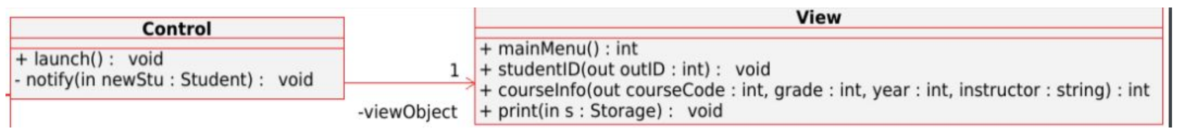
<u>Multiplicity :</u>

[1..],[0..*],[0],[2],[etc...]*

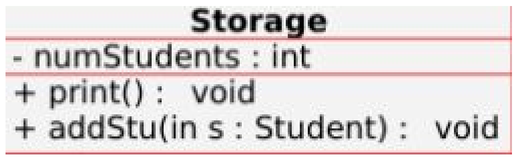
- Used to show that X has (n) Y objects.
- Star represents ‘many’ objects.
- In the case below Control has 1 View object


Control
+ launch() : void
- notify(in newStu : Student) : void
1
-viewObject
View
+ mainMenu() : int
+ studentID(out outID : int) : void
+ courseInfo(out courseCode : int, grade : int, year : int, instructor : string) : int
+ print(in s : Storage) : void

<u>Member Access:</u>

[#],[-],[+]

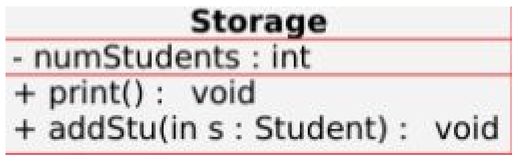
- (#) is protected, (-) is private, (+) is public.
- Must be used when specifying any data member or member function.
- In the case below, numStudents is private, and the two member functions are public.



<u>Data Members:</u>

[Access Specifier] [Name] : [Data Type]

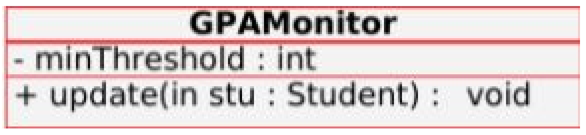
- In the case below the access specifier is private, numStudents is the name, and the type of the data is integer.



<u>Member Functions:</u>

[Access Specifier] [Function Name]([Input/output Type] [Name]:[Data Type]) : [Return Type]

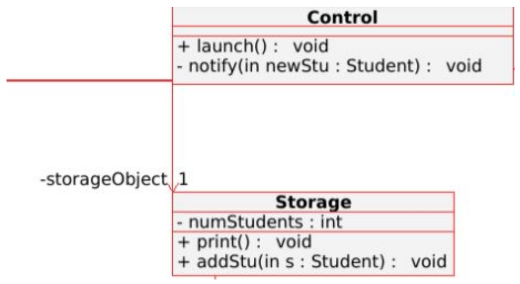
- (In) denotes that data is read into the function through a parameter reference.
- (Out) denotes that data is sent back to the parameter reference.
- (InOut) denotes that data is read in and sent back through the parameter reference.
- In/Out/InOut Specific to each parameter.
- The return type specifies the functions return type.
- In the case below, the public function update has a parameter named stu, which is of type Student, and which sends data by reference to the function. The return type of the function is void.



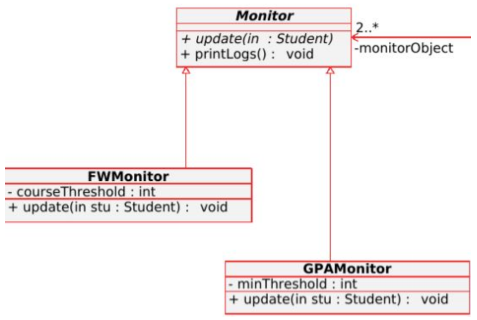
Relationships (Association) (Arrows):

[X->Y], [X-Y], [X- -Y], [X□Y]

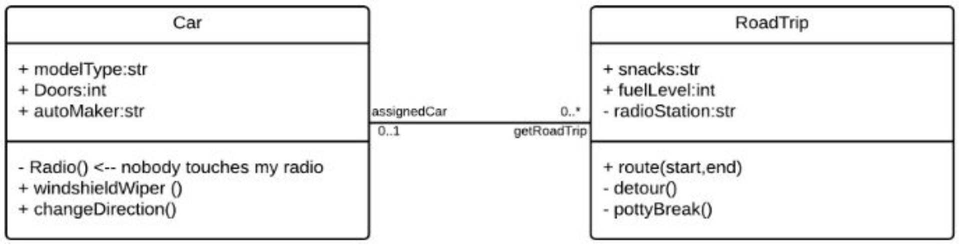
- Specifies the relationship between two objects
- The image below shows a 'Has-A' relationship between Control and Storage. With an Open arrow. Control 'Has-A' storage object.



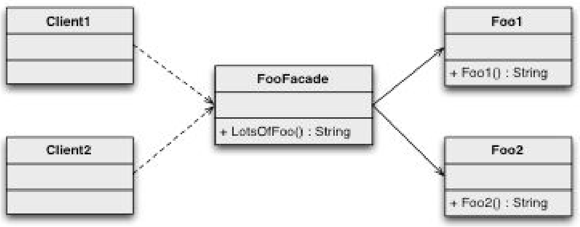
- The image below shows a 'Is-A' relationship between Monitor and the FWMonitor and GPAMonitor objects. With a closed arrow. FWMonitor and GPAMoritor 'Is-A' Monitor object. (They are derived from the base class).



- The image below shows a bidirectional relationship, which is denoted with a straight line. This relationship specifies that each object has at least one instance of the other. Car has an instance of RoadTrip and RoadTrip has an instance of Car.



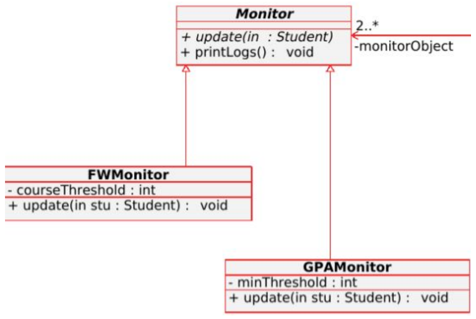
- The Image below shows a implementation relationship. The line for this relationship is dotted, as that specifies that both clients have a Facade object, where the code behind the Facade object's functions is not known.



Polymorphism (Pure Virtual Class/Function) (Abstract):

[Cass Name] or [Member Function] (ITALICIZED)

- In an abstract class, the class name is written in Italics.
- The pure virtual function is also italicized but the concrete instance is written normally.
- In the image below Monitor is a Abstract class with pure virtual function update. Update is then made concrete in GPA Monitor and FW Monitor.



Things To Not Include:

- Collection Classes or references to a collections object UNLESS ITS A VECTOR (from the STL Library).
- [*],[&] Reference Symbols. Instead use In/Out/InOut
- Getters/Setters, Constructor/Destructor, Friendship.
- Other objects as attributes, should instead show the relationship.