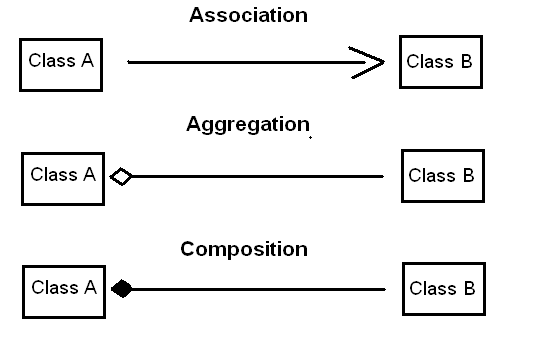
# **DEFINITIONS**

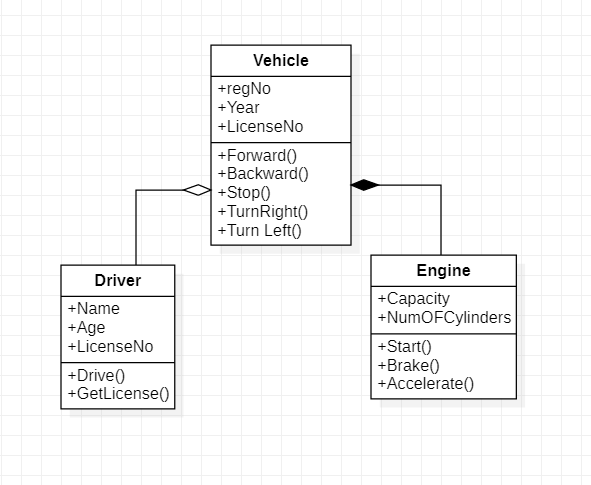
* **Association**: It is the relationship between 2 objects. Aggregation and Composition are the two forms of association.
* **Aggregation**: In aggregation, the child can independently exist even if the parent has been destroyed. For example, there’s a parent class school which has a child class student. Even if the school (parent) gets destroyed, the child still exist. It is symbolically represented by an empty diamond.
* **Composition**: In composition, one class owns another class (parent-child). However, the child class cannot exist once its parent has been destroyed. For example, class owns different parts such as engines, tires, and wheels. The different parts will not be able to function is the car is destroyed. It is symbolically represented by a filled diamond.

**Benefit of aggregation and Composition:**

Composition is considered to be flexible as compared to inheritance. Moreover, “has a” relationship is considered to be semantically correct compared to “is a” relationship.



# **CLASS DIAGRAM**



# **PYTHON CODE**

class Driver:  
 def \_\_init\_\_(self):  
 self.Name = None  
 self.Age = None  
 self.LicenseNo = None  
  
 def Drive(self, ):  
 pass  
  
 def GetLicense(self, ):  
 pass

class Engine:  
 def \_\_init\_\_(self):  
 self.Capacity = None  
 self.NumOFCylinders = None  
  
 def Start(self, ):  
 pass  
  
 def Brake(self, ):  
 pass  
  
 def Accelerate(self, ):  
 pass

class Vehicle:  
 def \_\_init\_\_(self):  
 self.regNo = None  
 self.Year = None  
 self.LicenseNo = None  
  
 def Forward(self, ):  
 pass  
  
 def Backward(self, ):  
 pass  
  
 def Stop(self, ):  
 pass  
  
 def TurnRight(self, ):  
 pass  
  
 def Turn Left(self, ):  
 pass