COMP2012

Object-Oriented Programming and Data Structures

INTRODUCTION

source: ubisoft.com

In this assignment we are going to implement the game UNO and practice on the following topics:

- operator overloading
- polymorphism
- public inheritance
- private inheritance
- friend
- RTTI
- Abstract Base Class
- enum class (See tutorial here)

HONOR CODE

We value academic integrity very highly. Please read the Honor Code section on our course webpage to make sure you understand what is considered as plagiarism and what the penalties are. The following are some of the highlights:

- Do NOT try your "luck" - we use some sophisticated plagiarism detection software to find cheaters. It is much better than most students think. It has been proven times and times again tricks didn't work. We also review codes for potential cases manually.
- The penalty (for **BOTH** the copier and the copiee) is not just getting a zero in your assignment - it is much more than that. It is simply not worth to cheat at all. You would hurt your friend and yourself by doing that. It is obvious that a real friend won't ask you to get involved in a plagiarism act in any way due to the consequences. Read the Honor Code again before you even try to think about cheating.
- Serious offenders will fail the course immediately and there may be additional disciplinary actions from the department and university.

RULES OF THE GAME

We follow most parts of the classic rules in this assignment. You need to pay attention to the details listed below.

THE CLASSIC UNO RULES

source unorules.com

UNO cards Setup: The game is for 3-5 players, ages 7 and over. Every player starts with seven cards, and they are dealt face down. The rest of the cards are placed in a Draw Pile face down. Next to the pile a space should be designated for a Discard Pile. The top card should be placed in the Discard Pile. *This card must be a Number Card. If not, draw another card until it is. And the game begins!*

Game Play: The first player is normally the player to the left of the dealer (you can also choose the youngest player) and gameplay usually follows a clockwise direction. Every player views his/her cards and tries to match the card in the Discard Pile.

You have to match either by the number, color, or the symbol/Action. For instance, if the Discard Pile has a red card that is an 8 you have to place either a red card or a card with an 8 on it. You can also play a Wild card (which can alter current color in play).

If the player has no matches or they choose not to play any of their cards even though they might have a match, they must draw a card from the Draw pile.

The game moves on to the next person in turn, *even if your newly drawn card can be played*. You can also play a Wild card, or a Wild Draw Four card on your turn.

The game continues until a player has one card left. The moment a player has just one card they must yell "UNO!". If they are caught not saying "Uno" by another player before any card has been played by other players, the player must draw two new cards. Assuming that the player is unable to play their last card and needs to draw, but after drawing, is then able to play/discard that penultimate card, the player has to repeat the action of calling out "Uno". The bottom line is - Announcing "Uno" needs to be repeated every time you are left with one card.

Note: A player must not play any card other than a Number Card in this last card of play.

Once a player has no cards remaining, the game round is over, points are scored, and the game begins over again. Normally, everyone tries to be the first one to achieve 500 points, but you can also choose whatever points number to win the game, as long as everyone agrees to it.

Action Cards: Besides the number cards, there are several other cards that help mix up the game. These are called Action or Symbol cards.

source: unorules.com

- **Skip** - When a player places this card, the next player has to skip their turn.
- **Draw Two** - When a person places this card, the next player will have to pick up two cards and skip their turn.
- **Reverse** - If the order of players is going clockwise, switch to counterclockwise or vice versa.
- **Wild** - This card represents all four colors, and can be placed on any card. The player has to state which color it will represent for the next player. It can be played regardless of whether another card is available.
- **Wild Draw Four (a.k.a Draw Four)** - This acts just like the wild card except that the next player also has to draw four cards as well as forfeit his/her turn. With this card, you must have no other alternative cards to play that matches the color of the card previously played. If you play this card illegally, you may be challenged by the next player to show your hand. If guilty, you need to draw four cards. If not, the challenger needs to draw six cards instead.

Scoring: When a player no longer has any cards and the game ends and he wins the game. The rest players count their points. All number cards are the same value as the number on the card (e.g. a 9 is 9 points, a 0 is 0 points). "Draw Two" - 20 Points, "Reverse" - 20 Points, "Skip" - 20 Points, "Wild" - 50 Points, and "Wild Draw Four" - 50 Points. You can decide what to do with the points gained by each player - for example, doing one push-up per each point you have!

NOTES

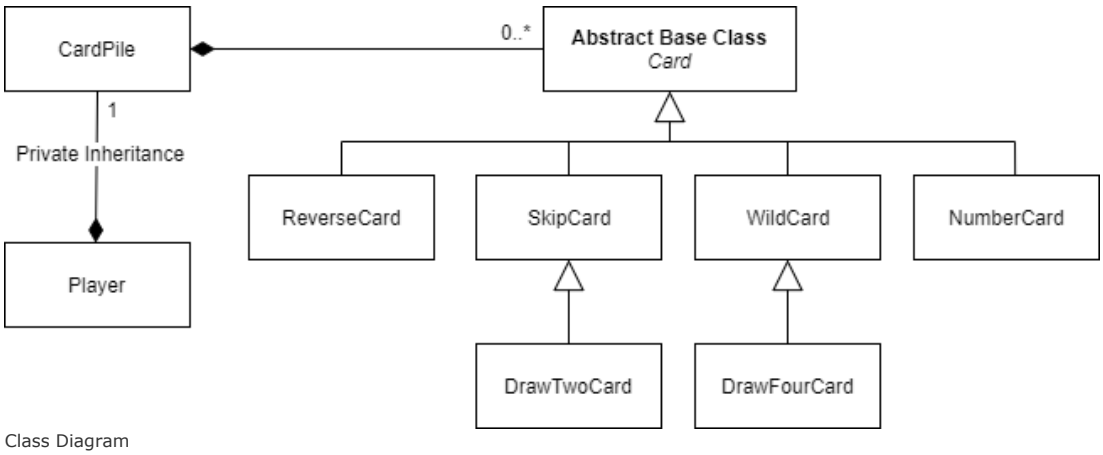
Some of you may have played 'In-House Rules UNO game' before. Please be aware that the followings 'In-House Rules' are not applied for this assignment:

1. Cannot cut-card [playing the identical card immediately without following the players' order].
2. Cannot accumulate the penalty to next player. If your previous player plays a Draw Two, you must draw two cards, even if you have a Draw Two in your hand.
3. Cannot play two identical card at a time.
4. When draw card is needed, you must draw the card and your turn is over, regardless the card being drawn matches with the top card or not.

In addition to the rules above, we assume our UNO game has three or more players, although our testing program does provide one-player or two-players option for debugging purpose. Any weird phenomenon happens in one or two players will not be considered as a fault when grading. For example, reverse card can has no effect card in 1-player or 2-players games; playing draw two card in a 1-player game may do nothing or draw himself two cards.

We also assume that the player will always say "UNO!" on their own during the game. Therefore, you program does not need to print extra UNO, but you are allowed to print 'UNO!' if you want.

CLASSES AND THEIR DESCRIPTIONS



Class Diagram

Note: You only need to implement the classes highlighted in yellow.

Classes Name	Files	Definition
enum class Color	enumAndConst.h	A enumeration class defining the colors. A Wild card and A Draw Four card has the color "meta" before it is played.

Card	Card.h
	Card.cpp
		An Abstract Base Class that has the following member functions.
		`bool operator^(const Card& t) const`
		It returns true if playing card t after this card is valid, otherwise return false.For example, "(a ^ b)" should return true if b can be played after a. Remark: we say that playing a Draw Four card is "legal" or "illegal" by considering if a player has any other card to play; but playing a Draw Four card after any card is always "valid".
		`void castEffect(Player*& currentPlayer, CardPile drawPile, CardPile discardPile)`
		A pure virtual function that triggers the effect of a card. All subclasses of Card should implement this function.
		Note, the sequence of playing a card should be:
		1. Judging if playing the card is valid; For example, it should stop when Red-5 trying to play after Blue-7.
		2. Removing the playing card from a player's hand;
		3. Casting the effect of the card (the castEffect function is called here);
		4. The card being added into the discard pile and becoming the top-card.
		`int getPoint() const`
		Return the private data member point.
		`void init()`
		does not do anything here for most of the card. This will be called when the deck is re-shuffled. WildCard and DrawFourCard assigned with specific color should restore its color to Color::meta in this function.
		`void serialize(ostream& os) const`
		A pure virtual function, which allows its derived class to print the card without overloading the operator<< in its derived class again. All non-abstract derived classes of it should have implemented this function.
		`Card::Card(Color, int)`
		A constructor with the parameters specifying the color and the point of the card.

NumberCard	NumberCard.h NumberCard.cpp	It is a derived class of **Card** (public inheritance). A NumberCard has a number from 0 to 9, and a color from four possible colors. When it is printed, it should show two characters. The first character is either Y, R, G, B, representing the color, and the second character is a digit 0 to 9. It should have a constructor `NumberCard::NumberCard(int, Color)` with the parameters specifying the number and the color of the card.
SkipCard	SkipCard.h SkipCard.cpp	It is a derived class of **Card** (public inheritance). It models the card "Skip". When it is printed, it should show two characters. The first character is the color, either Y, R, G or B, and the second character is lower-case letter 's'. It should have a constructor `SkipCard::SkipCard(Color)` where the parameter defines its color.
DrawTwoCard	DrawTwoCard.h DrawTwoCard.cpp	It is a derived class of **SkipCard** (public inheritance). It models the card "Draw Two". When it is printed, it should show two characters. The first character is the color, either Y, R, G or B, and thesecond character is plus sign (i.e. '+'). Please aware that: 1. Cards are drawn from the draw-pile, not the discard pile. 2. DrawTwoCard will skip the next player when it is played. 3. When there is not enough card to draw, all cards except the top card in the discard pile will be put back to the draw-pile and the draw-pile is shuffled. It should have a constructor `DrawTwoCard::DrawTwoCard(Color)` where the parameter defines its color.
ReverseCard	ReverseCard.h ReverseCard.cpp	It is a derived class of **Card** (public inheritance). It models the card "Reverse". When it is printed, it should show two characters. The first character is the color, either Y, R, G or B, and. the second character is lower-case letter 'r'.

		It should have a constructor `ReverseCard::ReverseCard(Color)` where the parameter defines its color.
WildCard	WildCard.h WildCard.cpp	It is a derived class of **Card** (public inheritance). It models the card "Wild Card". It prints differently before it is played and after it is played. Before it is played, it should print uppercase letters "WC". After it is played, a color should be fixed. It should print two characters. The first character is either R, B, G, Y and the second character should be lowercase letter "w". When a WildCard is played, it needs to ask the player to choose a color. It must be done by the Player's member function Color chooseColor() const. When we grade the assignment, we will check if you have successfully called the function chooseColor when the card is played. You should ask the player who plays this card to choose the color. When a WildCard is being shuffled in a card pile, you need to reset its color to Color::meta. It should have a default constructor with no parameter.
DrawFourCard	DrawFourCard.h DrawFourCard.cpp	It is a derived class of **WildCard** (public inheritance). It models the card "Wild Draw Four Card" or simply known as the "Draw Four Card". It prints differently before it is played and after it is played. Before it is played, it should print plus sign and digit four (i.e. "+4"). After it is played, a color should be fixed. It should print two characters. The first character is digit four "4", andthe second character is either R, B, G, or Y. When a DrawFour card is played, the player will be asked to choose a color first. Again, it must be done by the Player's member function Color chooseColor() const. Then, the next player will be asked whether he/she wants to appeal (challenge) or not. You must use the player's member function bool appealDrawFour() const to ask the player if he wants to appeal and review the current player's hand. Please refer to the comment written in Player.h for more details about how the appeal works. It should have a default constructor with no parameter.

CardPile	CardPile.h
	CardPile.cpp
		It is a collection of cards. Both draw-pile and discard pile are modeled by the **CardPile** class. A CardPile has an array of **Card** (cards). The data member size refers to how many cards the CardPile has. The bottom card should be stored in cards[0] and the top card should be stored in cards[size - 1].
		
		Member functions:
		
		`CardPile& operator+=(Card * c)`
		Add a card to the pile. The card will be added to the top of the pile. Do nothing if the parameter is a nullptr.
		
		`Card* removeCard(int index)`
		Remove and return the card at the index position. The array will be rearranged so that no space will be in between. Do nothing if index is out of bound.
		
		`Card* removeTopCard()`
		Remove and return the card at the top of the card pile.
		
		`const Card* getCard(int index) const`
		Similar to removeCard but it only returns a pointer to const which refers to the Card at index and the card should not be removed from the CardPile.
		
		`const Card* getTopCard() const`
		Similar to removeTopCard but it only returns a pointer to const Card which refers to the top card and the card should not be removed from the CardPile.
		
		`void shuffle()`
		To shuffle the pile. You need to shuffle the draw-pile only when there is no more card in the draw-pile. To do it, you need to put all except the top cards from the discard pile to the draw-pile and call this function to shuffle. WildCard and DrawFourCard should reset to Color::meta during shuffle.
		
		It should have a default constructor with no parameter.

Player	Player.h Player.cpp	Player inherits **CardPile** using **private inheritance**. It has two private data members: nextPlayer which is a Player pointer pointing to next player, and name which is a string that stores the name of the player. Note: the players should form a circular linked list. Member functions: `void drawCard(CardPile& drawPile, CardPile& discardPile, int number_of_cards)` It adds number_of_cards cards from the drawPile to Player's cards. As mentioned, if there is not enough card in the drawPile to draw, we need to put back all the cards in the discardPile (except the top one) to

		the drawPile and the drawPile will be shuffled. However, if the sum of discardPile and drawPile are not enough for the draw, this function should draw all possible cards for the player. For example, if the drawPile has 1 card, discardPile has 3 cards. Four cards need to be drawn. The player will first draw the card in the drawPile. Then 2 card from the discardPile (excluding the top card of the discardPile) are placed back to the drawPile. Shuffle the drawPile. The player draws the remaining two card from the drawPile.
		
		`Card* playCardAfter(const Card* topCard, int index)`
		
		The player tries to play the index-th card from his Card arrays (cards) after the Card topCard. The function shall return nullptr without doing anything if either:
		1. The play is invalid; or
		2. The index is out of bound (no such card).
		
		Otherwise, the card being played is removed from the Player's hand and the function returns the address of the card. Remember to change both variables cards and size when the last card is being played. We also assume that the player will always say "UNO!" on their own during the game. Therefore, you program does not need to print extra UNO, but you are allowed to print 'UNO!' if you want.
		
		`Color chooseColor() const`
		
		A function that you need to call to ask the player to choose a color after playing a Wild card or a Draw Four card. The function has already been completed for you.
		
		`bool appealDrawFour() const`
		
		A function that you need to call to ask the next player to appeal and review the current player's hand. The function has already been completed for you.
		
		`Player* getNextPlayer() const`
		
		Get the next player pointer.
		
		`bool win() const`
		
		To see if the player wins or not.
		
		`int getScore() const`
		
		Return the score of the player.

	game.cpp	A playable game. You can refer the logic of the game here.
	testNumberCard.cpp testReverseCard.cpp testSkipCard.cpp testWildCardandDrawFourCard.cpp	Test case of each class. You should at least be able to compile all testXXX.exe files. They should give you an idea on how the TA is going to grade your assignment.

TODO:

You task is to complete all the files shown below:

```
NumberCard.h NumberCard.cpp ReverseCard.h ReverseCard.cpp DrawTwoCard.h DrawTwoCard.cpp SkipCard.h SkipCard.cpp WildCard.h
WildCard.cpp DrawFourCard.h DrawFourCard.cpp Player.cpp
```

Meanwhile you are given the following files:

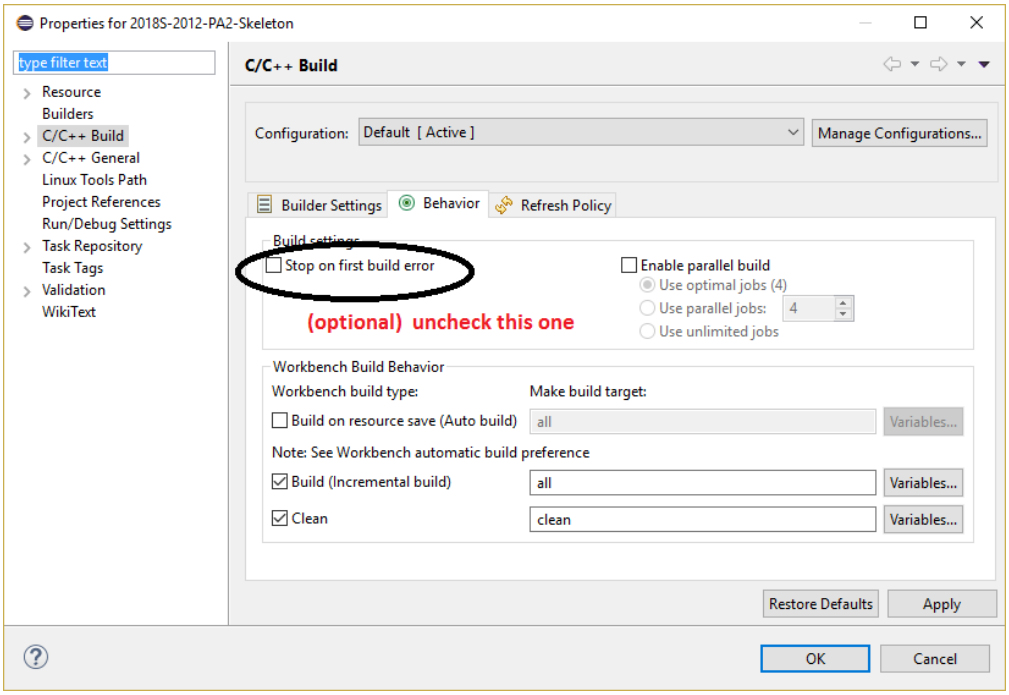
```
game.cpp testXXX.cpp Card.h Card.cpp CardPile.h CardPile.cpp Player.h enumAndConst.h Makefile
```

All these given files have already been completed for you. You don't need to submit these files. Even if you have submitted these files, they will be replaced by the TA during grading.

To compile your code, you need to create a makefile project (please check Lab 1 again on how to create a makefile project).

IMPLEMENTATION DETAILS:

- Please read the honor code and strictly follow it.
- You need to implement with a Makefile project. The Makefile is given in the skeleton.
- You should start with the given skeleton code.
- We will be grading your code on a Linux machine. Filenames are case sensitive in Linux. Therefore, when you want to include DrawTwoCard.h for example, you need to type #include "DrawTwoCard.h" but not "drawtwocard.h", although it would also compile on Windows.
- You are not allowed to use STL, global variables, static variable, goto. You can include if you wish to use some RTTI.
- You need to clean up your memory properly.
- You are not required to do any cin or cout inside your required function. It is likely you have committed some mistake if you do so.
- Optionally you may want to build the project partially before you have completed the assignment. In eclipse you need to turn off the "Stop on first build error" from the project properties. On linux/mac command prompt, you can type "make -k" for "keep going".



DOWNLOAD

Download Here or Download Here

Windows EXE

DEADLINE

23:59:00 on Apr 7th, 2018.

CANVAS SUBMISSION

The makefile would automatically generate a file called pa2.zip for you. Submit that file directly to Canvas. If you open the zip file, you should see all TODO files included in it. If you, however, cannot find this pa2.zip, you can create your own zip by simply zipping those files. Make sure you are using zip but not 7z nor rar.

Submit your work to Canvas. Please double check your submission after you have submitted it. Note: Canvas may append a number to the filename of the file you have submitted. e.g. pa2-1.zip. It is OK as long as you have named your file as pa2.zip when you submit it.

There will be a penalty of -1 point (out of a maximum 100 points) for every minute you are late. For instance, since the deadline of assignment 2 is 23:59:00 on Apr 7th, if you submit your solution at 1:00:00 on Apr 8th , there will be a penalty of -61 points for your assignment. However, the lowest grade you may get from an assignment is zero: any negative score after the deduction due to late penalty (and any other penalties) will be reset to zero.

FREQUENTLY ASKED QUESTIONS

Q: I am a Mac user and I got "ld: library not found for -lcrt0.o". What should I do?

A: This error comes from the flag -static at the line

```
g++ -o game.exe $(CFLAGS) -static obj/game.o $(CLASSES_OBJECT)
```

You need to remove the static flag

```
g++ -o game.exe $(CFLAGS) obj/game.o $(CLASSES_OBJECT)
```

Or to update the Makefile. The current skeleton has been updated to meet this change.