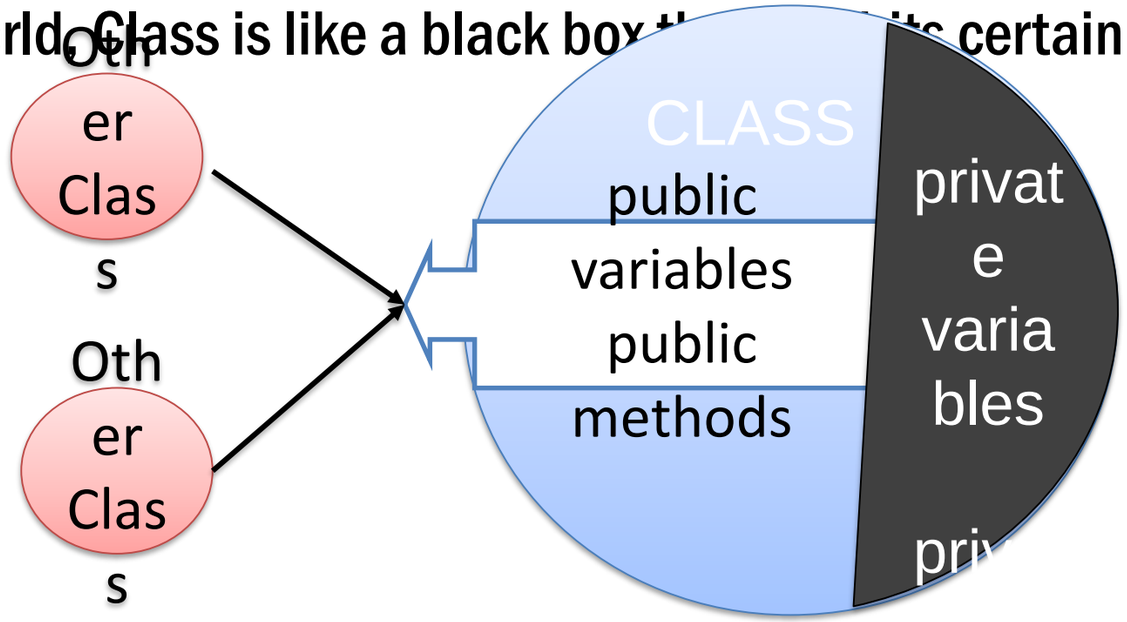
Implementing Encapsulation

UNDERSTANDING ENCAPSULATION

- Is the concept of hiding most of the data and internal functionality and exposing essential interfaces for interacting with the object
- Class is like a capsule which encapsulates methods and data to provide intended functionality
- To external world, Class is like a black box that exhibits certain behavior



IMPLEMENTING ENCAPSULATION

- To Implement Encapsulation
  - Protect the instance variables with private access modifier
  - Provide public getter and setter methods
  - Any method which is internal functionality of class must be made private

- Advantages
  - improves maintainability, flexibility and reusability
  - Fields can be made read-only
  - Helps in providing simple interfaces to other classes in the application

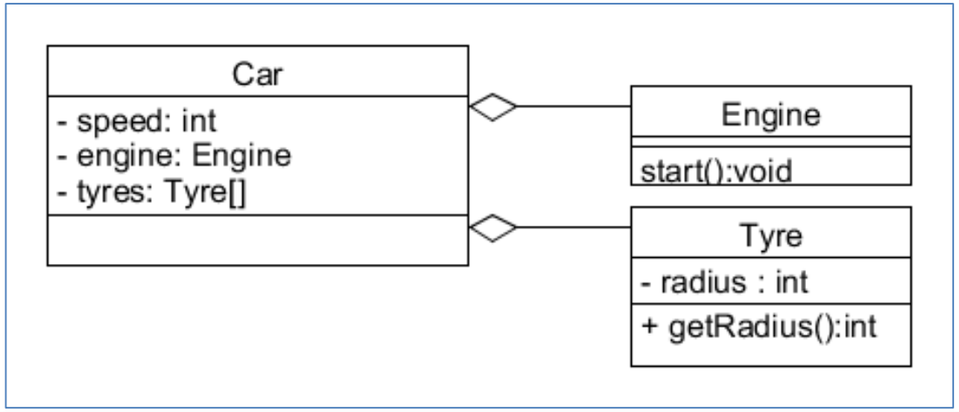
```
public class Employee {
 private String empName;

 //getter and setter (accessor and mutator)
 public String getEmpName() {
    return empName;
 }
 public void setEmpName(String
```

Implementing Aggregation

AGGREGATION AND COMPOSITION

- Aggregation is a relationship between classes, where object of one class contains objects of other classes
- The contained object can exist, even if the containing object ceases to exist
- HAS-A relationship between two classes
  - Ex: Account has Locker, Car has tyres
- Implemented by making the contained object reference as an instance variable in containing object




```
class Engine{
        public void start(){..};
}
class Tyre {
        int radius;
        public int getRadius(){..};
}
class Car{
        int speed;
        Engine engine;
        Tyre[] tyres = new Trye[4];
        … methods--     }
```

AGGREGATION AND COMPOSITION

- Composition is a stricter form of aggregation
  - contained object cannot exist, if the containing object ceases to exist
  - Ex. Account has Transactions

- Advantages
  - Helps design classes that follow good OO practices
  - Reuse of classes
  - Avoids code redundancy
  - Ease in Maintenance