SIM GLOBAL EDUCATION

UOW AUSTRALIA

SCIT

School of Computing and Information Technology
Faculty of Engineering & Information Sciences

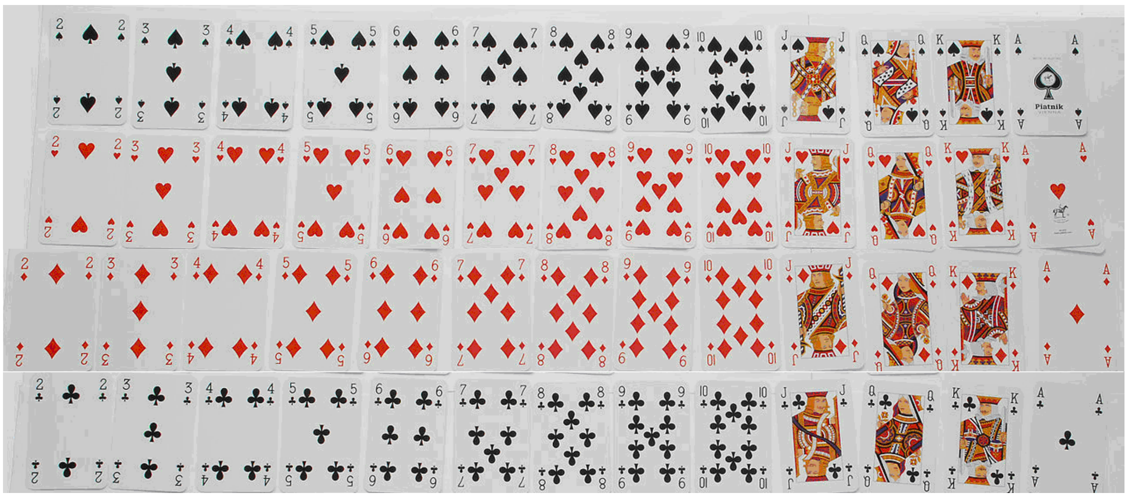
CSIT121
Object Oriented Design and Programming
Assignment 1

Objectives:

Practice java programming with classes and objects, constructors, copy constructors, enum type, array, `ArrayList` (generic version), overloaded methods, passing by reference and etc.

Task 1: (6 marks)

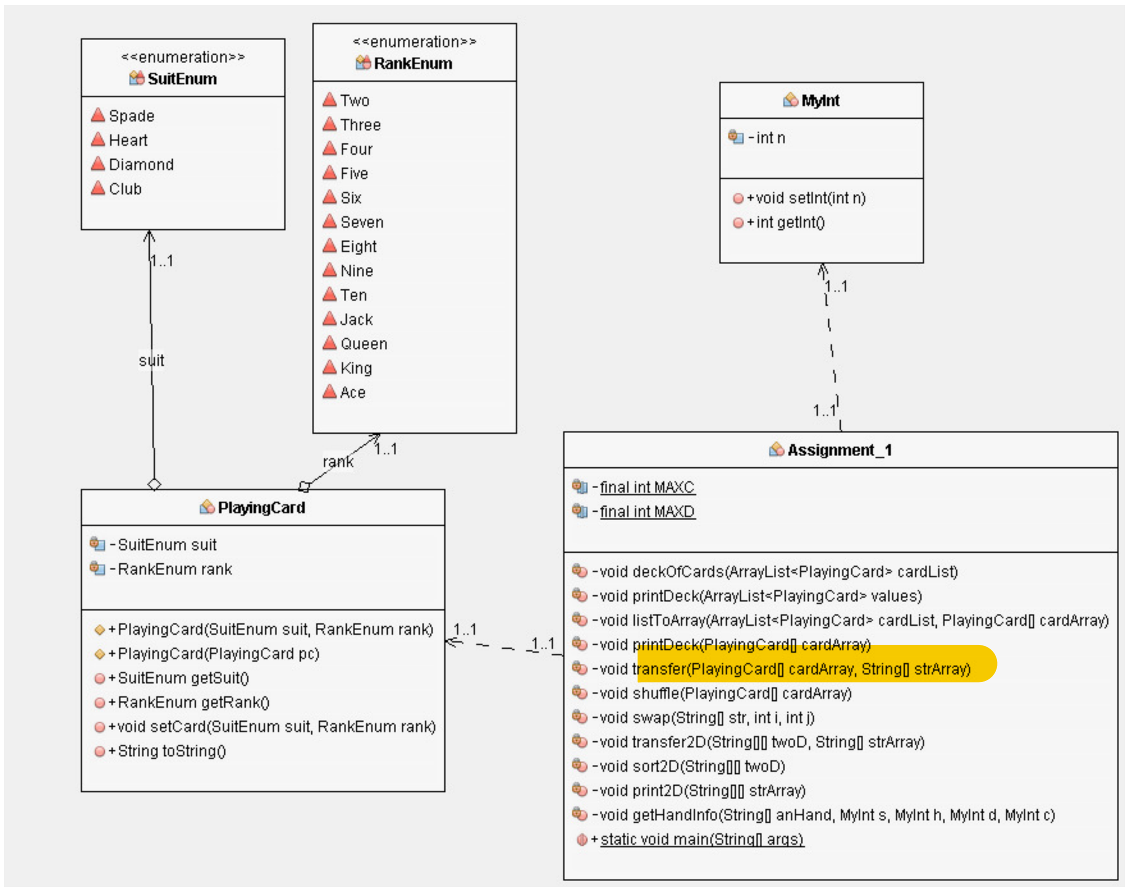
The following shows a deck of 52 playing cards:



- Four suits (Spade, Heart, Diamond and Club (in that order). We usually use characters `'S'`, `'H'`, `'D'` and `'C'` to denote the 4 suits.

- Rank values from Two, Three ……Ten, Jack, King, Queen and Ace (in that order). We usually use characters `'2'`, `'3'`, `'4'`, … , `'9'`, `'T'`, `'J'`, `'Q'`, `'K'`, `'A'` to denote the rank values.

The following UML diagram shows the objective of this assignment:

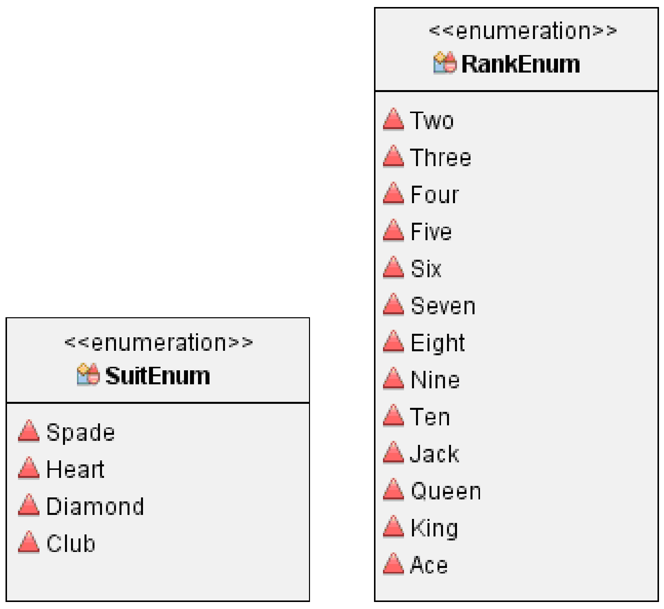


<<enumeration>>
SuitEnum
Spade
Heart
Diamond
Club
<<enumeration>>
RankEnum
Two
Three
Four
Five
Six
Seven
Eight
Nine
Ten
Jack
Queen
King
Ace
MyInt
-int n
+void setInt(int n)
+int getInt()
1..1
suit
1..1
rank
1..1
1..1
PlayingCard
-SuitEnum suit
-RankEnum rank
+PlayingCard(SuitEnum suit, RankEnum rank)
+PlayingCard(PlayingCard pc)
+SuitEnum getSuit()
+RankEnum getRank()
+void setCard(SuitEnum suit, RankEnum rank)
+String toString()
1..1
1..1
Assignment_1
-final int MAXC
-final int MAXD
-void deckOfCards(ArrayList<PlayingCard> cardList)
-void printDeck(ArrayList<PlayingCard> values)
-void listToArray(ArrayList<PlayingCard> cardList, PlayingCard[] cardArray)
-void printDeck(PlayingCard[] cardArray)
-void transfer(PlayingCard[] cardArray, String[] strArray)
-void shuffle(PlayingCard[] cardArray)
-void swap(String[] str, int i, int j)
-void transfer2D(String[][] twoD, String[] strArray)
-void sort2D(String[][] twoD)
-void print2D(String[][] strArray)
-void getHandInfo(String[] anHand, MyInt s, MyInt h, MyInt d, MyInt c)
+static void main(String[] args)

Let us know explore each of the tasks you need to design for this assignment:

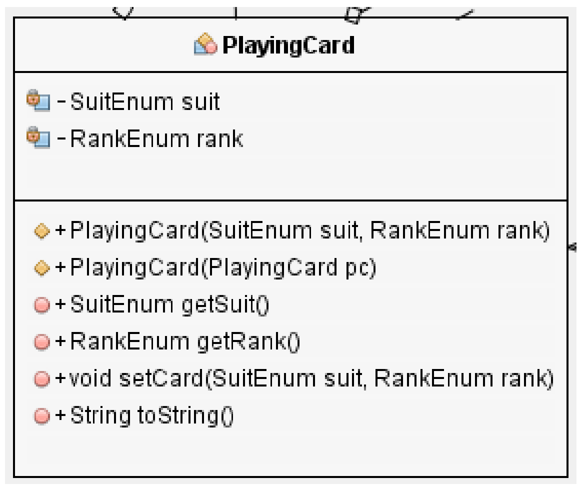
Task 1

Define two enum types, one for `Suit` whose constants (Spade, Heart, Diamond and Club) take one character parameter (the values as quoted above, `'S'`, `'H'` etc); and one for `Rank` whose constants (Two, Three etc) take one character parameter (the values as quoted above, `'2'`, `'3'` etc).



Task 2

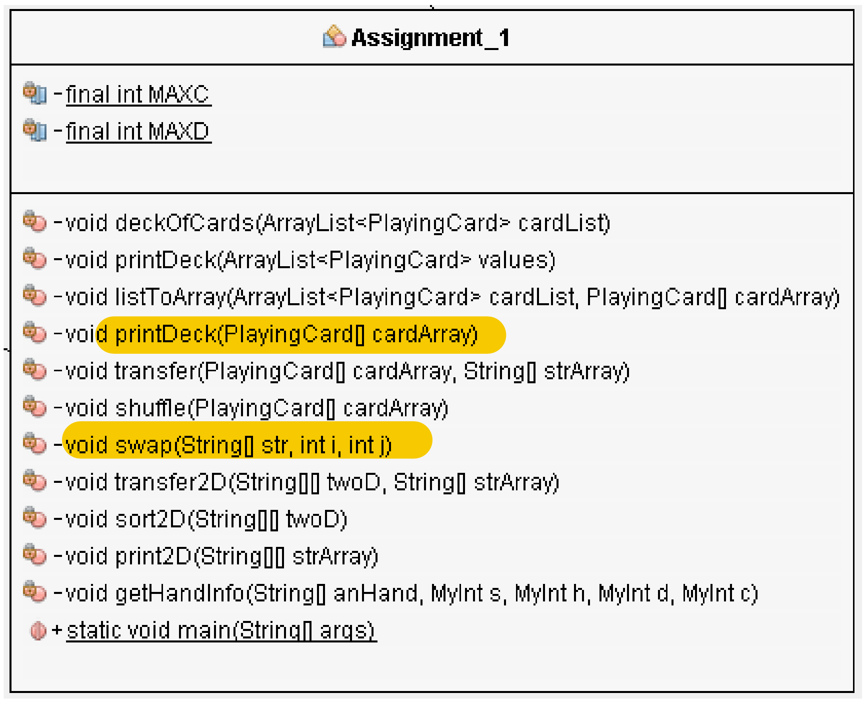
Suggest and implement a class called `PlayingCard` according to the following UML diagram:


PlayingCard
- SuitEnum suit
- RankEnum rank
+ PlayingCard(SuitEnum suit, RankEnum rank)
+ PlayingCard(PlayingCard pc)
+ SuitEnum getSuit()
+ RankEnum getRank()
+void setCard(SuitEnum suit, RankEnum rank)
+ String toString()

It is a very simple class with constructors, accessor methods, mutator method and a `toString` method (its format can be seen in the screen captured images).

Task 3

The following UML shows what you have to do in the main method:

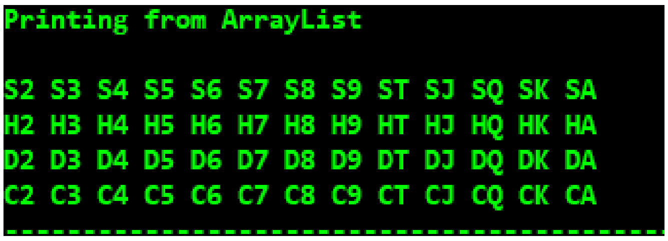


```
Assignment_1
-final int MAXC
-final int MAXD

-void deckOfCards(ArrayList<PlayingCard> cardList)
-void printDeck(ArrayList<PlayingCard> values)
-void listToArray(ArrayList<PlayingCard> cardList, PlayingCard[] cardArray)
-void printDeck(PlayingCard[] cardArray)
-void transfer(PlayingCard[] cardArray, String[] strArray)
-void shuffle(PlayingCard[] cardArray)
-void swap(String[] str, int i, int j)
-void transfer2D(String[][] twoD, String[] strArray)
-void sort2D(String[][] twoD)
-void print2D(String[][] strArray)
-void getHandInfo(String[] anHand, MyInt s, MyInt h, MyInt d, MyInt c)
+static void main(String[] args)
```

Task 3a

The duty of the main method is simple. You construct an `ArrayList` of `52` (`MAXD`) playing cards by calling the method `deckOfCards` to achieve the target; and display them (by calling the `printDeck` method to do this task). Note that each set of playing card has `13` (`MAXC`) of spades, hearts, diamonds and clubs. When the two methods are called in the main method, you will see the following display:

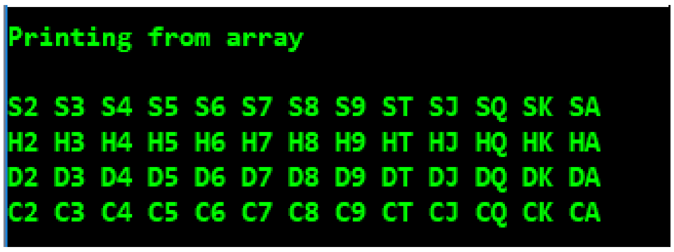


```
Printing from ArrayList

S2 S3 S4 S5 S6 S7 S8 S9 ST SJ SQ SK SA
H2 H3 H4 H5 H6 H7 H8 H9 HT HJ HQ HK HA
D2 D3 D4 D5 D6 D7 D8 D9 DT DJ DQ DK DA
C2 C3 C4 C5 C6 C7 C8 C9 CT CJ CQ CK CA
------------------------------------------
```

Task 3b

The next duty of this assignment is to transfer all the objects stored in the `ArrayList` (i.e. the list constructed in the **Task 3b**) playing cards to an array of playing cards (invoke `listToArray` method). During the transfer, *you need to have new objects to be stored in the array i.e. using the copy constructor of `PlayingCard`*; and print this array (by calling the `printDeck` method to do this task). Note that you have two overloaded `printDeck` method. When the two methods are called in main, you will get the following display:



```
Printing from array

S2 S3 S4 S5 S6 S7 S8 S9 ST SJ SQ SK SA
H2 H3 H4 H5 H6 H7 H8 H9 HT HJ HQ HK HA
D2 D3 D4 D5 D6 D7 D8 D9 DT DJ DQ DK DA
C2 C3 C4 C5 C6 C7 C8 C9 CT CJ CQ CK CA
```

Task 3c

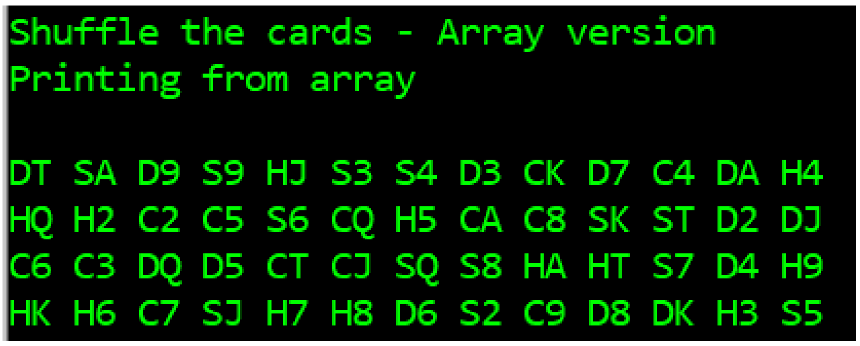
In this task, you construct an array of String objects by calling the `transfer` method. This method actually stores all the objects constructed in **Task 3b**

You are required to shuffle (the `shuffle` method) the array. You are not allowed to use the `shuffle` method defined in the `Arrays` class. What you will do in the shuffle method is to generate a positive integer `k` (reasonable value); and you need to shuffle the cards `k` times. To do this, you generate two integers `i` and `j`, and swap the playing card `i` and the playing card `j`.

In the above UML diagram, you see the two methods:

- The `swap` method swaps the `i`th and the `j`th string objects
- The `displayStringArray` method displays the `String` objects according to the above display.

When the `shuffle` method and the `printDeck` method are called, you will get the following display:



```
Shuffle the cards - Array version
Printing from array

DT SA D9 S9 HJ S3 S4 D3 CK D7 C4 DA H4
HQ H2 C2 C5 S6 CQ H5 CA C8 SK ST D2 DJ
C6 C3 DQ D5 CT CJ SQ S8 HA HT S7 D4 H9
HK H6 C7 SJ H7 H8 D6 S2 C9 D8 DK H3 S5
```

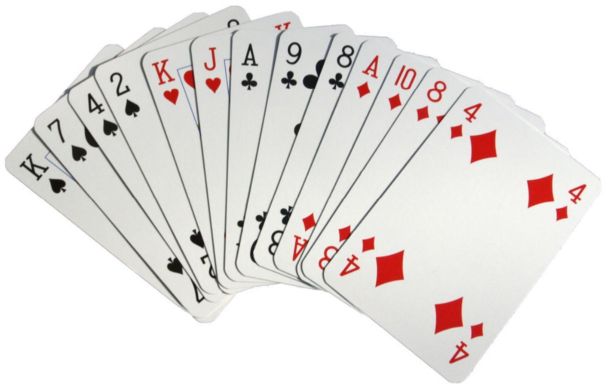
➔ Information will be stored in a 2D array in Task 3d

Task 3d

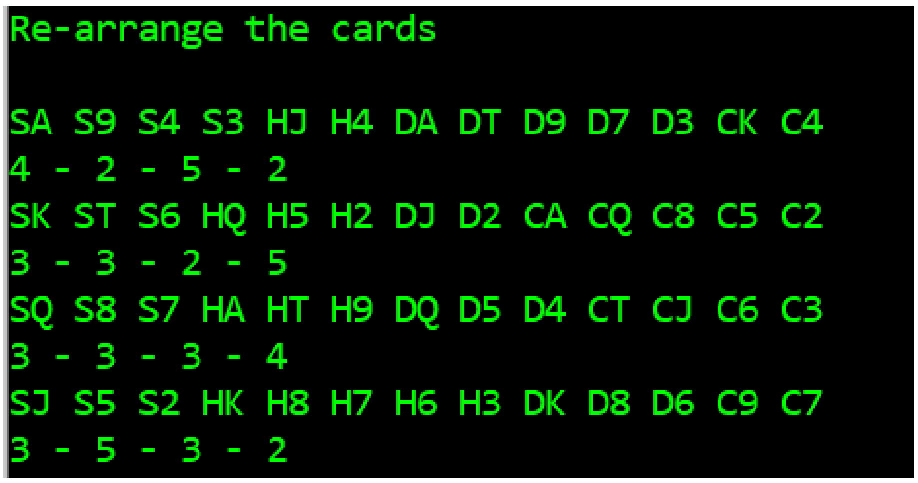
In the above screens captured, objects are printed by calling the `toString` method (implicitly) to print out the objects.

You then define a 2D array with 4 rows and 13 columns storing the randomly shuffled array (as shown in the above screen)

For each row (considered to be a hand), you sort them. Note that you should use the sort method in Arrays class. The following shows a picture of a hand:



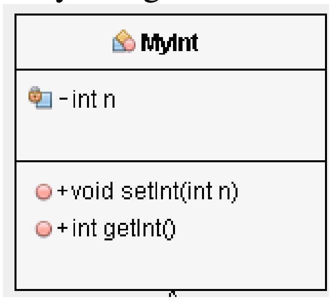
In each hand, you count the number of spades, hearts, diamonds and clubs and display the distribution. In the above hand, you have 4 spades, 2 hearts, 4 diamonds and 3 clubs.



```
Re-arrange the cards

SA S9 S4 S3 HJ H4 DA DT D9 D7 D3 CK C4
4 - 2 - 5 - 2
SK ST S6 HQ H5 H2 DJ D2 CA CQ C8 C5 C2
3 - 3 - 2 - 5
SQ S8 S7 HA HT H9 DQ D5 D4 CT CJ C6 C3
3 - 3 - 3 - 4
SJ S5 S2 HK H8 H7 H6 H3 DK D8 D6 C9 C7
3 - 5 - 3 - 2
```

Explore passing by reference by using the following class:



by getting the hand info (or distribution) i.e. the method `getHandInfo` as shown in the UML diagram

IMPORTANT

Put all your classes in a file called **`YourName_A1.java`** and make sure that this file can be compiled and can be executed. Upload **ONLY** this file to Moodle**. ALL ZIP FILE SUBMISSION WILL BE REJECTED**

No re-submission will be allowed after grading.

In the above file, remember to put down your name and also the following declaration (some similar contents):

// Tell me if it is your own work, and whether you have passed your
// program to your friends etc etc etc
// and willing to accept whatever penalty given to you.

- **Wrong file name -0.5 mark**
- **No declaration, no name etc -0.5 mark**

- **Failing to demo -1 mark**
- **Programs indentations and alignment of statements -0.5 mark**
- **Late penalty: -0.1 mark per hour**