CS 5004 Module 6 Worksheet
Inheritance vs. Composition

Each worksheet is to help you remember and understand the materials contained in each module so that you can get the most out of your learning experience. We start by asking questions that are covered in the module but will also ask you to apply the knowledge in new ways. By taking the time to answer each question carefully using your own thoughts and words, these worksheets can serve as the basis for review for the more formal assessments in this class.

1. What is meant by *software architecture?*

The structure of the software, how these elements are organized and how they relate to each other.

2. Why is a UML approach to design useful?

UML is programming language agnostic and acts as a lingua franca. It is like a common notation that developers can then turn around and implement them. UML is also high enough in abstraction that non-technical and semi technical group can understand.

3. Explain three (3) benefits of *code reuse?*

1) Reused code is likely to be more robust than whatever we can write from scratch.
2) Reusing existing code lessens time and effort.
3) Reused code has the advantage of being tested periodically and realistically and thus becomes robust with age. It is a good idea to reuse it whenever applicable.

4. What is the difference between *sub-typing* and *sub-classing*? Give an example of each.

Subclasses allow one to reuse the code inside classes - both instance variable declarations and method definitions. Thus they are useful in supporting code reuse inside a class. For example, the car class extends the vehicle class, car class is the subclass of vehicle class and it can use all the implementation of the super class.

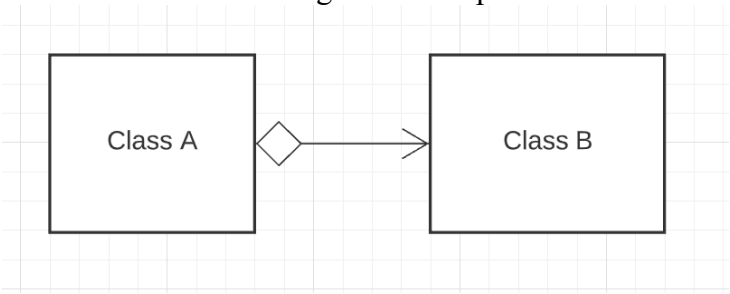
Subtyping on the other hand is useful in supporting reuse externally, giving rise to a form of polymorphism. It occurs when we drive an interface from another. For example, in our homework we define a shape class, and the circle class and rectangle class are subtypes of shape class. They inherited only the object, not the implementation.

Sub-classing is a kind of sub-typing, but sub-typing may not necessarily be a sub-classing.

5. What is the difference between *inheritance* and *composition*. Give an example of each.

The composition is a design technique in which our class can have an instance of another class as a field of our class. Inheritance is a mechanism under which one object can acquire the properties and behavior of the parent object by extending a class.
In other word, inheritance is like a “is a” relationship, while composition is like a “has a” relationship.

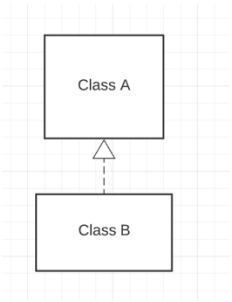
6. Suppose that there is a *compositional relationship* between two classes called `ClassA` and `ClassB`:
    a. Draw the UML class diagram that expresses this relationship.



    b. Declare the required class definition that shows this relationship.

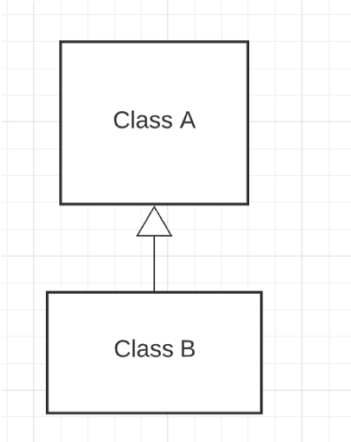
    Class B is part of class A, class A contains object of class B and use some methods of class B.

7. Suppose that there is a *implements relationship* between two classes called `ClassA` and `ClassB`:
    a. Draw the UML class diagram that expresses this relationship.



    b. Declare the required class definition that shows this relationship.
    Class B implement class A. Class A is a interface class, and it only contains objects, not implementations.

8. Suppose that there is an *extends relationship* between two classes called `ClassA` and `ClassB`:
    a. Draw the UML class diagram that expresses this relationship.



    b. Declare the required class definition that shows this relationship.

Class B extends class A. Class B extends all the method implantation of class A except for private variables and methods.