Card Games Documentation

Andrada Iorgulescu and March 6, 2022

Delete the explanatory text (in italics) and only include your answers.

Incorporate code excerpts, screenshots, and other media as needed to complete this assignment. Expectations for the length of response are included, but quality is more important than quantity. Please do not add filler text to meet the word amount expectation.

Framework Overview

Overview

One of the most important parts of this assignment was creating a common framework for components that shared commonalities.

Describe your framework. What are the commonalities that you abstracted out, and what was the rationale for the decisions that you made? *(About 500-700 words)*

If you used the default assignment, **exclude the abstraction prompts provided***. Write about the abstractions you made in addition to what was already suggested.*

System(s)

The default project was to create a system. It wasn't described in that way in each individual assignment, but after completing the individual assignments you should have created an application that could be described as a system according to the definition from the reading "Thinking in Systems":

> **A system is an interconnected set of elements that is coherently organized in a way that achieves something.**
>
> *A system must consist of three kinds of things:* **elements, interconnections, and a function or purpose***.*

Describe the elements, inputs, outputs, and results in your framework. *You may want to add a diagram to show how your system works. (About 500-700 words)*

Object-Oriented Programming

In creating a work of this scope in C# you were also challenged with exploring and experimenting with fundamental object-oriented design principles (polymorphism, inheritance,

and encapsulation). **Put your results into context; define the terms and explain how they were used in your project.**

Polymorphism

Definition:

Brief code excerpt(s) from your project:

Explain usage in your project:

Inheritance ("is a")

Definition: a class that inherits attributes/behaviors from another.

Brief code excerpt(s) from your project:

```
namespace MakingADeckClass
{
    public class Deck : Utility
    {
```

Explain usage in your project:

Encapsulation

Definition: A group of members that act as a single unit/object. Object hides information that is not necessary to the user.

Brief code excerpt(s) from your project:

```csharp
public void PrintCard()
{
    if (Value == 10)
    {
        Console.WriteLine($"Jack of {Suit}");
    }
    else
    {
        if (Value == 11)
        {
            Console.WriteLine($"Queen of {Suit}");
        }
        else
        {
            if (Value == 12)
            {
                Console.WriteLine($"King of {Suit}");
            }
            else
            {
                if (Value == 13)
                {
                    Console.WriteLine($"Ace of {Suit}");
                }
                else
                {
                    Console.WriteLine($"{Value} of {Suit}");
                }
            }
        }
    }
}
```
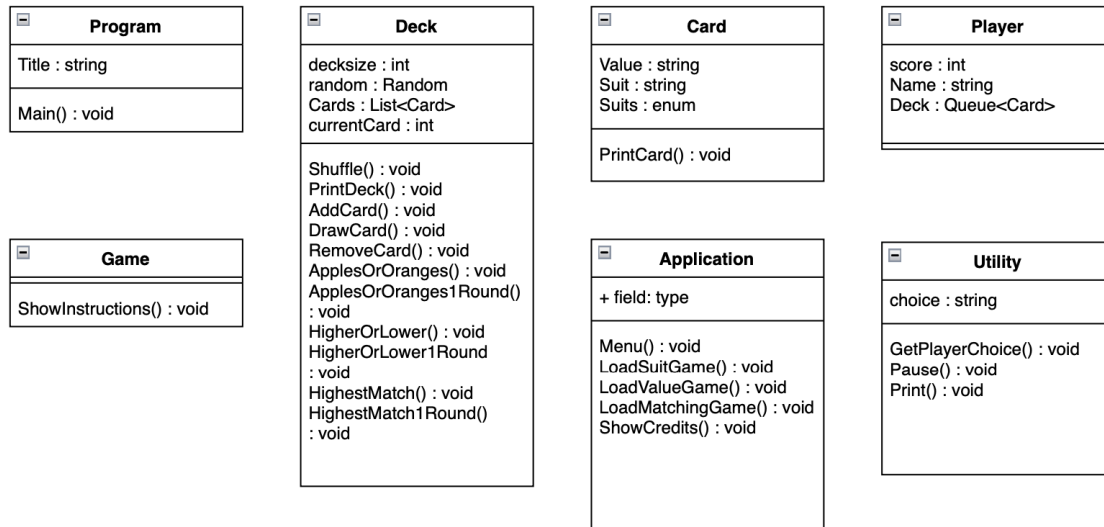
Explain usage in your project:

Experimentation and Questions

*Part of this assignment was to experiment as you problem-solved and thought critically about the way the application could be developed. The questions and experimentation (the process of learning how to create this type of application) is essential. **Exclude the question prompts provided**. Write about the questions you had during development that were not suggested.*

Describe your process. What were the questions that arose as you worked through the process? What did you try that worked? What did you try that failed? What did you learn and find most valuable about the process? *(About 1,000-1,500 words)*

UML Diagrams

Program
Title : string
Main() : void

Deck
decksize : int random : Random Cards : List<Card> currentCard : int
Shuffle() : void PrintDeck() : void AddCard() : void DrawCard() : void RemoveCard() : void ApplesOrOranges() : void ApplesOrOranges1Round() : void HigherOrLower() : void HigherOrLower1Round : void HighestMatch() : void HighestMatch1Round() : void

Card
Value : string Suit : string Suits : enum
PrintCard() : void

Player
score : int Name : string Deck : Queue<Card>

Game
ShowInstructions() : void

Application
+ field: type
Menu() : void LoadSuitGame() : void LoadValueGame() : void LoadMatchingGame() : void ShowCredits() : void

Utility
choice : string
GetPlayerChoice() : void Pause() : void Print() : void

UML Diagram Explanation

Describe your structure and explain your design decisions (i.e., the rationale for your decisions). Include information about inheritance, relationships between classes, and the types of structures that you chose.

Playtesting

Document at least one playtesting session (you can use the in-class playtest session information).

Test date: February 21, 2022

Names of testers:

1) Herbert Chillis

2)

What were the three most successful aspects of your project?

- *The game has a title and a menu to see what games you want to play.*
- *The game modes work when playing them and give you a point when you guess right.*
- *It shows the cards and decks when being printed out.*

What were the three biggest issues with your project?

- *The game modes don't have more than one turn. When you guess correctly or wrong the game takes you back to the menu.*
- *Highest match does not let you guess which is the highest match of the card. //incomplete*
- *Maybe have the player type in a number that they can choose what game mode to play then typing the name of it out.*

How will you use what you learned to improve your application for the next version?

- I would like to add some kind of text art to make my project look a little more intentional
- I would also like to fix/complete the games that didn't fully run
- I'd also like to give the player the option to continue the game endlessly or go back to the menu

Credits

- Shuffle code adapted from https://stackoverflow.com/questions/273313/randomize-a-listt

- Card class was built with tutoring help from Karen Spriggs (tutor).

- PrintDeck and AddCard methods in Deck class were also built with help from Karen Spriggs (tutor).