FIT2081
Laboratory for Week 05

Relates to Lecture 04

This Lab is one of 11 you will attend during the semester. Each lab is worth 4 of your final marks for the unit. The best 10 marks will be used.

In the last 30 minutes of the each lab (or before if you ask and the tutor has time) your tutor will mark you. You will be asked a few questions to determine whether you **confidently** understand the work you are presenting. You will be awarded a mark of 0 if **any** trace of plagiarism is detected during this questioning. You may prepare your solution before the lab in which case you can ask to be marked at the earliest opportunity.

Purpose:

- Practice coding small classes and test drivers for these classes

Important

- Each question should be answered in its own class in its own project or within a common project. Please note the compiler compiles all classes in a project at once (this can be changed but not recommended for beginners)

Q1 (1 mark) – Code a Coin class

The class should allow the coin to be flipped to produce a random face up (heads or tails). It should also be able to report its state (i.e. which face is currently uppermost) to driver code via an appropriate accessor. Only code methods required to meet the preceding specification.

Exercise the Coin class using a driver class in which a coin is flipped 1000 times and the number of heads and tails obtained is displayed.

Your tutor will expect you to run this driver and explain why the results are the expected results.

Q2 (1 mark) – Use an Array of Coin

Exercise the Coin class using a driver class in which 5 coins are flipped, as a group, 1024 times. Display the number of times exactly 4 tails are obtained. Display the number of times exactly 5 tails are obtained. Use an array of Coin.

Your tutor will expect you to run this driver and explain why the results are the expected results.

Q3 (1 mark) – Coding a Playing Card class

The class has two instance variables of type int which together hold a state of card value and card suit (1= clubs, 2 = diamonds, 3 = hearts, 4 = spades).

Code the following constructor and methods:

Constructor:
2 int parameters to set the state. The state should be set via mutators.

Mutators:
setValue/setSuit should both take a single int parameter and should validate this value before using it to set their respective instance variable.
They should both return a Boolean to indicate to driver code the success/failure of the set.

Accessors
getValue should return a car's current value as a String ("Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King").

The best way to perform the necessary translation from int to String is to use an Array of String initialised appropriately. Then use the card's current int value as an index to look up the corresponding String in the Array.

getSuit … (similar to getValue)

toString
Should return a String like "Ace of Spades", "Five of Clubs", "Jack of Diamonds" etc.

Exercise the Card class with a driver class

Q4 (1 mark) – Coding a Deck of Cards class

Only attempt this if you feel you are up to the challenge otherwise be content with 3 out of 4 for this week (which is still a distinction btw).

- Code a Deck class that holds 52 unique (no duplicates) Card objects in an array of Card.
- The constructor should create a deck with the cards in value order within suits.
- There should be a shuffle method which randomises the order of cards in a full deck and
- A deal method which returns a Card from the deck or null if the deck is empty.
- The Deck class should also have a fill method to restock its array of Card when empty and
- a toString method which leverages the Card class's toString method.

Exercise the Deck class with a driver class by dealing an entire deck.