**Case study:**

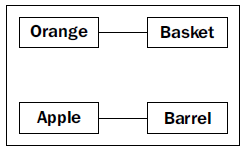
**Remind what is UML diagram**

**UML:**

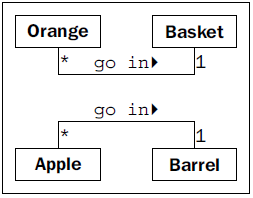
* **Acronym for Unified Modeling Language**
  + **What:** The diagram below describe **relationships between** each class we have
    - **For example: *Orange*** have a association with ***Basket,*** *and vice versa;*

Same goes for **Apple** and **Barrel**

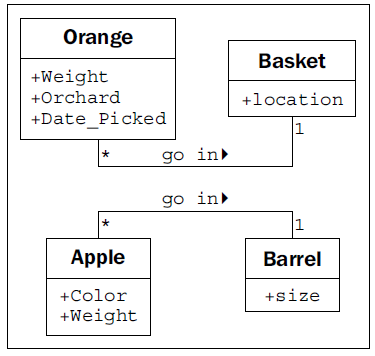
* + **Why:** 
    - Very intuitively, easy to imagine
  + **How:**

****

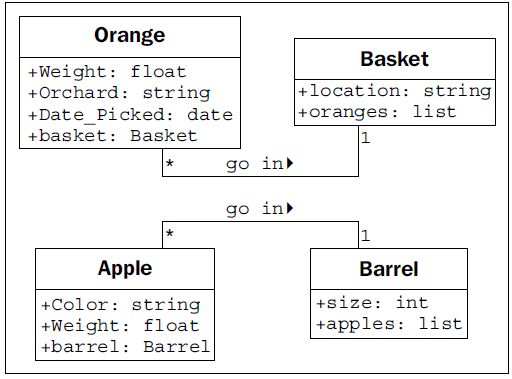
* The diagram above only shows **"there is an association between each class",** it doesn't show how many oranges can go into **1** basket; or how many baskets can go into **1** orange.

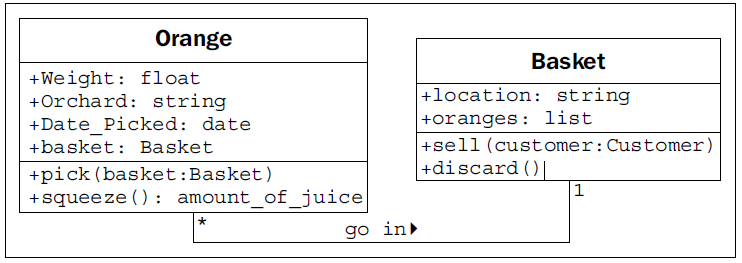
****

* However, we can specify this by quickly add additional information such as 1:n or 1:1
* go in: "A" can in to "B"; and many "A" (represents by \*) can
  + In regrex \* - means "0 or more instances of the preceding regex token"



This is additional attributes or properties … etc, (obviously), however, to represent the "1:n" relationship above, *Basket* and *Barrel* need one more attribute of which the functionality is to keep track of how many and what oranges it stored and vice versa,

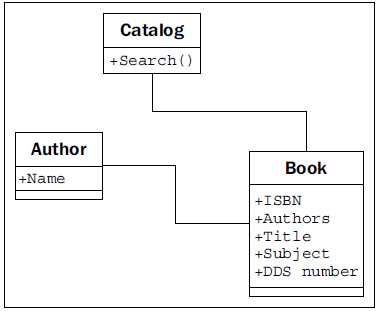




**Case study:**

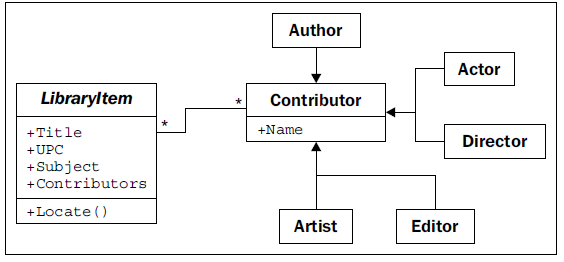
Catalogs contain lists of books. People search them to find books on certain subjects, with specific titles, or by a particular author.

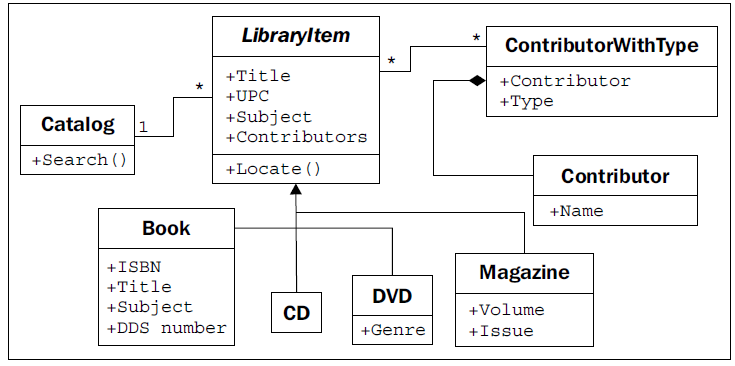
* Books can be uniquely identified by an International Standard Book Number (ISBN).
* Each book has a Dewey Decimal System (DDS) number assigned to help find it on a particular shelf.
* Book as the most important object, with several attributes already mentioned,
  + author;
  + title;
  + subject;
  + ISBN;
  + DDS;



One option is to add attributes to each of our LibraryItem **subclasses**:

* That hold the information we need, such as *Author on Book*, *Artist on CD (optional);*
* Make the relationship to those properties all point to the Contributor class.





**Your exercises** will be using inheritance, access specifiers (private , protected and public) in python to design these class with the following attributes and method

Requirements: have docstring in every class and every methods of that class

Catalog:

search → public

search:

* For example, query a book of a title "Angels and Demons" (by Dan Brown), we want the DDS number (which is the shelf number of which we can use to identify in that library). Then LibraryItem will *locate* for that Book to which DSS number,
* The DVD is located by returning the genre and title of the DVD. The user can then visit the DVD section, find the section containing that genre, and find the specific DVD as sorted by the titles.

LibraryItem:

Locate → public

to Locate a book, we return the DDS number

Title, UPC, Subject, Contributors → public attributes

ContributorWithType:

Contributor → public

Type → protected

Book:

ISBN, Title, Subject, DDS number → public

CD:

Artists → public

DVD:

Genre, Actors, Directors …→ public

Magazine:

Volume, Issue , Editors