ECS769P – Advanced Object-Oriented Programming

Coursework 1

Dr. Ling Ma & Dr. Julian Hough

A Card Game – “Pontoon”

Johannes Hartmann
Queen Marry University London
MSc Software Engineering
170904935

Introduction

The application is a console based Black Jack (Pontoon) game using C++ as programming language. The game is played between a computer player and a human player.

How to compile

First make sure you are in the root folder of this game. To compile the code just run the following command in a terminal. It deletes previously existing code and compiles the project:

- make clean && make

The rules how to compile the code are defined in the Make file.

Note: Make sure the latest gcc compiler version is installed to compile the code.

This was tested on an Ubuntu Linux system. For windows systems make sure that make files can executed.

How to execute

To start the game, simply execute the following command in a terminal at the project root folder:

- ./build/main

How to play

When you start the game, you get dealed two cards automatically and your hand will be printed to the console. Then you will be asked to stick or twist, which means that you can have another card or stick with your cards. After you finish your round by choosing to stick, it's the banks turn to play.

After the banks turn, the result of the game will be printed to the console and you will be asked if you want to play another round.

Programming Approach

The application is implemented using the Model View Controller approach. Furthermore, the cards of the game are implemented as a card library, enabling other programmers to reuse the card library to implement other games.

The “controller” in the application is implementing the game rules, using the cards library as model.

The “view” is responsible for all IO actions. It is divided into two classes: View, FileLogger. The view class is responsible for all IO actions at the console and the FileLogger class is responsible for creating and writing the log file of a game.

Cards Library

The structure of the card game library is the following: First, a card consists of two card properties representing the face and the suite of a card. These properties are given to the card as template parameters. The card ensures with static_assert that both template parameters are subtypes of the abstract class CardProperty. By doing so it is ensured that the face and the suite of a card implement some important methods which are used to generically create a deck.

A deck is a container which is holding all cards used for the game. It is also a template class parameterised by a face and a suite. It can automatically setup the deck, by using the cards incrementFace and incrementSuite functions. This setupDeck function may also be overridden in a potential subclass to implement a different behaviour. Furthermore, it also supports to shuffle the deck. This is done by a shuffle algorithm defined by the abstract class ShuffleAlgorithm. This algorithm can be given as a parameter to the shuffle function of a deck. There is also a default shuffle algorithm implementation which is used if no algorithm is specified as a parameter. This default shuffle algorithm iterates over the deck 100 times and randomly swaps two cards in the deck. Finally, you can see the top card of a deck and pop this top card.

The last part of the library is the hand of a player. It is again a template class parameterised by a face and a suite. It has also a maximum size specified in the constructor. You can add cards to the hand until the handSize is reached. The hand is also inerrable to get specific cards out of the hand. It also provides a getValue function which returns the current value of all cards in the hand. This method may also be overridden to implement a different behaviour.

To use this card library to implement a game, a potential user must implement his own face and suite classes and instantiates a deck and an arbitrary number of hands with this classes. A potential user also may define his own deck and hand classes to implement different behaviours. The library also provides a face and suite class for a normal French card set.