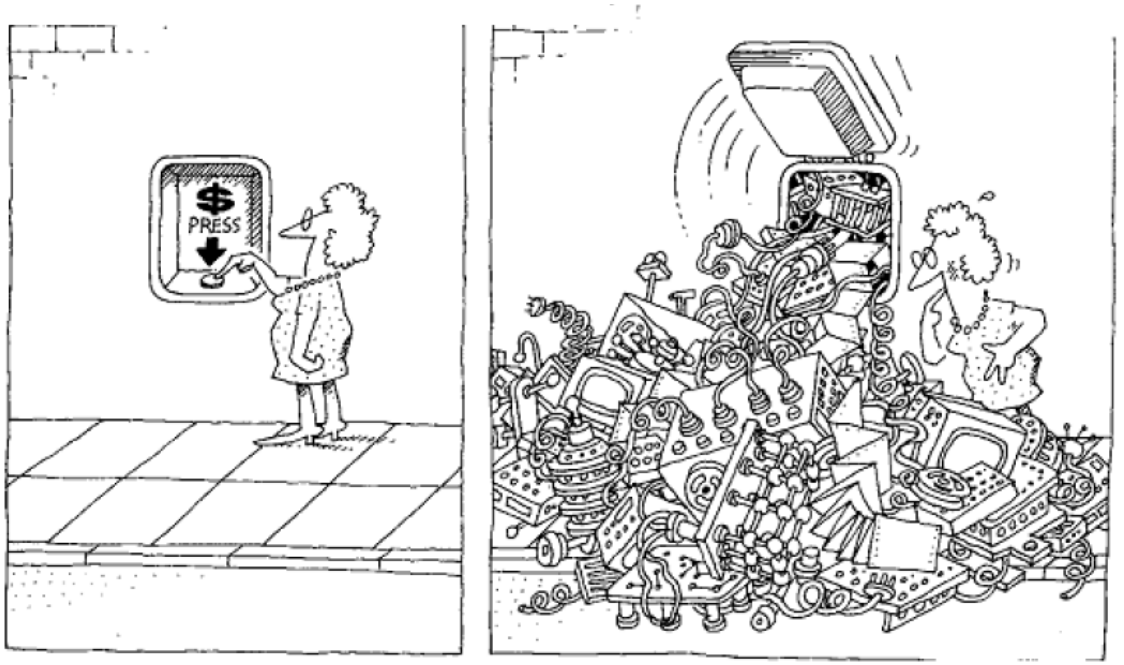
OOPs fundamentals and UML basics

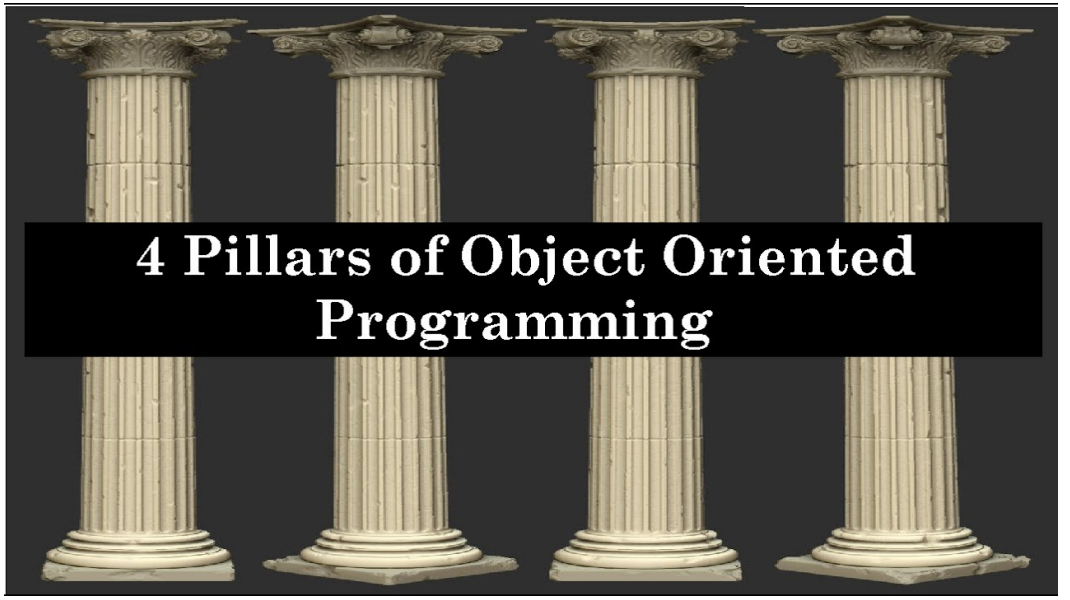
Dealing with Software complicity



The task of the software development team is to engineer the illusion of simplicity.

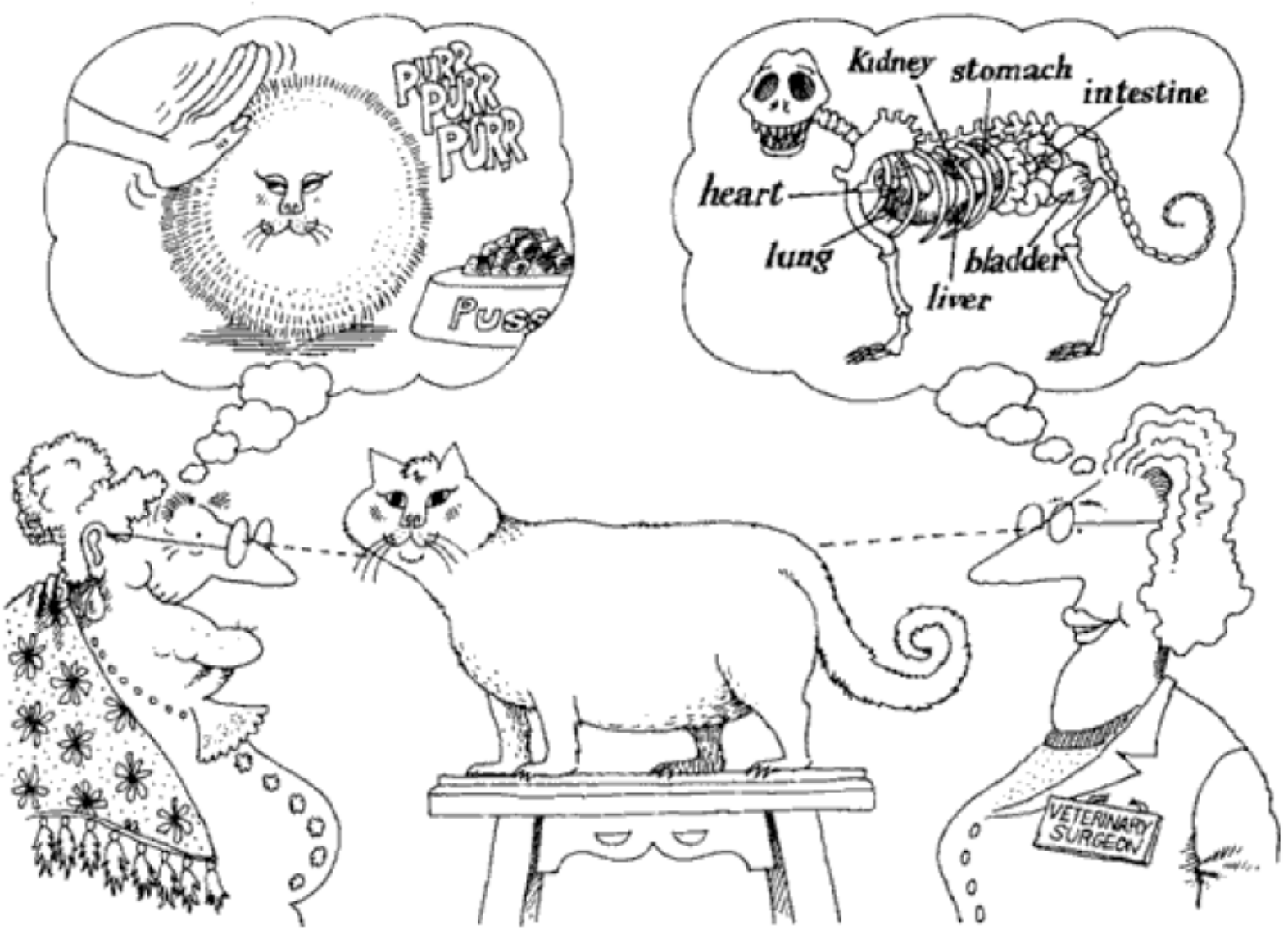
Pillors of OO

- Abstraction
- Encapsulation
- Modularity
- Hierarchy



Abstraction

- Fundamental ways that we use to cope with complexity
- "abstraction arises from a recognition of similarities between certain objects, situations, or processes in the real world, and the decision to concentrate upon these similarities and to ignore for the time being the differences" -Hoare
- An abstraction denotes the **essential characteristics of an object that distinguish it from all other kinds of objects** and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.



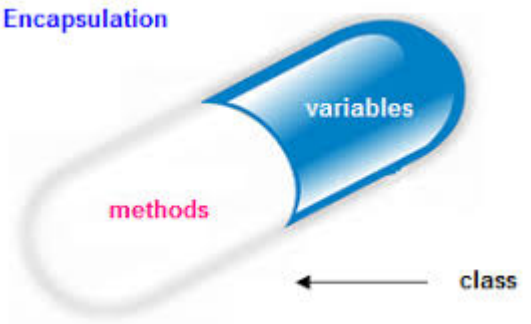
Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

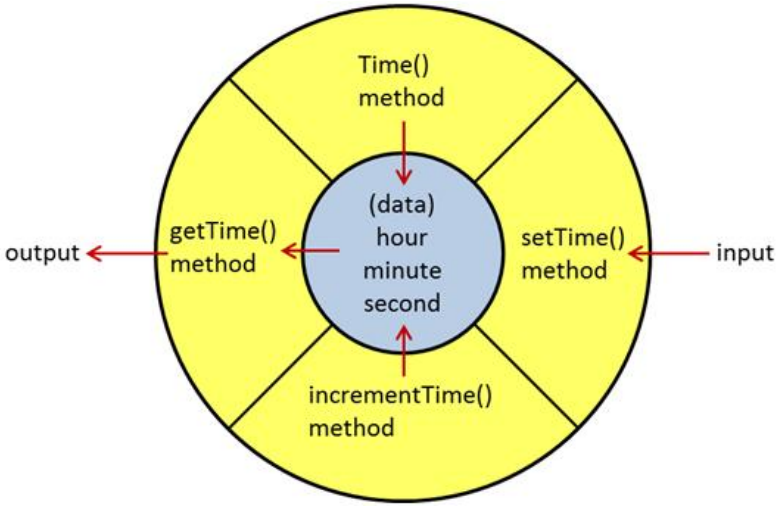
Encapsulation

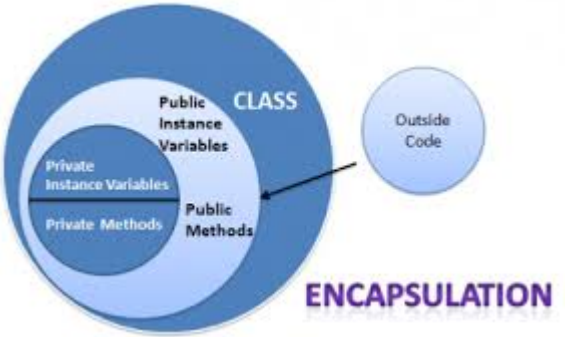
- Abstraction and encapsulation are *complementary concepts*: abstraction focuses upon the observable behavior of an object, whereas encapsulation focuses upon the implementation that gives rise to this behavior.
- Encapsulation is most often achieved through information hiding which is the process of hiding all the secrets of an object that do not contribute to its essentialcharacteristics; typically, the structure of an object is hidden, as well as the ,implementation of its methods.
- Information hiding is tool to achive encapsulation

Understanding encapsulation

```
class Hacker{
Account a= new Account ();
a.account_balance= -100;
}
```

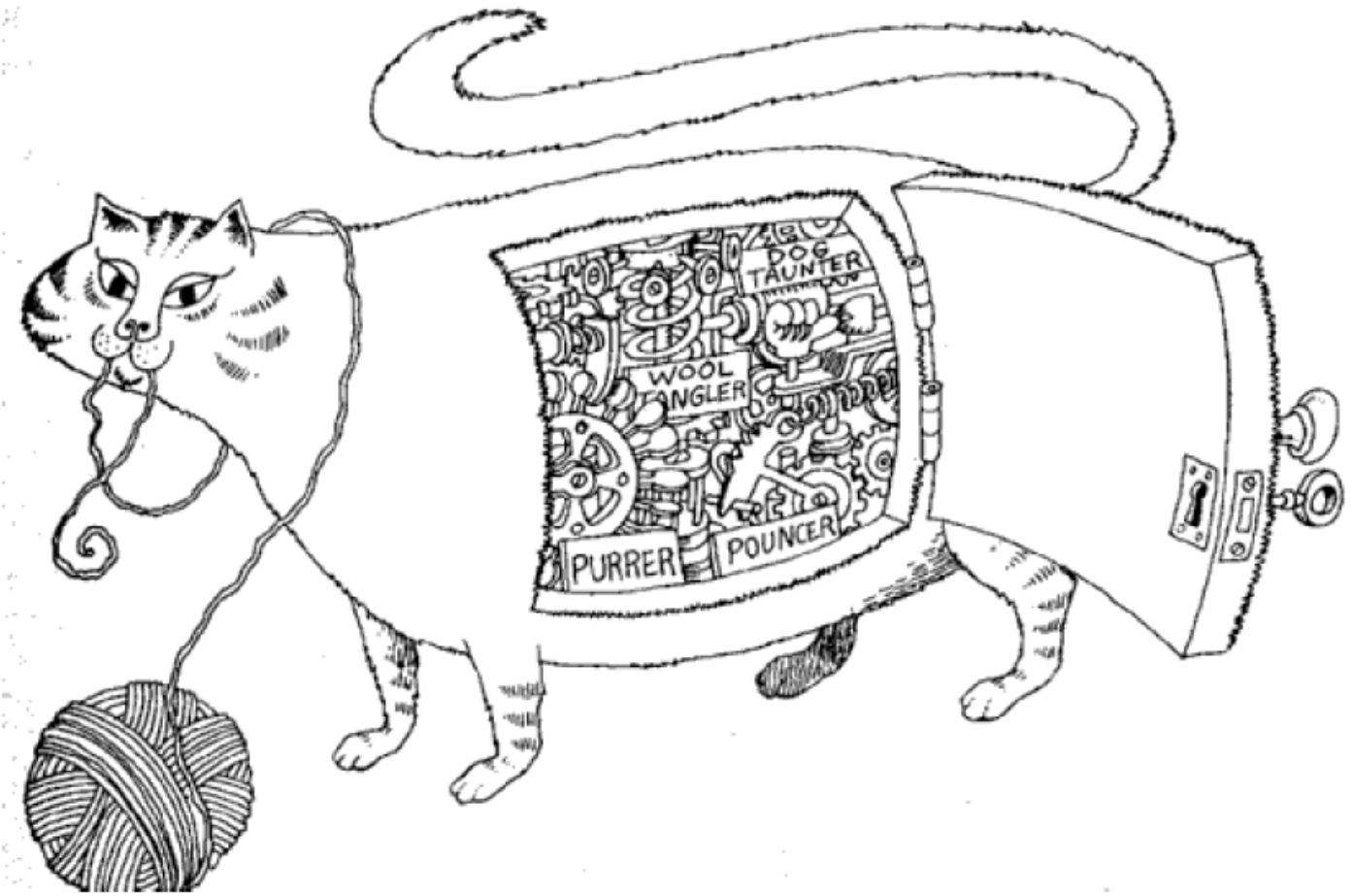






Encapsulation

- Changing data in organized way, by using data hiding and appying business constraints
- Encapsulation= data hiding + constraints



Encapsulation hides the details of the implementation of an object.

Encapsulation

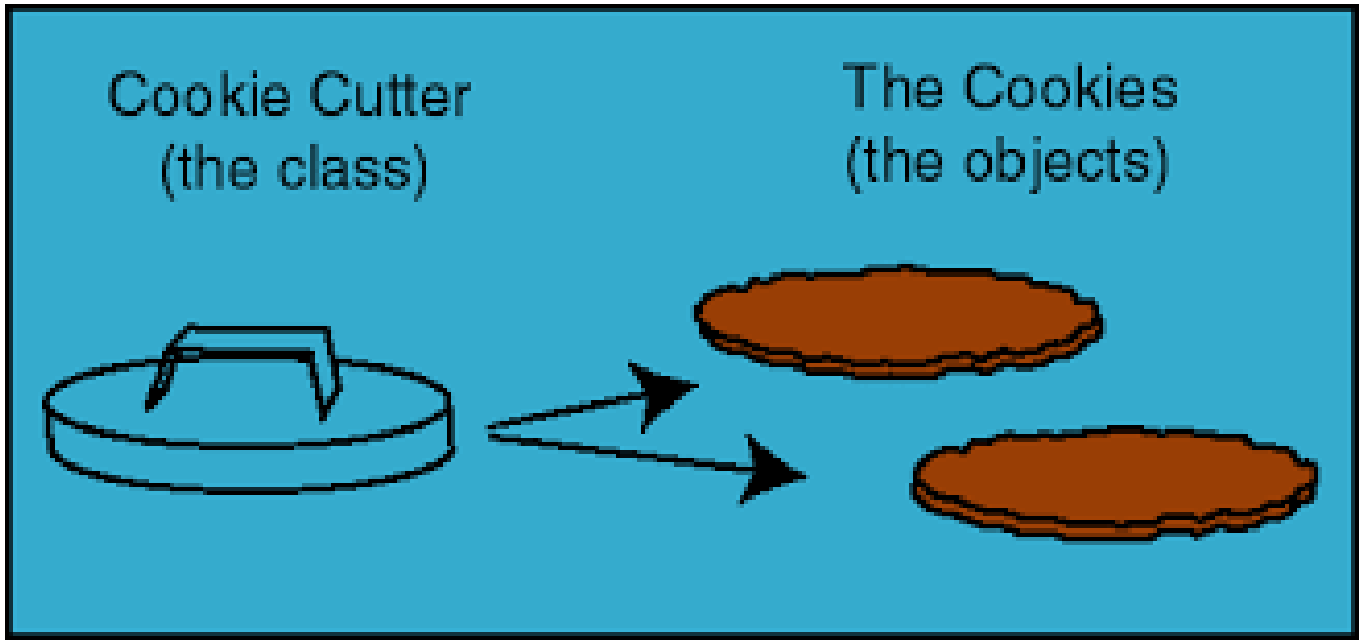
- "for abstraction to work, implementations must be encapsulated"- Liskov.
- **In practice, this means that each class must have two parts:**
  1. **an interface and**
  2. **an implementation**.
- **The interface of a class**
  - captures only its outside view
  - encompassing our abstraction of the behavior common to all instances of the class.
- **The implementation of a class**
  - comprises the representation of the abstraction as well as the mechanisms that achieve the desired behavior.

Modularity

Modularity packages abstractions into discrete units.

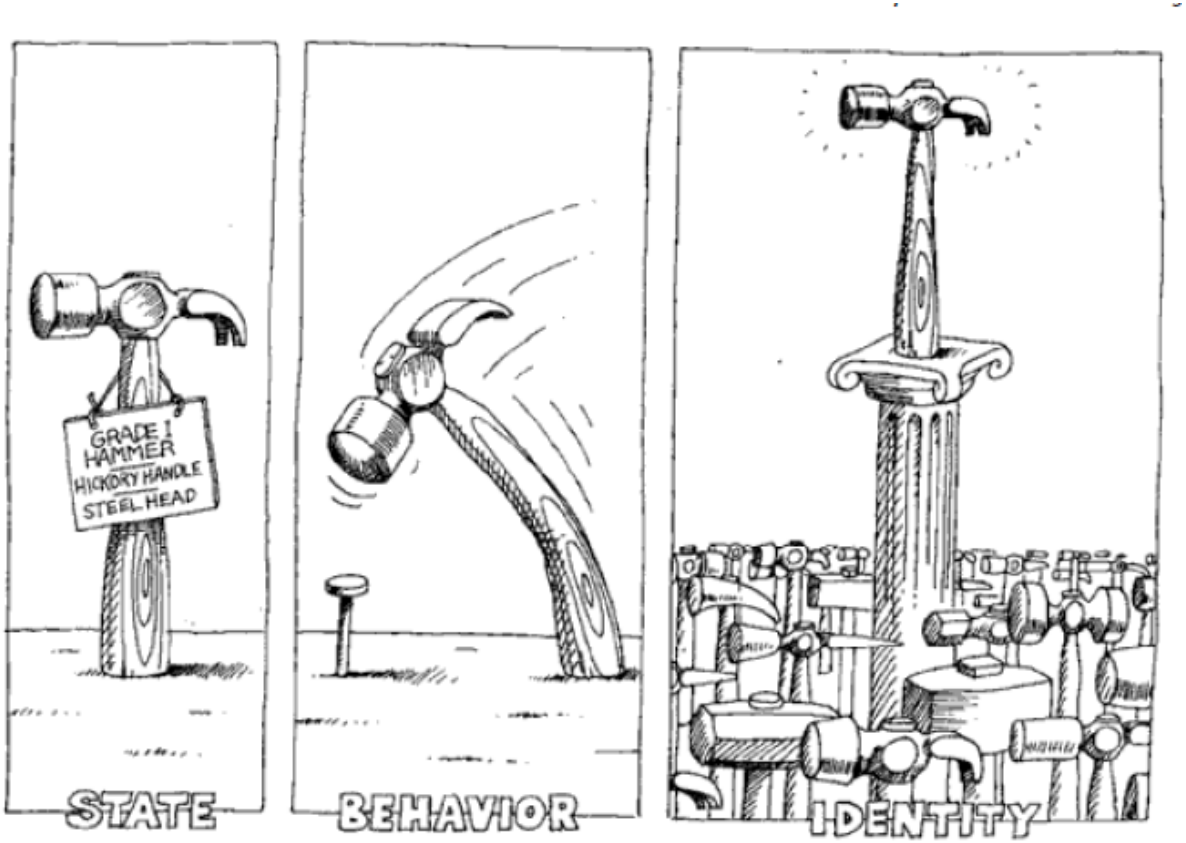
Hierarchy

Class vs Object



Objects

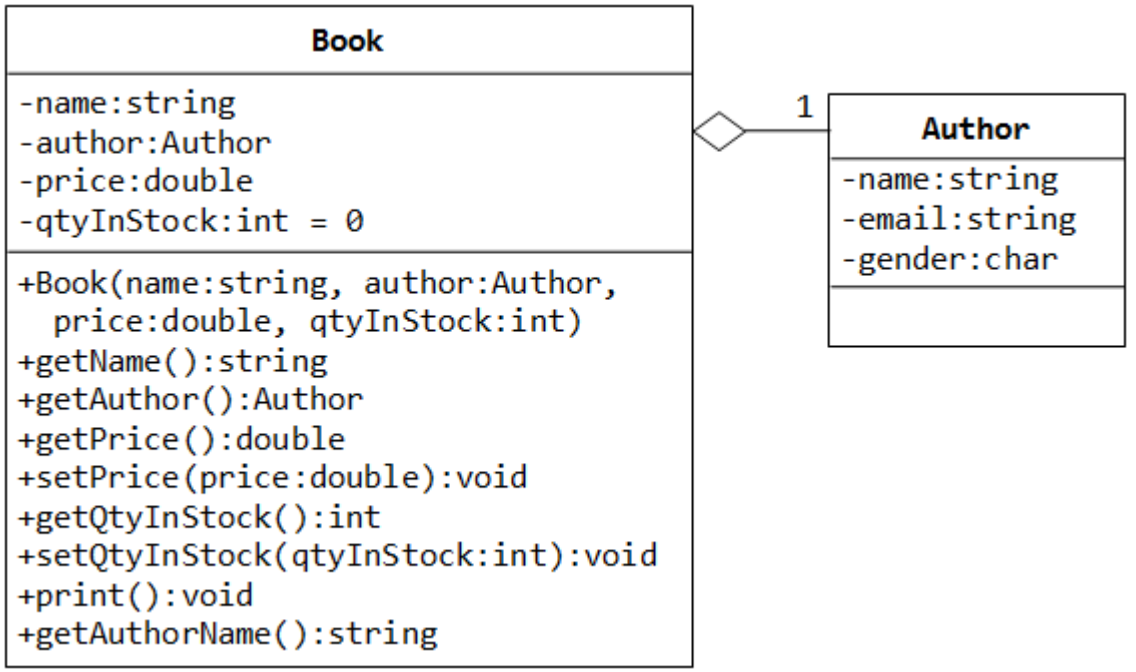
- Every object have 3 things
  - Identity
  - State
  - behavior



An object has state, exhibits some weii-defined behavior, and has a unique identity.

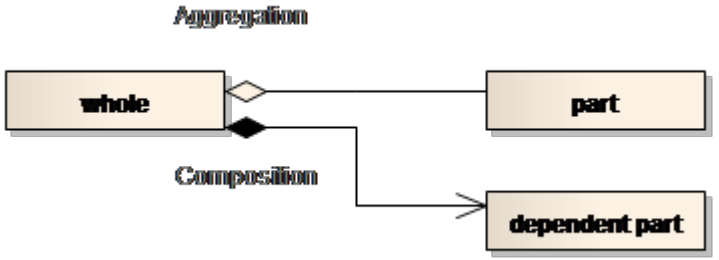
An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class; the terms instance and object are interchangeable.

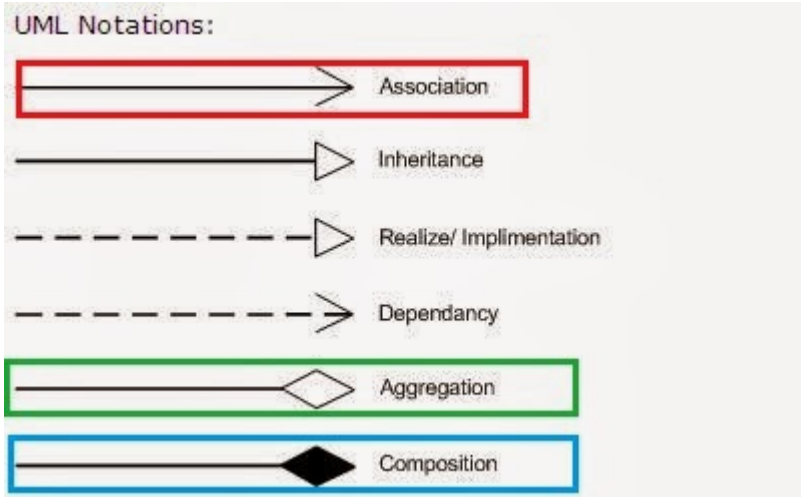
Class Diagram



Relationship between Objects

- USE-A
  - Passanger using metro to reach from office from home
- HAS-A (Association)
  - Compostion
    - Flat is part of Durga apartment
  - Aggregation
    - Ram is musician with RockStart musics group
- IS-A
  - Empoloyee is a person





UML Class Diagram