SENG2200/6220 –Programming Languages & Paradigms
Computer Lab for Week 4, Semester 1, 2019

Objectives

This lab aims to review the knowledge of *UML*, *data structure* and OO *programming skills* (Java and C++). At a successful completion of the exercises, you should be able to create UML class diagram from specification, understand/implement inheritance and polymorphism of OO design.

Questions

1. Write a doubly-linked list in Java, such that it uses two sentinel nodes and a linear structure. As well as the sentinel reference, the list should allow the identification of a "current" node within the list, and also insertion before the current node and removal of (the object associated with) the current node. Make sure that you also specify the Node class for the list.

2. Draw a UML diagram to represent the following information.
   a. A Vehicle is an object that can hold a certain number of passengers. Each vehicle also has a warning system, but this is different for each vehicle.
   b. A Wheeled Vehicle is a Vehicle that also has a certain number of wheels, a Wheeled Vehicle's warning system is typically a horn.
   c. A Car is a Wheeled Vehicle with four wheels, and its warning system is a horn. It also has a number of doors.
   d. A Bicycle is a Wheeled Vehicle with two wheels, and its warning system is a bell.

3. Using the following code as the main method of your program, write C++ classes that work according to the description in Q2 (and the following code):

```
a. int main() {
b.   //Create a car with 5 passengers and 4 doors
c.   Car* car = new Car(5,4);
d.   cout << "The car has " << car->getNumDoors();
e.   cout << " doors\n";
f.   //Create a bicycle built for two
g.   Bicycle* bike = new Bicycle(2);
h.   //Combine vehicles into one array
i.   Vehicle* vehicles[] = {car, bike};
j.   for (int i=0; i<2; i++) {
k.     cout << "Vehicle " << i << " has warning sound: ";
l.     cout << vehicles[i]->warning() << endl;
m.   }
n.   return EXIT_SUCCESS;
o. }
```

4. Convert the C++ code from Question 3 to Java. Also write an equivalent `main()` function in Java.

5. Think about how a software solution involving the set of classes in Question 2 to Question 4 might need to be extended in the future.

6. Assume that you have implemented code in C++ that follows the list specification in Question 1. Adapt this class to behave like a Queue. Go through this adaptation process twice – once using traditional C++ composition, and again using private inheritance.

7. Rewrite your List and Node classes of question 1, so that the Node class is a "private inner class" within the list class. What are the advantages of using this class structure.