
    }
    return (Array.IndexOf(Suits, card1[1]) < Array.IndexOf(Suits, card2[1]));
}
```

Thank you for the attention

Available on GitHub:

Code:

- https://github.com/greenfox-academy/**bramble100**/tree/master/week-02/**SortablePokerHands**

Slides:

- https://github.com/greenfox-academy/**bramble100**/tree/master/week-02/**demo**