A Beginner's Guide to UMLs

What is a UML?

Unified Modeling Languge (UML) is a general purpose modeling language designed to provide a standard way to visualize the design of a system.
In other words, a UML is a graphical model that can be used to visualize a software's design. In either how it behaves or how it's structured. In your preliminary project report, you'll be using a type of **structural UML**.

What's the Purpose of a UML?

- To help communicate some aspect of a system and to better undestand it in a graphical view.
- To help us build and refactor an application.
  - Conveys ***what and how to build***
- Emphasizes **classes**, **attributes**, **operations**, and **relationships** that are of interest.
- Used to describe a detailed design for a programmer to follow in writing source code.

Class UML Diagrams

There are many different types of UML diagrams stemming from two families. However, to remain in the scope of this class you'll only need to worry about Class UMLs.

> A Class UML is a structural diagram, meaing that you are showing how you are **implementing** and **creating objects**.

Class UMLs are structure diagrams that are fundamental to OOP solutions. It shows the classes in a program, their attributes (public and private members), operations (methods/functions), annd the relationships between classes.

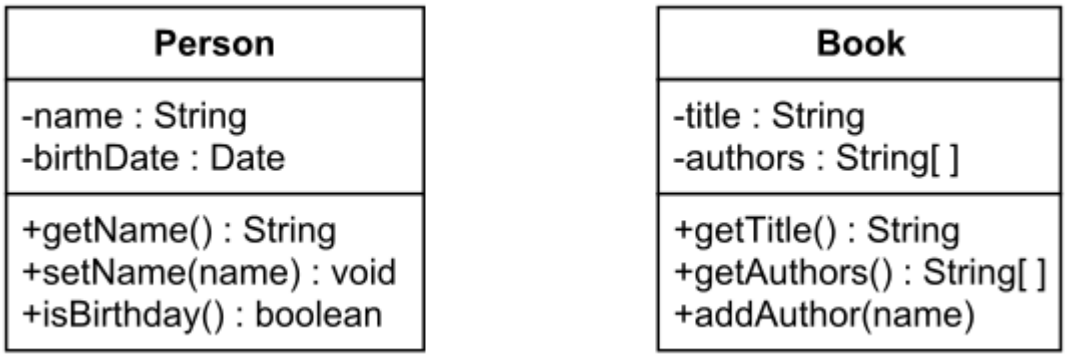
There are three parts to a class UML:

1. The name of the class is listed at the top.
2. The attributes are found in the middle.
3. The methods (functions) are found at the bottom.

> Make sure you take note of what data type each attribute and the return type of every method listed in your diagram. This comes after a colon and the name of the attribute/method.

> Use a `+` sign to show that a member is public and a `-` sign to show that a member is private. For example, a private integer called `customerId` would be shown in your UML as an attributed listed as `- customerId: int`

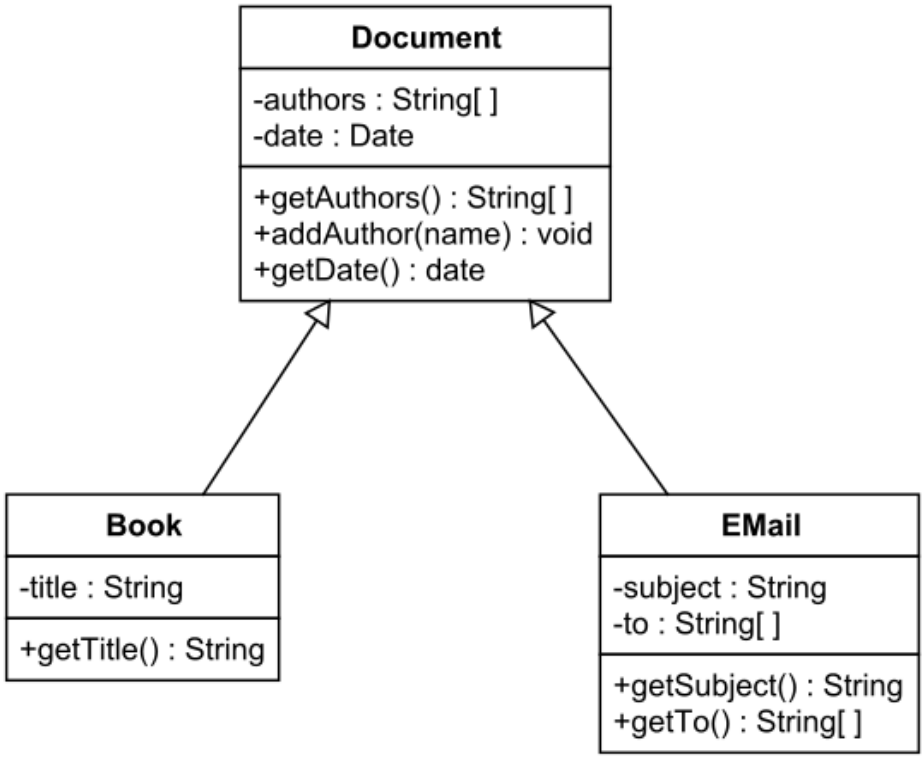
Example:



Representing Inheritance Relationships

Use arrows between classes with inheritance. The child (derived) class will point to the parent class.

Example:



In the example above, the `Document` class is inherited by the `Book` and `EMail` classes. Hence, the arrows pointing from the derived clsases to the inherited class.

> If your class is abstract, meaning it has virtual functions, please **italicize** the name of that class. As well for each of its pure virtual function.

Inheritance vs Composition

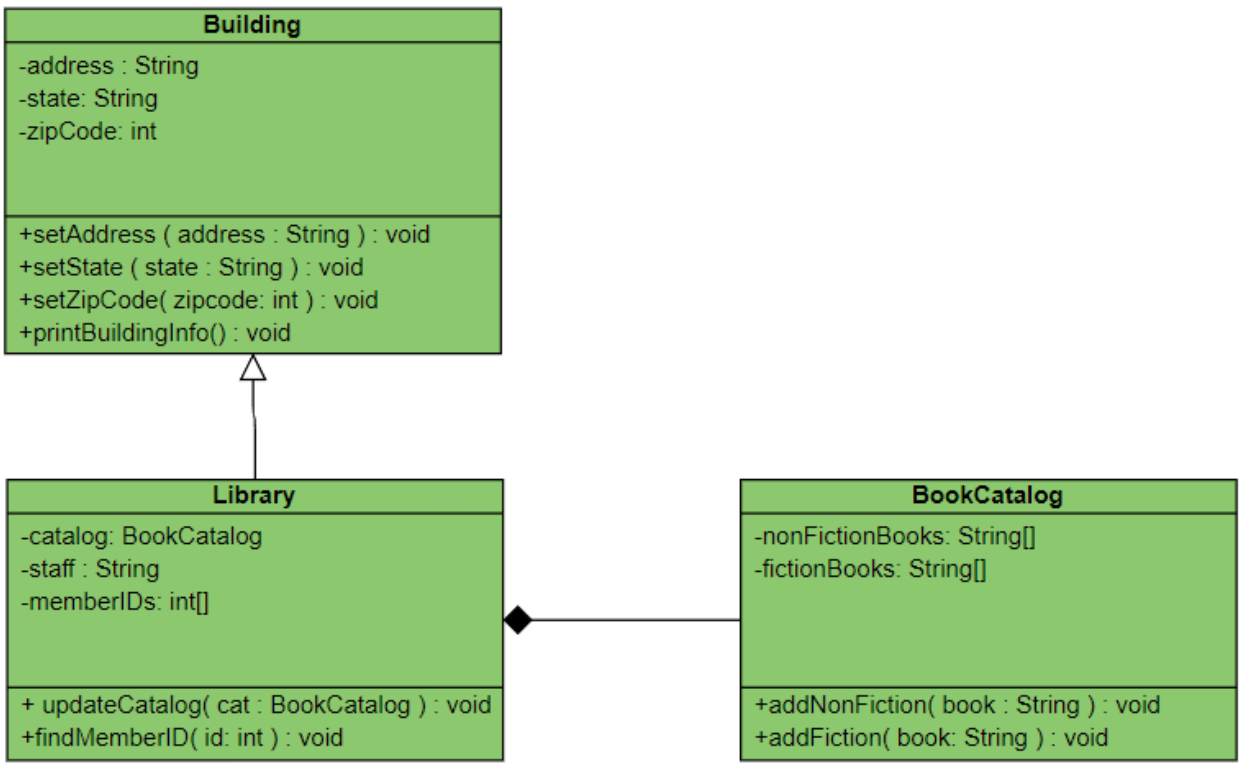
Not all relationships between classes are based on inheritance. Some are based on composition.

Composition can be described as a ***HAS-A*** relationship. Meaning one class has an instance of another.
While Inheritance can be described as a ***IS-A*** relationship. Where one class is derived from another as one of its member variables.

> Neither **HAS-A** or **IS-A** are acronyms at all, they are simply read as "has a(n)" and "is a(n)".

An example of this could be the relationship between a `Building` class, a `Library` class, and a `BookCatalog` class.

- The `Library` class has an **IS-A** relationship with the `Building` class, hence it inherits from the `Building` class. (Inheritance)
- The `Library` class has a **HAS-A** relationship with the `BookCatalog` class. As in the `Library` class there will be a `BookCatalog` object as a variable.(Composition)
  A Class UML for this program could look something like this:



> An arrow with a solid line with a solid diamond is used to show composition. While a solid line with an unfilled arrowhead indicates a class derived from another class. (Inheritance)

How to start?

Well, to draw UML diagrams I recommend you find a UML diagram maker such as Canva, LucidCharts, VisualParadigm, or which ever one best suits you.
The bottom line is though, to start your preliminary report by thinking of **your** OOP solution and how you'll structure your program. Once you have a solid idea of what you want to do, follow the UML conventions and create your diagram. I wish you all the best of luck!