University of Westminster
Department of Computer Science

5CCGD012W	Game Programming Patterns – Coursework (2022/23)
Module leader	Klaus Draeger
Unit	Coursework
Weighting:	50%
Qualifying mark	30%
Description	Object Oriented Programming and Design.
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO1 Identify and justify good practices in the development of object oriented software; - LO2 Apply acquired knowledge of concepts, characteristics, tools and environments to adapt to new computational environments and programming languages which are based on object oriented principles; - LO3 Design, implement efficiently applications based on a OOP language, given a set of functional requirements.
Handed Out:	October 2022
Due Date	9th January 2023
Expected deliverables	**A zip file containing the developed project**
Method of Submission:	Electronic submission on Blackboard via a provided link close to the submission time.
Type of Feedback and Due Date: BCS CRITERIA MEETING IN THIS ASSIGNMENT	Written feedback within 15 working days. **2.1.1 Knowledge and understanding of facts, concepts, principles & theories** **2.1.2 Use of such knowledge in modelling and design** **2.1.3 Problem solving strategies** **2.2.1 Specify, design or construct computer-based systems** **2.2.4 Deploy tools effectively** **2.3.2 Development of general transferable skills** **3.1.1 Deploy systems to meet business goals** **4.1.1 Knowledge and understanding of scientific and engineering principles** **4.1.3 Knowledge and understanding of computational modelling**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website: **http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Objective:

Your mission in this assignment is to write a simple card game. The players take turns playing cards, trying to reduce each other's health to 0.

Overview of the game:

The game is played by two players, one human and one controlled by the computer (it is OK to have the computer player play randomly, but feel free to go with something more sophisticated).

Each player starts the game with 20 health and all their cards in their deck. On a player's turn, they

- Draw a card from the deck, adding it to their hand;
- Play any number of cards from their hand, applying their effect and then adding them to their discard pile. They need to play enough cards to reduce the size of their hand to 5 or less.

A player loses the game if

- Their health is 0 or less, or
- They are supposed to draw a card but cannot because their deck is empty.

Each player's deck initially contains the following cards:

- "Painful lesson" – opponent loses 2 health and draws a card. The deck contains 5 of these.
- "Spite" – both players lose 1 health. The deck contains 6 of these.
- "Full heal" – restores your health back to 20. The deck contains 1 of these.
- "Switcheroo" – swaps the contents of your hand with your opponent's (note that the switcheroo card being played is no longer in your hand at this time, so the opponent does not get it). The deck contains 2 of these.
- "Refresh" – lose 3 health and shuffle your discard pile back into your deck. The deck contains 2 of these.
- "Peek" – look at the top cards of both players' decks. The deck contains 4 of these.

Class Structure

You should implement the following principal classes (you might need more!):

1. An abstract **Player** class with subclasses HumanPlayer and ComputerPlayer.
2. An abstract **Card** class with subclasses for each of the types of cards above.

They should have several required members, listed below. You will also need suitable constructors and possibly getter methods for some attributes.

Player class (worth 20 marks)

The Player class contain at least the following data:

1. **opponent,** a pointer to the other player
2. **health,** representing their current health
3. vectors **deck, hand, discard_pile** each containing pointers to cards

It should have at least the following methods:

1. **void loseHealth(int i)** to lose the given amount of health
2. **bool drawCard()** to try to add the top card from the deck to the hand. Returns false if the deck is empty, true otherwise.
3. **void playCard(Card *c)** to play a card by removing it from the hand, applying its effect, and moving it to the discard pile.
4. **void myTurn()** to play a game turn. This should be a **pure virtual** method and overridden in both subclasses.
5. **bool hasLost()** to check if the player has lost (because at any point their health was 0 or less, or drawCard() returned false)

HumanPlayer Subclass (worth 10 marks)

The HumanPlayer subclass should have at least implement:

1. **void myTurn()** overriding the base class version. The player can choose to play any cards in their hand or end their turn (if they have no more than 5 cards left). After each card, check if either player has lost and end the game with a suitable message if they have.

ComputerPlayer Subclass (worth 10 marks)

The ComputerPlayer subclass should have at least implement:

1. **void myTurn()** overriding the base class version, either choosing cards randomly or using some smarter strategy.

Card Class (worth 10 marks)

The Card class should have at least the following **pure virtual** methods, to be overridden in each subclass:

1. **string getName()** for getting the card's name.
2. **void effect (Player *p)** for implementing a card's effect, to be applied to the given player and/or their opponent.

Card Subclasses (worth 5 marks each)

Each type of card listed in the game description should have its own dedicated subclass. They should override the virtual methods in the base class to represent their respective names and effects.

Required functions

In addition to the classes, you need to implement at least the following functions:

1. **void shuffle(vector<Card*> &cards) (worth 5 points)** to randomise the order of the cards in a deck, both for the initial setup and as part of playing a Refresh card.

2. The **main** function **(worth 15 points)** to do the initial setup (create players, shuffle their decks) and run the main loop (alternate player turns until one loses).

Marking criteria

Upload in BB your zipped source code. The assignment will be graded based on the following criteria:

- Implementation of the classes and methods defined in the coursework specification.
- Robustness of the code (error management)
- Readability of the code and presence of comments.

Coursework marking scheme:

Criterion and range	**Indicative mark**	**Comments**
Player class	16-20	Class has all the required members, and is well written and commented
	7-15	Class is lacking some members OR is not well written and commented
	0-6	Not done, or class is lacking members AND is not well written and commented
HumanPlayer subclass	8-10	Class has all the required members, and is well written and commented
	4-7	Class is lacking some members OR is not well written and commented
	0-3	Not done, or class is lacking members AND is not well written and commented
ComputerPlayer subclass	8-10	Class has all the required members, and is well written and commented
	4-7	Class is lacking some members OR is not well written and commented
	0-3	Not done, or class is lacking members AND is not well written and commented
Card class	8-10	Class has all the required members, and is well written and commented
	4-7	Class is lacking some members OR is not well written and commented
	0-3	Not done, or class is lacking members AND is not well written and commented
Card subclasses (per class)	4-5	Class has all the required members, and is well written and commented

	2-3	Class is lacking some members OR is not well written and commented
	0-1	Not done, or class is lacking members AND is not well written and commented
Shuffle function	4-5	Works by randomly permuting any given vector
	2-3	Limited or buggy
	0-1	Not done, or does not work at all
Main function	11-15	Sets up the data and executes the output/input/execute loop.
	2-10	Implemented, but has bugs or is missing features
	0-1	Not done, or does not work at all