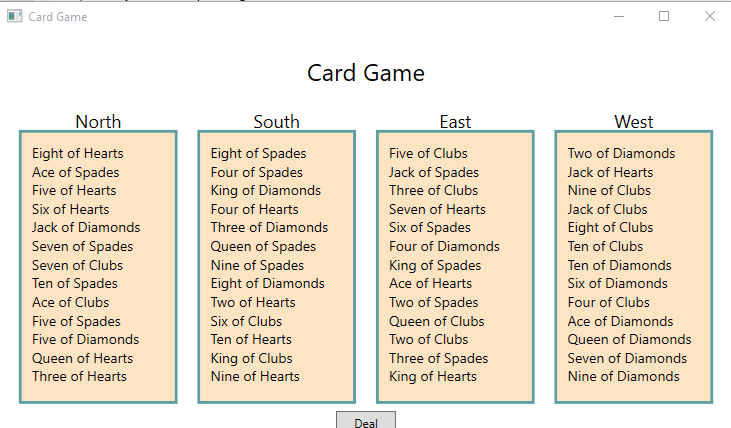
Student: Brian Johnston

Class COP2362

Instructor: Hamilton

Assignment 4-4

Print screen



Code:

Hand.cs

using System;

using System.Collections.Generic;

namespace Cards

{

class Hand

{

public const int HandSize = 13;

private List<PlayingCard> cards = new List<PlayingCard>(HandSize);

public void AddCardToHand(PlayingCard cardDealt)

{

if (this.cards.Count >= HandSize)

{

throw new ArgumentException("Too many cards");

}

this.cards.Add(cardDealt);

}

public override string ToString()

{

string result = "";

foreach (PlayingCard card in this.cards)

{

result += card.ToString() + "\n";

}

return result;

}

}

}

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Cards

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public const int NumHands = 4;

private Pack pack = null;

private Hand[] hands = { new Hand(), new Hand(), new Hand(), new Hand() };

public MainWindow()

{

InitializeComponent();

}

private void dealClick(object sender, RoutedEventArgs e)

{

try

{

pack = new Pack();

for (int handNum = 0; handNum < NumHands; handNum++)

{

hands[handNum] = new Hand();

for (int numCards = 0; numCards < Hand.HandSize; numCards++)

{

PlayingCard cardDealt = pack.DealCardFromPack();

hands[handNum].AddCardToHand(cardDealt);

}

}

north.Text = hands[0].ToString();

south.Text = hands[1].ToString();

east.Text = hands[2].ToString();

west.Text = hands[3].ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

Pack.cs

using System;

using System.Collections.Generic;

namespace Cards

{

class Pack

{

public const int NumSuits = 4;

public const int CardsPerSuit = 13;

private Dictionary<Suit, List<PlayingCard>> cardPack;

private Random randomCardSelector = new Random();

public Pack()

{

this.cardPack = new Dictionary<Suit, List<PlayingCard>>(NumSuits);

for (Suit suit = Suit.Clubs; suit <= Suit.Spades; suit++)

{

List<PlayingCard> cardsInSuit = new List<PlayingCard>(CardsPerSuit);

for (Value value = Value.Two; value <= Value.Ace; value++)

{

cardsInSuit.Add(new PlayingCard(suit, value));

}

this.cardPack.Add(suit, cardsInSuit);

}

}

public PlayingCard DealCardFromPack()

{

Suit suit = (Suit)randomCardSelector.Next(NumSuits);

while (this.IsSuitEmpty(suit))

{

suit = (Suit)randomCardSelector.Next(NumSuits);

}

Value value = (Value)randomCardSelector.Next(CardsPerSuit);

while (this.IsCardAlreadyDealt(suit, value))

{

value = (Value)randomCardSelector.Next(CardsPerSuit);

}

List<PlayingCard> cardsInSuit = this.cardPack[suit];

PlayingCard card = cardsInSuit.Find(c => c.CardValue == value);

cardsInSuit.Remove(card);

return card;

}

private bool IsSuitEmpty(Suit suit)

{

bool result = true;

for (Value value = Value.Two; value <= Value.Ace; value++)

{

if (!IsCardAlreadyDealt(suit, value))

{

result = false;

break;

}

}

return result;

}

private bool IsCardAlreadyDealt(Suit suit, Value value)

{

List<PlayingCard> cardsInSuit = this.cardPack[suit];

return (!cardsInSuit.Exists(c => c.CardSuit == suit && c.CardValue == value));

}

}

}

PlayingCard.cs

namespace Cards

{

class PlayingCard

{

private readonly Suit suit;

private readonly Value value;

public PlayingCard(Suit s, Value v)

{

this.suit = s;

this.value = v;

}

public override string ToString()

{

string result = string.Format("{0} of {1}", this.value, this.suit);

return result;

}

public Suit CardSuit

{

get

{

return this.suit;

}

}

public Value CardValue

{

get

{

return this.value;

}

}

}

}

Suit.cs

namespace Cards

{

enum Suit { Clubs, Diamonds, Hearts, Spades }

}

Value.cs

namespace Cards

{

enum Value { Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King, Ace }

}