# COSC 1047 – Winter 2017 – Assignment 5
# See D2L For Due Date

This assignment is designed to give you some experience writing and using collection classes. This assignment is done using Eclipse. Import the .zip file that is provided on D2L and when it is complete, export it again as a .zip. All submissions should be done through D2L. You can submit and resubmit as many times as you want up to the due date and time. If you have any problems submitting, email me – alangille@cs.laurentian.ca - in a timely manner. **Make sure your name, student number and a brief description of your program appear at the top of each file in comments. Also make sure that you use comments to clarify your code for marking.** ***Failure to do either of these tasks may result in a deduction of marks.***

***Submit solutions to questions 1 and 2.***

***Remember if you work in partners to include all names in the source code. The person who is left out will receive zero.***

Note: It is imperative that you import the a5-w17.zip file from D2L properly or you won't be able to work in the proper project environment. Although the screen shots are a bit outdate these instructions are correct:

http://agile.csc.ncsu.edu/SEMaterials/tutorials/import_export/

---

Grading:

For the submitted questions the grading is as follows:

| | |
|---|---|
| Proper documentation, formatting, comments and Java Docs where requested: | 5 marks |
| Program compiles properly and runs: | 5 marks |
| Program runs properly for most of the examples and requirements of the question: | 5 marks |
| Program runs properly for all of the examples and requirements of the question: | 5 marks |
| Extra work where applicable: | 5 marks |

Small non-critical errors and suggestions for improving code will be given as comments and should be addressed in future assignments. Major errors and failure to follow the question guidelines will result in deductions per the scheme listed above.

1. (20 marks, ) Consider the classes included in the q1 package. These include , and . Card represents a playing card, while is a simple collection class that represents … wait for it … a deck of cards. The class uses an array and you should take a minute to review how it works. We've discussed in the lectures how a linked list is a data structure "engine". Your task in this question is to re-work the class to use Java's built-in collection class instead of the array that is there now. Here are a few considerations:

   - You need to change the data field in to a (don't forget any necessary import statements).
   - The constructor needs to be rewritten. Start with an empty and add all 52 cards in order. Look at how the constructor works now to know what order they should be put in. Keep in mind that a takes a single integer as an argument that determines both the suit and the rank.
   - needs to be rewritten. You'll need to find a way to randomize the cards without losing any of them. You could try using an iterator to grab a card from a random location then moving to a new random location in the list and adding the card back in – but that's just one possible approach.
   - needs a bit of work too – as with the current method, you need to remove the top card and return it. The top must leave the list – we don't deal cards AND keep them in the deck unless we are playing at a crooked table.
   - needs to return the number of cards left in the deck. This ~could~ be used when dealing to figure out if there are enough cards left to deal.
   - should return true if there are no cards left in the deck.
   - should be rewritten. Take a look at the existing one. It prints out the cards left in the deck. You don't need to worry about the rank and suit, they are part of the class. I suggest an here to walk through the .

   Just to be clear – ALL of your modifications should be done in the class. Do NOT modify Card or . The markers will run to make sure that your class works properly. Again, the idea here is to recognize that we can change the engine of a data structure without changing its external interface (i.e., it still works the same). As you are modifying , comment out some of the lines in so that it's not trying to test methods you haven't adjusted yet.

2. (30 marks, , ) Every ~real~ gamer carries around a bag of dice just in case of a game-emergency. Your job in this question is to rewrite the class so that it is generic and can hold any type of object. This means you need to change the datafield and then correct all of the methods that use it (12 marks). Then, when this is done, write a GUI that allows you to create a bag of dice objects. This means your GUI has the following functionality:
    a. Create an empty . It is ok to have it pre-set to use dice as in
    b. (3 marks) The GUI should have inputs so that you can create dice and add them to the bag. You can use text fields but this might be a good place for radio buttons or a drop down menu since there are only a few valid dice configurations.
    c. (3 marks) The GUI should allow you to grab a random die from the bag and roll it. This can be done with a single click – grab-and-roll. Remember, the method doesn't actually remove the from the bag.
    d. (3 marks) The GUI should allow you to remove a die from the bag. For this you'll need to specify the configuration of die being removed – i.e., removing a 6-sided die.
    e. (3 marks) The GUI should allow you to print the dice that are in the bag.
    f. (3 marks) The GUI should display (on demand) the number of occurrences of a particular configuration (ask the user for the configuration of interest).
    g. (3 marks) For good measure your GUI should also allow you to print out the number of dice in the bag and the current capacity of the bag.

    You are given the class. No modifications should be needed for this class. You are given the class to use as a reference. I recommend that you copy this class so you can use it as a reference. ***Leave this class with an array engine, DO NOT change it to a LinkedList.***