Assignment 3 Report

# Name

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Result |
| hello | “Hello hello! I want to play a game of Lincoln with you 😊” |  |
| 12 | The game will refer to me as 12 |  |

# Reading Rules

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Result |
| Yes | The rules are not displayed |  |
| no | Rules displayed |  |
| 12 | Repeat of the question |  |

# Number of Rounds

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Result |
| 12 | Game starts |  |
| two | Error |  |

# Input the first card

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Result |
| 1 | Asks for next card |  |
| Two | Asks for an integer to be put in |  |

# Input the second card

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Result |
| Same card as before | Loops as card was input last time |  |
| 2 | Plays the card |  |
| Three | Loops for a response |  |

# Pillars of OOP in my code

Encapsulation – Encapsulation through the design of classes is present throughout my code, and 7 classes excluding the program class exist, albeit it 2 of them inherited from an interface class. All classes bar the program class come with attributes (except the interface) and methods.

Abstraction – All classes make use of private/public access modifiers, with functions and attributes unique to these classes used and worked with. The inherited Player Classes make use of private attributes, but do not make use of private methods as these methods call other public methods from Hand. This is done so that the response from the method calls from the Hand class remain specific to the players.

Polymorphism – in the PlayLincoln class the NumParse() function takes 2 forms and uses static polymorphism to do this. One version of NumParse takes no arguments and this is the function that gets called first, as this lets the player play the first card. Then there is another version of the NumParse function that takes in an integer argument, lets the player pick their card as normal, then checks if this integer and the card picked match. This is because the integer put through is the result of the first NumParse function. An added bonus with this is that it was possible to have the same function output two different responses based on context.

Inheritance – I use an interface called IPlayer, and then inherit Human and Computer from those. I decided to use an interface because while both the computer and human players needed similar methods, how these get done, and what attributes these classes had, were different from one another. Therefore using an interface, where I could only use abstract classes to override, was the easiest and most efficient option. The Human and Computer class are declared in the same file underneath the IPlayer interface.

# Checklist







