Swinburne University of Technology
Faculty of Business & Enterprise

ASSIGNMENT 2

Assignment Value: 10% of your final mark
The assignment may be done individually or as pairs

Due Date:
8:30 am Monday, 22 of September, 2014
Note Late submissions will attract a penalty
(1 second late is considered to be 1 day late)

Submission Requirements

You must submit all source code, all executables and any other required files or folders in a single ZIP file via the ESP submission system.

https://esp.ict.swin.edu.au/

The name of the .ZIP file must use this naming convention:
1111111_Ass2.ZIP (where 111111 is your student id)
(or 1111111_222222_Ass2.ZIP if submitting as a pair)

Testing your software prior to submission and demonstration

Prior to submitting your assignment, please try to install your assignment on a different PC to where you have developed your system.

UnZip the assignment to a different path than where you have developed your system. Then test your system fully.

Section 1. (This section is worth 6 marks out of 10)

Create a Windows Form. It must have:

- Two multiline Labels to display some simple text
  A number of buttons labelled as follows
  - Add A Player
  - Add A Card
  - Add 20 Default Cards
  - Deal One Card to One Player
  - Deal a Card to All Players
  - Remove One Card from One Player
  - Remove a Card from All Players
  - Display Totals
  - Find a Card

Behaviours

Your form must demonstrate the following behaviours:

- **When form is loaded**
  - create a New Game object

- **When the user clicks Add A Player**
  - Your application asks the user to enter a player name
  - If a player with that name already exists, display "Player already exists". Exit Process
  - Create a new player object with that name
  - Add the player*** to the game's playerlist
  - (*** As we know, we do not add a player to the list, but we add a "reference to a player")

- **When the user clicks Create One Card**
  - Your application asks the user to enter a suit name
    - Limit the values to Hearts, Clubs, Spades, Diamonds
  - Your application asks the user to enter a card value
    - The value must be in the range 1 to 20

    NOTE: You may use a pop-up form with controls such as radio buttons etc to make user input easier
  - If that card already exists, display "Card already exists". Exit Process
  - Create a new card object with the suit name and numeric value
  - Add the card*** to the game's deck
    (*** As we know, we do not add a card to the deck, but we add a "reference to a card")

- **When the user clicks Create 20 Default Cards**
  - Write code that creates 20 cards and add them to the game's deck
  - The cards may be any values e.g. 9 Hearts, 4 Diamonds, 4 Spades …
  - This button is simply a time saver. The same effect could be achived if the user clicked the Create One Card button 20 times

- **When the user clicks Deal One Card to One Player**
  - Your application asks the user to enter a player name
    - If the player does not exist, display "Player does not exist" and exit the process
  - Your program must remove the first card from the game's deck of cards
  - It must then add this card to the player's list of cards

- **When the user clicks Deal a Card to All Players**
  - For each player in the game:
  - Remove the first card from the game's deck of cards and add it the player's list of cards
  - Stop the process if the deck contains zero cards

- **When the user clicks Remove One Card From One Player**
  - Your application asks the user to enter a player name
    - If the player does not exist, display "Player does not exist" and exit the process
  - Your program must remove the first card from the player's list of cards
  - It must then add this card to the game's deck of cards

- **When the user clicks Remove a Card From All Players**
  - For each player in the game:
  - Remove the first card from the player's list of cards and add it the game's deck of cards

- **When the user clicks Display Totals**
  - Your program must display the sum of numeric values of all cards pocessed by each player

    E.g. Tom 37
    Susan 25
    Harry 31
    Linda 13

- **When the user clicks Find A Card**
  - The program asks the user to enter a card name
    - If that card does not exists, display "Card does not exist". Exit Process
  - Your program must display the player who pocesses this card

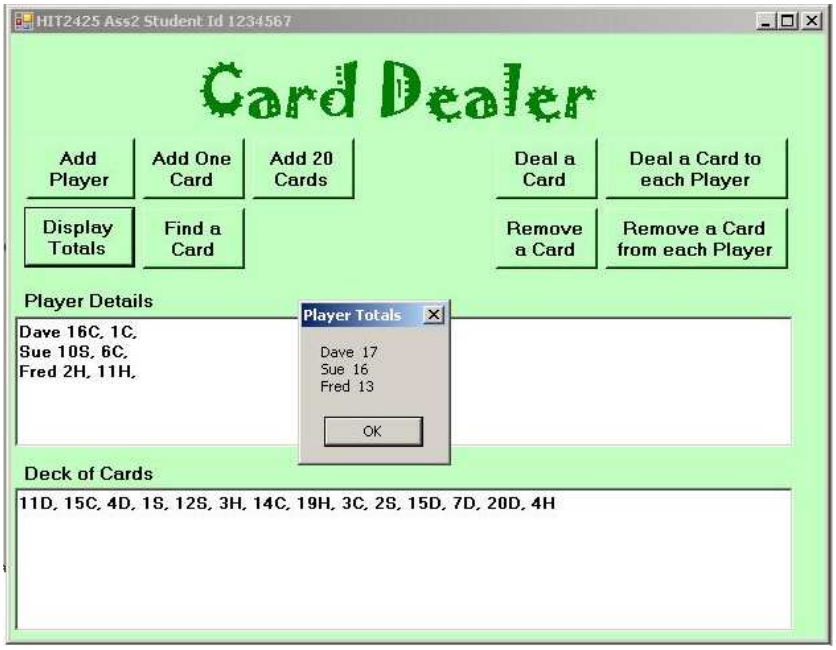
    e.g. Enter Card shortname: 5H This card is owened by Tom
    e.g. Enter Card shortname: 7D This card is owened by Linda
    e.g. Enter Card shortname: 4S This card is not currently owned by any player

- **After the form is loaded and after each button is clicked update the Two Labels.**
  - Label 1 shows all the players and the cards that they own:

    E.g. Tom 7H, 3C, 4D, 10S, 6D, 4H, 7S
    Susan 5D, 6S, 4s, 1H, 9D
    Harry 10H, 10D, 9C, 2C
    Linda 2S, 3S, 4C, 3D, 1C

  - Label 2 shows all the cards that exist in the deck *(i.e. all cards that are not pocessed by any player):*

    E.g. 5S, 7C, 8D, 1S, 5C, 10, 2D, 3H, 6C, 4S, 5H



Class Descriptions

Below is a list of Class Descriptions.
You must follow these descriptions precisely.

Class: Game

Private Instance Variables

mPlayerList	ArrayList	A list of all players
mDeck	Deck	A deck

You may add additional Private Instance variables if you wish

Public Instance Variables

None
You may not add any Public Instance variables

Constructor

ParameterList
None

Public Properties

ReadOnly	PlayerList()	ArrayList
ReadOnly	Deck ()	Deck

You may not add any Public Properties

Public Methods

Name	Return Data Type	Parameters	
AddCardToDeck	-	aCard As Card	Adds a card to mDeck
RemoveCardFromDeck	Card	-	Removes first card from mDeck
FindPlayer	Boolean	PlayerName As String	Returns True if the Player's name exists in PlayerList
GetPlayer	Player	PlayerName As String	Returns a Player that has a matching name. Returns Nothing if no match found
AddPlayer	Boolean	PlayerName As String	Adds a player to mPlayerList
GetPlayerWithCard	Player	Card ShortName as String	Return the player who pocesses this card. If no match is found, return Nothing
DealOneCardToPlayer	Boolean	aPlayer As Player	Removes first card from mdeck and adds it to the player's cardlist If mdeck has no cards then return False
RemoveOneCardFromPlayer	Boolean	aPlayer As Player	Removes first card from player's cardlist and adds it to the mdeck. If the player has no cards then then return False

You may not add any Public Methods

Private Methods

You may add additional Private Methods if you wish

Deck Class

Private Instance Variables

mCards ArrayList List of cards

You may add additional Private Instance variables if you wish

Public Instance Variables

None

You may not add any Public Instance variables

Constructor

Initialises Instance variables

ParameterList None

Public Properties

ReadOnly Cards() ArrayList

You may not add any Public Properties

Public Methods

Name	Return Data Type	Parameters	
FindCard	Boolean	Suit As String NumericValue As Integer	Returns True if a card with the same suit and numeric value already exists
AddCard	-	aCard As Card	Adds a card to the mCards
RemoveCard	Card	-	Removes a card from mCards

You may not add any Public Methods

Private Methods

You may add additional Private Methods if you wish

Card Class

Private Instance Variables

mSuit	String	Full Name of suit (e.g. Hearts)
mNumericValue	Integer	Value of the card (e.g. 9)

You may add additional Private Instance variables if you wish

Public Instance Variables

None

You may not add any Public Instance variables

Constructor

Initialises Instance variables

ParameterList:

Suit	String
NumericValue	Integer

Public Properties

ReadOnly	Suit	String
ReadOnly	NumericValue	Integer

You may not add any Public Properties

Public Methods

Name	Return Data Type	Parameters	Comments
ShortName	String	-	Returns a string showing the NumericValue and Abbreviated suit of the the card E.g. returns 9H for the 9 of Hearts

You may not add any Public Methods

Private Methods

You may add additional Private Methods if you wish

Player Class

Private Instance Variables

mName	String	Player's name
mCardList	ArrayList	List of cards pocessed by the player

You may add additional Private Instance variables if you wish

Public Instance Variables

None

You may not add any Public Instance variables

Constructor

Initialises Instance variables

ParameterList:

PlayerName String

Public Properties

ReadOnly	Name()	String
ReadOnly	CardList()	ArrayList

You may not add any Public Properties

Public Methods

Name	Return Data Type	Parameters	Comments
FindCard	Boolean	Suit As String NumericValue As Integer	Returns True if a card with the same suit and numeric value already exists
AddCard	-	aCard As Card	Adds a card to the players CardList
RemoveCard	Card	-	Remove a card from the players CardList
Total	Integer	-	Returns the sum of all numericvalues of all items in CardList

Section 2. (Each task in this section is worth 1 mark)

a. Swap cards between two players

- Create a button on the form labelled Swap Cards
- When the user clicks this button
  - The system swaps all the cards owned by Player 1 with the cards owned by Player 2
  - The label that displays player details is updated

This requires a some additions to the Game class:

Public Methods

Name	Return Data Type	Parameters	Comments
SwapPlayerCards	-	Player 1 name As String Player 2 name As String	Swaps cards between two players.

b. Create Auto Deck

- Create a button on the form labelled Create Auto Deck
- When the user clicks this button
  - The system asks the user to enter number of suits required
  - Suits are added in alphabetical order
    - If the user requests 2 suits then two suits are used (Clubs & Dimonds)
    - If the user requests 3 suits then three suits are used (Clubs & Dimonds & Hearts) …
  - The system asks th euser to enter number of cards required in each suit
  - If the user enters 8, then the system will create 8 cards in each suit e.g. 1 Clubs to 8 Clubs
  - The system then automatically adds the appropriate cards to the game's deck of cards
  - The label that displays cards is updated

c. Suffle Cards

- Create a button on the form labelled Shuffle Cards
- When the user clicks this button
  - The system executes the ExchangesPositions method 100 times
  - The label that displays cards is updated

This requires a some additions to the Deck class:

Private Instance Variables

mRandNoGen Random

Constructor

Creates a new Random object

Public Methods

Name	Return Data Type	Parameters	Comments
ExchangesPositions	-		Swaps the position of two cards in the Cards list. Each position is a random number in the range 0 to Cards.Count -1

d. Card/Player History

- Create a button on the form labelled Display Card/Player History
- When the user clicks this button
    - The system displays every card in the game and the names of players have pocessed the card
        - A player may have pocessed one card a number of times
        - If the card is not currently pocessed display "No owner"

      You can display this output in a messagebox / listbox / label etc.

e.g. 1 Clubs: Tom, Harry, Linda, Tom
2 Clubs: Linda, No Owner
3 Clubs: No Owner
4 Clubs: Susan
…

This requires a some additions to the Card class:

Private Instance Variables

mPlayerList	ArrayList	A list of all players who have pocessed the card

Public Properties

ReadOnly	mPlayerList()	ArrayList

Public Methods

Name	Return Data Type	Parameters	Comments
AddPlayerToHistory	-	Player	Adds a player to the card's player list

Marking Guide

Each student will start with 10 marks.
This assumes that you will submit a **fully** working system.
Marks will be deducted where portions of your submission

- have not been implemented
- do not work properly
- do not meet specifications

Requirement, Function or Specification	Maximum Deduction if not completed / not correct
Lots of reasonable code, but the application fails in many places. Ignore remainder of marking guide.	-8
Add A Player works correctly	-3
Create One Card works correctly	-3
Create 20 Cards works correctly	-1
Deal 1 Card to Player works correctly	-3
Remove 1 Card from Player works correctly	-3
Displays Totals works correctly	-1
Find A Card works correctly	-1
Swap Cards between Players works correctly	-1
Create Auto Deck works correctly	-1
Shuffle Cards works correctly	-1
Card / Player History works correctly	-1
Poorly named variables and/or procedures	-1
Lack of comments	-1
If **any** method contains more than **30** lines of code	-1