CSci 3081W: Program Design and Development
Homework 2 – UML Basics and Designs
Fall 2022

Due Dates:

- **Full Paper -** Sunday, October 23rd, 2022, before 11:59pm.

Overview: This homework consists of two parts. The first part will be similar to homework 1 in that you will be writing about UML and how it works (like a guide). For part 2, you will be given some different scenarios and you will have to draw a UML class diagram in order to solve these designs. Make sure to start early, you won't be able to complete this assignment if you start it the day that it's due.

Part 1: (25 points)
UML (Unified Modeling Language) encompasses many different aspects of documenting artifacts of software systems. In this course, we do not use all of these aspects nor will we learn them. Here are all the UML diagrams: Class diagram, Object diagram, Use case diagram, Sequence diagram, Collaboration diagram, Activity diagram, Statechart diagram, Deployment diagram, and Component diagram. We will only be doing Class Diagrams.

UML's building blocks are broken down into "things". Things can be structural, behavioral, grouping, annotational, or relationships. We will not cover behavioral or grouping things. Within each of the other things, we may also omit subthings.

Structural things: Class, Interface, Collaboration, Use Case, Component, Node.
We will only focus on Classes.

Annotational things: Annotation
We will focus on Annotations.

Relationships: Dependency, Association, Realization/Implementation, Generalization/Inheritance, Aggregation, and Composition
We will focus on all of these relationships. (These are also the arrows from workshop 2.)

For part 1, write at least one page describing and explaining how to use each of the following aspects of UML: Classes, Annotations, and all Relationships. Make sure to completely describe all aspects of each! Provide examples and diagrams/pictures as necessary. Cite your sources.

Part 2: (75 points, 25 pts per subproblem)
For each of the situations listed below, draw a UML Class Diagram using Lucid Chart to design the system.

A. Ducks

In this system, we will represent a small ecosystem. In this ecosystem, we have primarily different types of ducks. The environment is a pond. The pond has many attributes including depth, water clarity, surface area, surface temperature, permanent vs. temporary, as well as a vector that holds all the ducks in the pond. (Hint: think of the data type that best suits each of these variables.) The pond will only have two functions: add_duck and remove_duck.

Regarding the ducks, it will be very similar to the previous workshops and labs that you have worked on that had ducks in them. We will have different types of ducks all of which can reside in this pond: Mallard, Redhead, Marbled, Alabio, and Canvasback.

The ducks have many functions: display, swim, quack, fly, land, migrate_in, and migrate_out. The ducks also have attributes: color, pos_x, and pos_y. There should be getters for each of the attributes. Regarding setting the attributes, this should be done in the constructor.

When a duck displays, it'll just print out the type of duck that it is.

For swim, it'll take in two coordinates and then swim to that position.

For quack, it'll just print quack. All ducks have the same quack function. (Hint: does the base quack need to be virtual then?)

For fly, it'll take two coordinates then hover in the air. For land, if the duck is flying, then it'll land back in the pond at those coordinates. (Hint: use a boolean variable to indicate whether the duck is flying or not.)

For migrate_in, the duck will be added to the pond. For migrate_out, the duck will be deleted and removed from the pond. (Hint: remember the big 3)

Remember that you're just making a UML class diagram. You are not coding this! Make use of annotations in your UML class diagram to cover some of the non

obvious aspects of your system.

B. Vehicles

In this problem, we will consider the relationships and properties of **vehicles**. All vehicles should have the following properties:

- Color (“blue”, “red”, “pink”, etc.)
- Manufacturer (“Yamaha”, “Ford”, etc.)
- Year built (2011, etc.)
- Velocity (20.3, 56.9, etc.)
- Num Wheels (2, 4, etc.)

When a vehicle is built (i.e. instantiated with a constructor), all 5 properties should be set. The manufacturer, year built, and number of wheels cannot be modified afterwards. However, Color, and Velocity can be changed. Ensure you have functions that can return the value of all of these properties, and functions that only modify Color and Velocity.

The types of vehicles in this problem can be separated into two categories: motorized and non-motorized vehicles.

Motorized vehicles have a fuel level (0.0-1.0), and engine horsepower (an integer) associated with them upon creation, and two additional methods, one to turn on, and the other to run off the vehicle. The state of the vehicle (on/off) should be stored in the class. Motorized vehicles also have an alphanumeric VIN number that is assigned to them upon the creation of the vehicle and cannot be changed after the fact. You can refill the fuel of motorized vehicles and change the engine horsepower, so you should have functions to modify these properties.

Non-motorized vehicles must be secured with a chain or U lock when they’re not being used, which means they must have a lock and unlock function. The state of the vehicle (locked/unlocked) should be stored in the class.

The following six vehicles will be considered in this problem:

- Motorcycle
- Bicycle
- Scooter (non-electric)
- Tricycle
- Truck
- Car

All classes must have functions to access each property, but not all properties should be able to be modified. Therefore, all member variables should be private, and only some should have functions to set the variables to a new value.

Create the UML class diagram that relates vehicles, motorized vehicles, non-motorized vehicles, and the six vehicles considered in this problem. Make sure to encapsulate all of the properties and functions specified above.

C. **Drone/Uber Simulation System**

In this problem, you will be drawing the UML diagram of the Drone Simulation System project. We will not be giving the problem description but the actual code itself. From there, you will have to create the UML class diagram which incorporates the project system design itself.

Important: Do not create the class diagrams for the classes under

- apps/graph_viewer
- libs/routing

The project codes can be found in the lab05 repo.
Note: This lab will only open starting from Oct 7
https://github.umn.edu/umn-csci-3081-F22/public-lab05