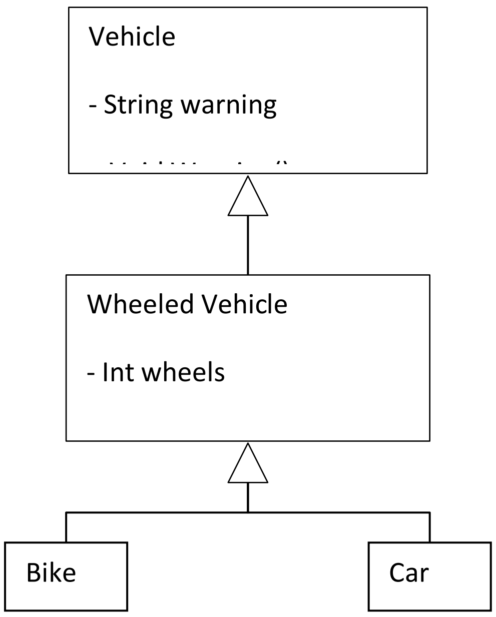
SENG2200 Week 4 Solutions

1. Write a doubly linked list, two sentinels (Head and Tail), insertion before current:

```
insertAfterCurrent(Object O)
{
        Node newNode = new Node(O);
        newNode.setNext(  current.getNext());
        newNode.setPrev (current );
        current.setNext (newNode);
        current.getNext.setPrev ( newNode );

}

insertAtHead(Object O)
{
        current = sentHead;
        insertAfterCurrent(O);
}

insertAtTail(Object O)
{
        current = sentTail.getPrev();
        insertAfterCurrent(O);
}
```

   Standard data, Node class, getters, setters and current manipulators

2. Vehicle question – *I hope this is what you were looking for*



3.

```
Class Vehicle

{

  Public:

           Void Warning();

 protected:

           String warning;

}


Class WheeledVehicle : public Vehicle

{

protected:

Int wheels;

}


Class Bike : public WheeledVehicle

{

          Public Bike(int numberWheels);

}

Bike::Bike(int numberWheels)

{

Wheels = numberWheels;

Warning = “Bell”;

}

Class Car : public WheeledVehicle

{
```

```
Public:
        Car(int numberWheels);
Private:
        Int passengers;
}
Car::Car(int numberPassengers, int numberWheels)
{
Wheels = numberWheels;
Warning = "Horn";
Passengers = numberPassagngers;
}
```

4.

Same, but:

Public class Bike extends WheeledVehicle

One of the classes will also have to have the public static void main()

5.

More vehicles, of different types (not just wheeled, perhaps 'plane' and 'boat')

Other functions (passengers, drive, fuel, repair, etc)

6.

a) Class queue contains a private LL called lovalvar, its interface extends as required and just forwards the call on (eg 'enqueue' calls 'localvar.addToTail')
b) Inherits the LL privately. Thus has access to the functions (will be private, so not able to be accessed externally). Implement public methods that call those functions. Enqueue() now just directly call addToTail()

```
Class queue : private LinkedList
{…
        Void enque(object) { addToHead(object); }
```

}

7.

Public class LinkedList

{

…

Private class Node // private class should be internal to LL class

// normally will have a private constructor.

{…}

}

Advantages are that Node can't be interfered with or accessed by other classes, preserving the idea of encapsulation and information hiding.

Disadvantages is you can't reuse Node, such as for a 'single linked list' implementation (not a huge issue here as all other similar classes such as stack will use LL)

Side note on inheritance in c++:

```
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A
{
    // x is public
    // y is protected
    // z is not accessible from B
};

class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};
```

```cpp
class D : private A
{
    // x is private
    // y is private
    // z is not accessible from D
};
```