Inheritance: (40 minutes)

1. Instruction

1) Create Gradle Project.
2) Copy all folders in **the zip file** (except folder test) to your Gradle source code folder.
3) You are to implement the following classes (detail for each class is given in section 3 and 4) – Make UML for these classes too, otherwise your score will cap to 80%.

a) **BaseCard**	(package logic.card)
b) **NumberCard**	(package logic.card)
c) **DrawTwoCard**	(package logic.card)
d) **GameLogicUtility**	(package logic.utility)

4) JUnit for testing is in package test. Put test files in a correct folder.
5) **To submit:**
   5.1. **Export your project to a JAR file, with source code.**
   5.2. **Submit the JAR file on MyCourseville.**

2. Problem Statement: ONU

Figure 1 ONU Game logo

We create UNO-liked puzzle game called ONU. The basic rule is very similar to UNO, but ONU is a single player game. ONU's win condition is that player has to empty his/her hand before the deck is out.

In every turn, the player has to play 1 card from his/her hand by discarding it to the discard pile. However, each card type can only be played if certain condition(s) is satisfied.

There are 2 types of cards in ONU.

- a "number" card which has a number and a color on it. A number card can be played when it has the same color or same number as the top card of the discard pile. There are number 1 to number 4 cards for each color, RED, BLUE, YELLOW, and GREEN (16 in totals) in the deck.
- The second type is a "draw-two" card which has a draw-symbol and color on them. A player who plays a draw-two card has to draw 2 cards from the deck (there is only one player in our game anyway). A draw-two card can only be played when it has the same color as the top card on the discard pile. There are 4 "draw-two" cards in the deck, one for each color.

The game will end when the player's hand is out of cards or the deck is out of cards.

4. Implementation Detail

The class package is summarized below.

* In the following class description, only details of IMPORTANT **fields and methods are given.** *

4.1 Package logic.card /* You must implement this package from scratch*/

4.1.1 class BaseCard /* You must implement this class from scratch*/

This class is a base class for all kind of Cards. It contains all common elements which a card should have.

Field

Name	Description
- CardColor color	Color of the card. CardColor is enum in the package logic.game.
- CardSymbol symbol	Symbol of the card. CardSymbol is enum in the package logic.game.

Method

Name	Description
+ BaseCard(CardColor color)	This is the Constructor. Set color as one given in the parameter.
+ void play()	This method is called when the card is played. The results of actions are different for each type of Cards.
+ boolean ruleCheck()	This method returns true if the card can be played according to the rule and returns false when the card is illegal to play. The rules are different for each type of Cards.
+ getter-setter for all fields.	

4.1.2. class NumberCard /* You must implement this class from scratch*/

This class is a type of Card. The top card of the discard pile must have same color or the same symbol to make this card playable. Playing a number card will change the top card of the discard pile to this card but no special effect happens.

Method

Name	Description
+ NumberCard(CardColor color, CardSymbol symbol)	This is the Constructor. Set the information of the super class. Set symbol as one given in the parameter.
+ void play()	Use method setTopCard in class GameLogic to set this card to the top of the discard pile. You can call the method by using ***GameLogic.getInstance().setTopCard(.....)***. *You do not need to check for play legality.
+ boolean ruleCheck()	return true if the top card of the discard pile has the same color or same symbol as this card. Otherwise, return false. You can use ***GameLogic.getInstance().getTopCard()*** to get the top card of the discard pile.

4.1.3. class DrawTwoCard/* You must implement this class from scratch */

This class is a type of Card. The top card of the discard pile must have same color to make this card playable. Playing draw-two card will change the top card of the discard pile to this card and the player has to draw 2 cards from the deck.

Method

Name	Description
+ DrawTwoCard(CardColor color)	This is the Constructor. Set the information of the super class. Set symbol as ***CardSymbol.DRAW***.
+ void play()	Use method setTopCard in class GameLogic to set this card to the top of the discard pile. Then, use method draw in class GameLogic to draw 2 cards by setting the parameter of the method as 2. You can call the methods by using ***GameLogic.getInstance().setTopCard(.....)***. and ***GameLogic.getInstance().draw(.....)*** *You do not need to check for play legality.
+ boolean ruleCheck()	return true if the top card of the discard pile has the same color as this card. Otherwise, return false. You can use ***GameLogic.getInstance().getTopCard()*** to get the top card of the discard pile.

4.2 Package logic.game

4.2.1. class GameLogic /*ALREADY PROVIDED*/

This class is the main game system. This class provides cards in deck, hand and game rules. The game will be run via this class.

4.2.2. enum CardColor /*ALREADY PROVIDED*/

This enum contains values of color in this game.

4.2.3. enum CardSymbol /*ALREADY PROVIDED*/

This enum contains values of symbol on the cards.

4.3 Package logic.utility /* You must implement this package from scratch*/

4.3.1 class GameLogicUtility /* You must implement this class from scratch */

This class contains a method which will be used in GameLogic. Note: This is not the proper design of classes structure but it is required for scoring.

Method

Name	Description
+ boolean drawRule()	This method checks all cards in **ArrayList<BaseCard> hand** from GameLogic as follows ● Return false, if there is any legal card to play in hand. ● Return true, if there is no legal card in hand. You can get the hand of the player by using *GameLogic.getInstance().getHand()*.

4.4 Package main

4.4.1 class Main /*ALREADY PROVIDED*/

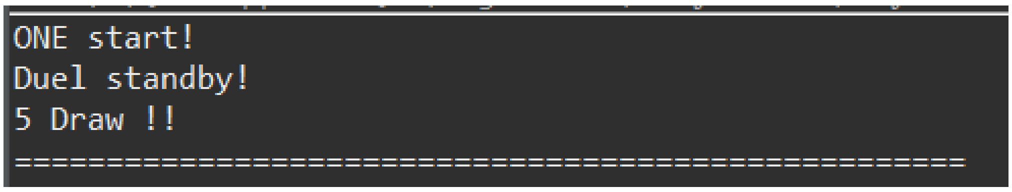
This class is the main program. You don't have to implement anything in this class. You can test the program by running this class.

4.5 Package test.grader /*ALREADY PROVIDED*/

This package provides JUnit test for each of your class.

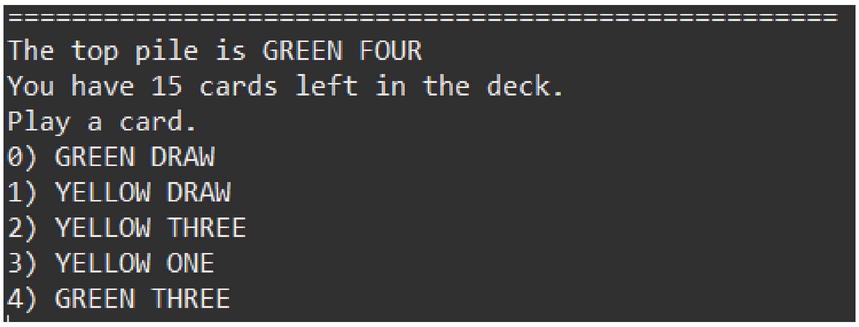
5. Finished Code Run Example

5.1. Start the game



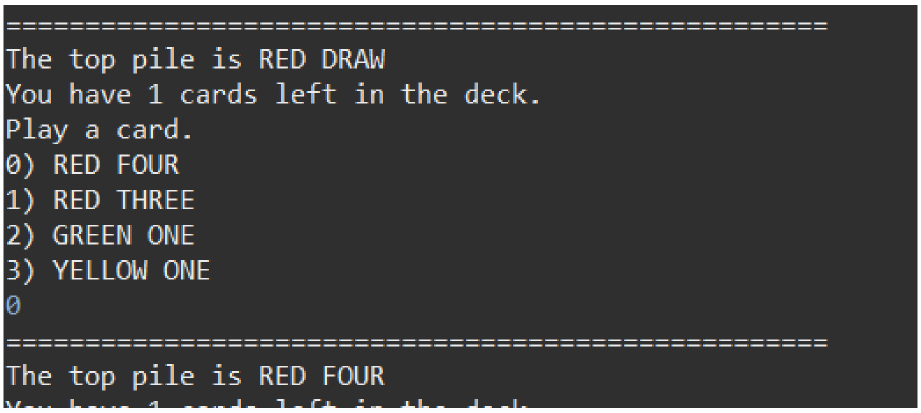
```
ONE start!
Duel standby!
5 Draw !!
===================================================
```

5.2. Game play



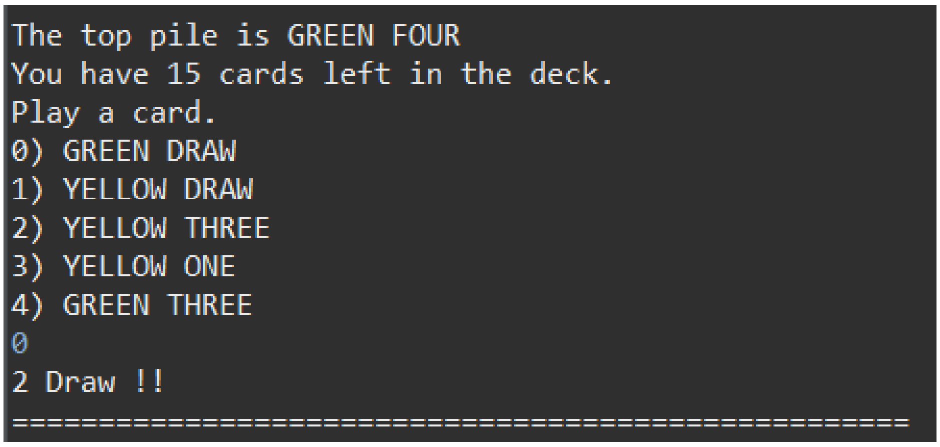
```
=====================================================
The top pile is GREEN FOUR
You have 15 cards left in the deck.
Play a card.
0) GREEN DRAW
1) YELLOW DRAW
2) YELLOW THREE
3) YELLOW ONE
4) GREEN THREE
```

5.3. Play number card



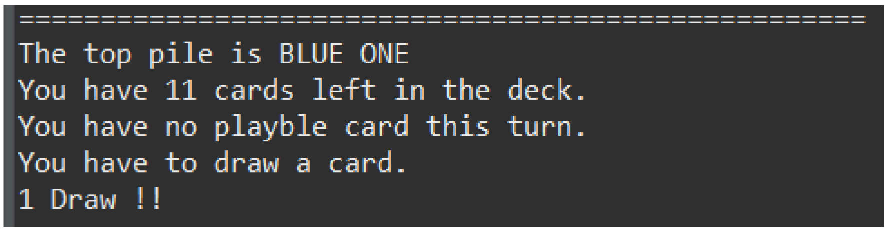
```
=====================================================
The top pile is RED DRAW
You have 1 cards left in the deck.
Play a card.
0) RED FOUR
1) RED THREE
2) GREEN ONE
3) YELLOW ONE
0
=====================================================
The top pile is RED FOUR
```

5.4. Play draw-two card



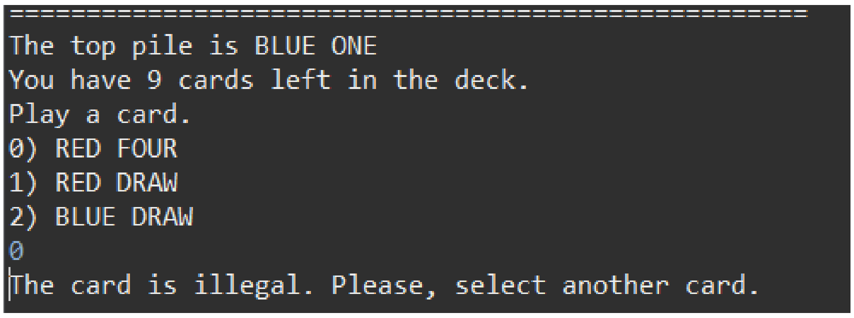
```
The top pile is GREEN FOUR
You have 15 cards left in the deck.
Play a card.
0) GREEN DRAW
1) YELLOW DRAW
2) YELLOW THREE
3) YELLOW ONE
4) GREEN THREE
0
2 Draw !!
=====================================================
```

5.5. Draw a card by rule



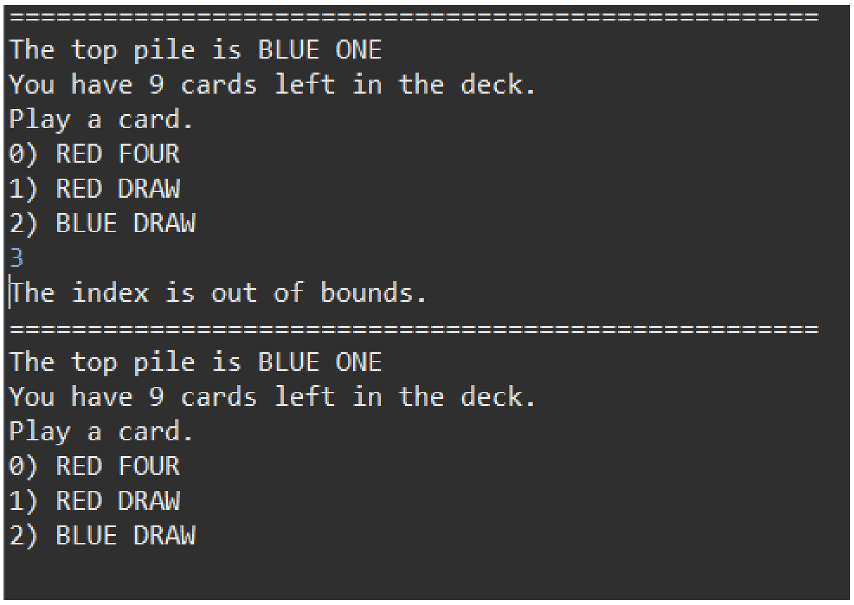
```
====================================================
The top pile is BLUE ONE
You have 11 cards left in the deck.
You have no playble card this turn.
You have to draw a card.
1 Draw !!
```

5.6. Play illegal card



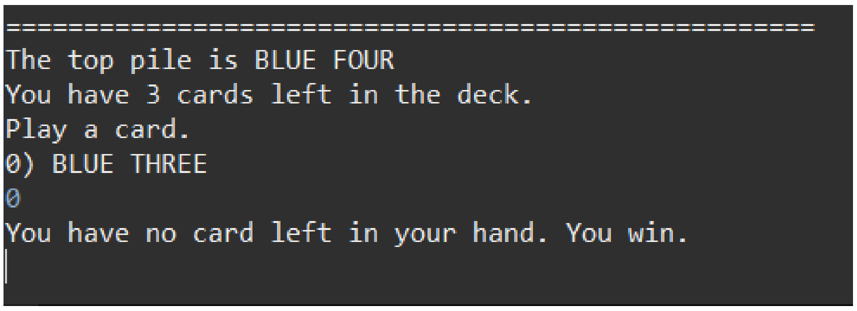
```
====================================================
The top pile is BLUE ONE
You have 9 cards left in the deck.
Play a card.
0) RED FOUR
1) RED DRAW
2) BLUE DRAW
0
The card is illegal. Please, select another card.
```

5.7. Input index out of bounds



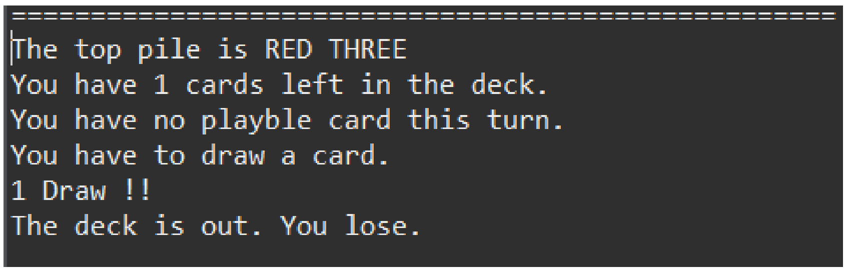
```
====================================================
The top pile is BLUE ONE
You have 9 cards left in the deck.
Play a card.
0) RED FOUR
1) RED DRAW
2) BLUE DRAW
3
The index is out of bounds.
====================================================
The top pile is BLUE ONE
You have 9 cards left in the deck.
Play a card.
0) RED FOUR
1) RED DRAW
2) BLUE DRAW
```

5.8. Win the game (empty the hand)



```
====================================================
The top pile is BLUE FOUR
You have 3 cards left in the deck.
Play a card.
0) BLUE THREE
0
You have no card left in your hand. You win.
```

5.10. Lose the game (empty the deck)



```
====================================================
The top pile is RED THREE
You have 1 cards left in the deck.
You have no playble card this turn.
You have to draw a card.
1 Draw !!
The deck is out. You lose.
```

6. Score Criteria

The maximum score for the problem is 13. Make UML for the 4 classes too, otherwise your score will cap to 80%.

6.1 Implement Class BaseCard, which does not have complete functions, correctly = 3 points

6.2 Class NumberCard:	**= 3.5 points**
testConstructor	= 1 points
testPlay	= 1 points
testRuleCheck	= 1.5 points
6.3 Class DrawTwoCard:	**= 3.5 points**
testConstructor	= 1 points
testPlay	= 1.5 points
testRuleCheck	= 1 points
6.4 Class GameLogicUtility:	**= 3 points**
testRuleCheck	= 3 points