# CIS\*3260 W21 – Assignment 1

# Questions and Answers – Part 3

**Are we being graded on how detailed our comments are within our Ruby code?**

Yes, a little bit. It would fall under OO style and/or Ruby style and is up to the TAs discretion.

**Are we supposed to use the use case template from the lectures:**

**- Use case name**

**- Primary Actor**

**- Goal - description of use case purpose**

**- Stakeholders List**

**- Initiating Event**

**- Main Success Scenario**

**- Post Conditions**

**- Alternate flows or exceptions**

**- Use cases Utilized**

**- Scenario Notes**

**or can we just have this (would the following be an example of a use case for this assignment?):**

**- Use case name: i.e. Use case 1: Rolling a die  
  - Primary Actor: Player 1**

**- Initiating Event: The player wants to roll a die and know its results**

**- Main success scenario:   
 Player creates a die, sends in the sides and colour, rolls the die and checks the results.**

Your choice of Use Case is somewhat flexible, but your second example would be fine. It just needs to be a legitimate Use Case, with at least a serviceable Main Success Scenario.

**And the tests would just be following the main success scenario and checking if what we expected matches the actual results?**

Yes.

**How do we test the abstract classes if those methods have been overridden in the child class?**

You don’t have to test the abstract class. You just have to test its methods inside at least two of its children class instances (to make sure it is generally inherited).

You could create an “abstract class” instance and show that the method exists, can be called, but produces an error message, or whatever you decided it should produce, but this is not necessary.

**In the Q and A you mentioned that calling something like coin.getArray (where it would send the array that holds the coin and dice for the class) would be breaking encapsulation.**

**Instead of sending the array, would it be okay if we sent a clone/copy of the array in coin.getArray? Would that be considered not breaking encapsulation since the other classes won't be able to change the original array?**

Yes, that would not break encapsulation.

This approach has advantages and disadvantages as a design technique. Depending on the compiler/interpreter, code done this way can produce “faster” and possible more streamlined code.

However, it still has the “feeling” of exposing the internals of the object in an “unnatural” way and not elegant from an OO purist perspective.

However, the principle of encapsulation definitely still holds in this scenario.

**When it says "if you want all items, supply the symbol :all",**

**does this mean that we pass in all the selected items**

Yes

**or does it mean we just empty the bag contents in the hand ignoring the description?**

No

**For the Player class: are all the throws being stored (in something like an array)?**

Yes