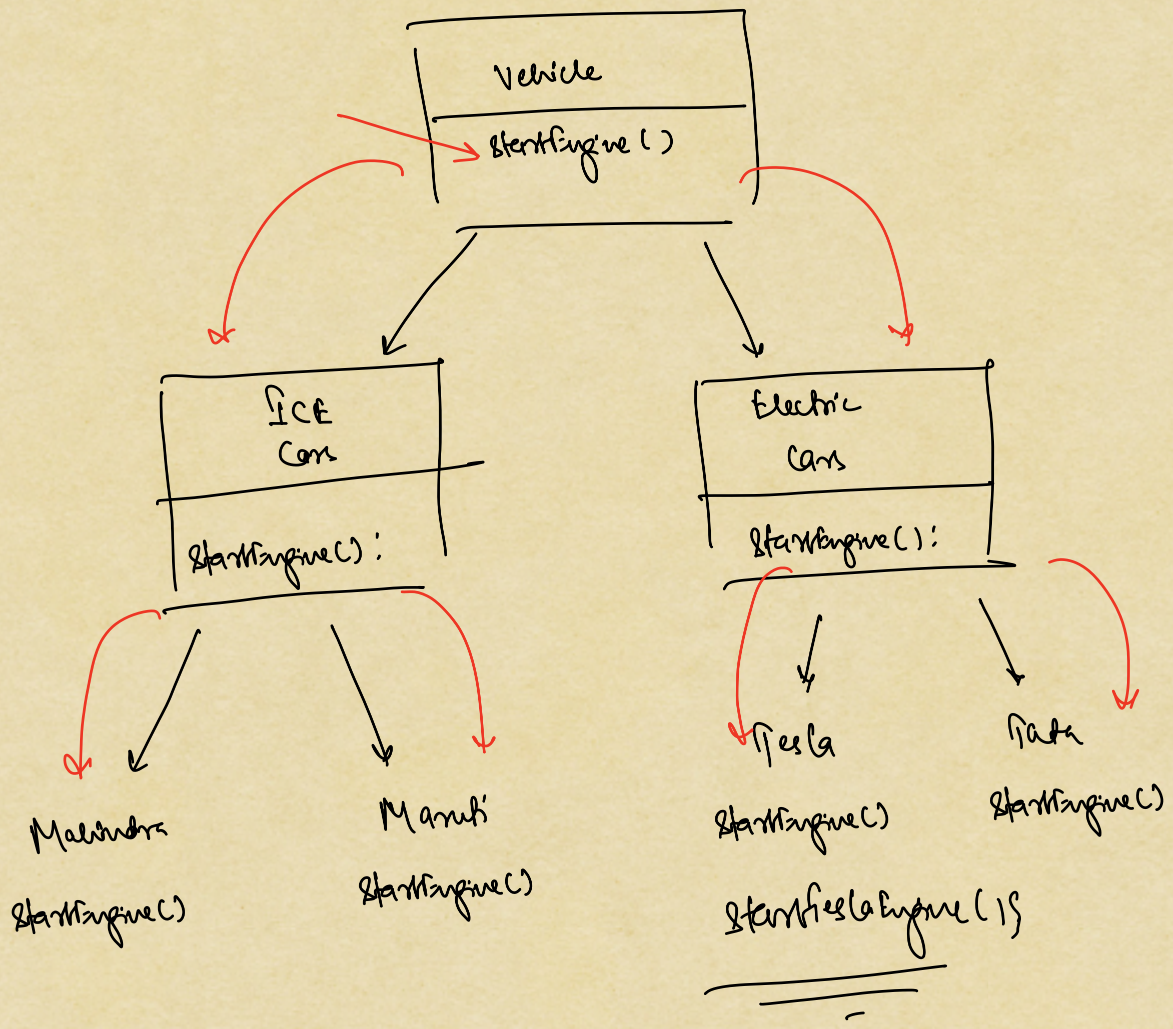
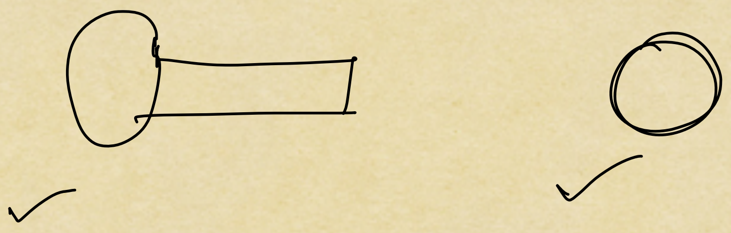
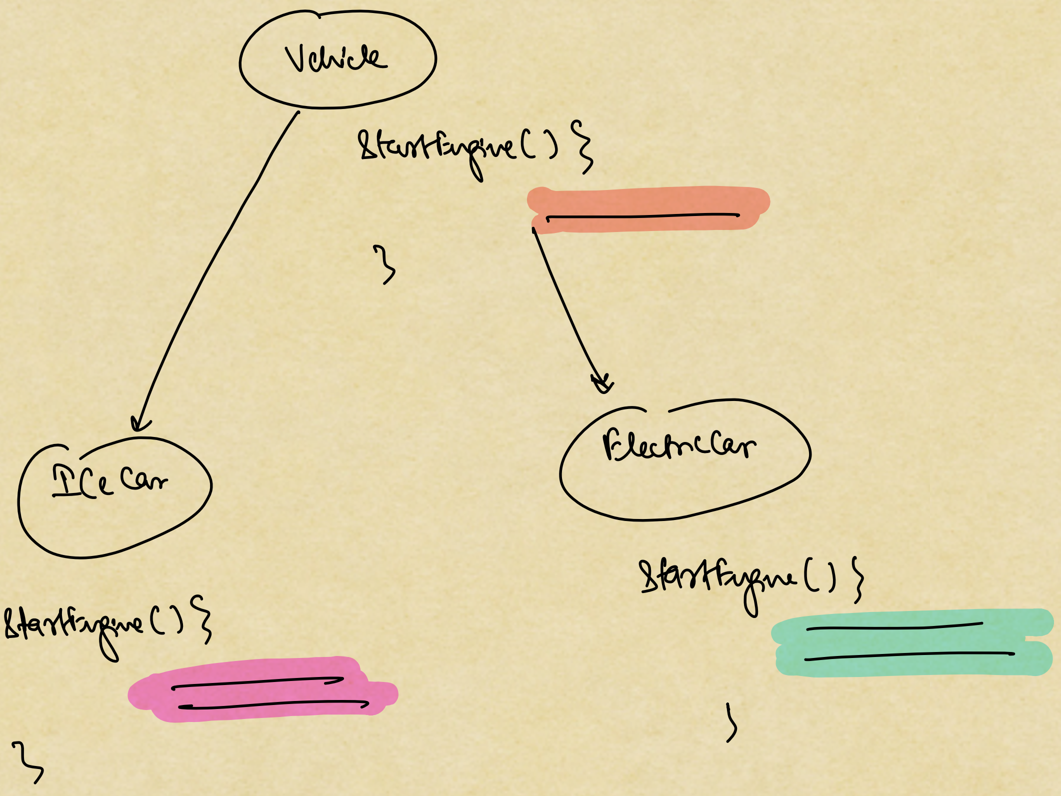
→ Method overriding:-

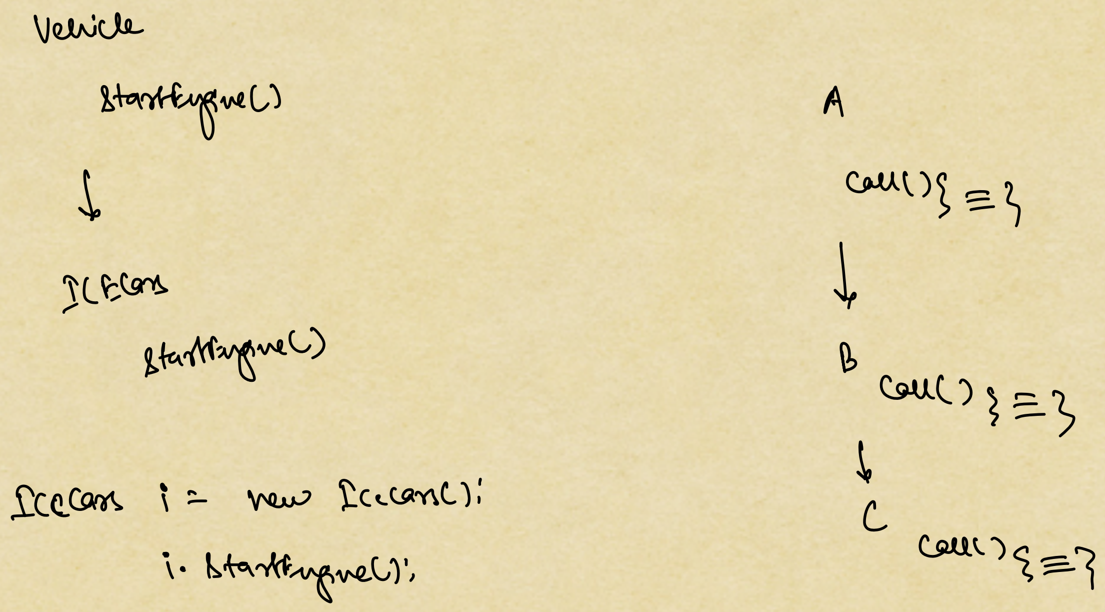




* method signature should remain the same but logic/ implementation inside the method should be diff:- ⇒ method overriding

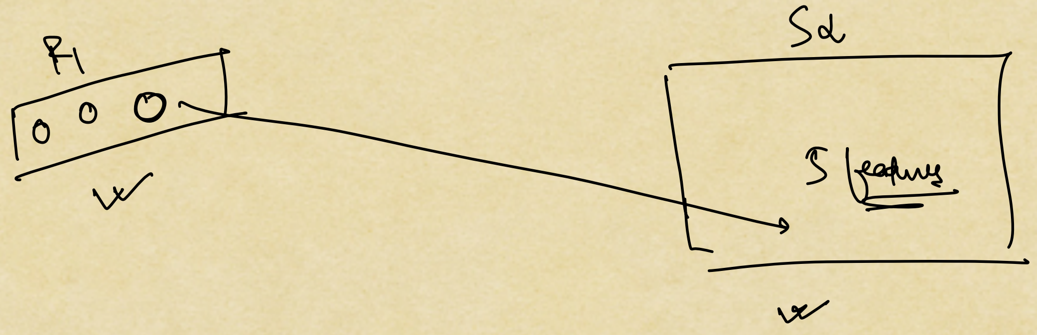


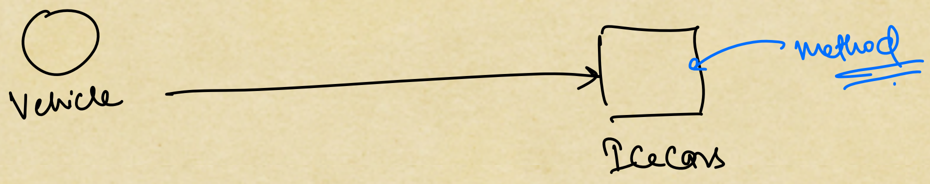
⇒ when method signature is same in parent and child class but the impl differs ⇒ method overriding.



```
IceCars i = new IceCars();
    i.StartEngine();
```

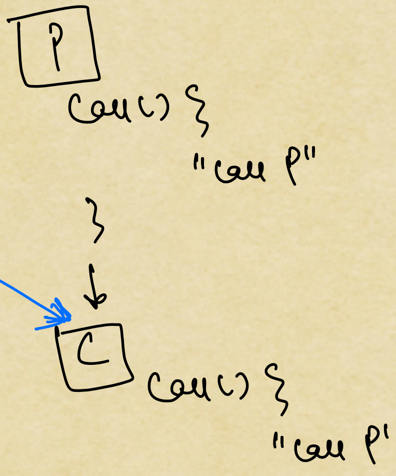
```
Vehicle v = new Vehicle();
        v.startEngine();

Vehicle v = new IceCars();
        v.startEngine();
```





* In case of upcasting the method invoked will always be the method present in Object.

```
P p = new C();
  p.call();
     => call P
```



}

$$\begin{tikzcd}
P \text{ } 3 \arrow[d] \\
C \text{ } 3+2
\end{tikzcd}$$

P p = new C();

$\text{p.} \left.\begin{array}{l} \_\_\_\_ \\ \_\_\_\_ \\ \_\_\_\_ \end{array}\right\}$ $\underbrace{\text{only 3 method calls possible}}_{\text{methods invoked from C.}}$

In case of upcasting;

Parent p = new child();

i) methods possible to execute, are the methods present in Parent class.

ii) methods executed would the methods from child class.

$$\begin{tikzcd}
 & \text{attributes} \\
\text{self} \arrow[ur] \arrow[r] & \underline{\underline{\text{methods}}} \arrow[l, bend left]
\end{tikzcd}$$

execution $\rightarrow$ object

→ YesBank closed down

→ PhonePe → YesBank

$\downarrow$ 24 hrs

IciciBank

↓

Loose coupling

```
YesBankAPI {

1
checkBalance        int getBalance( String userName) {

                    }

2
sendmoney           int sendMoney( int amount, String toUser) {

                    }   1 - Success        2 - inprogress
                        0 - failure

3
changeUPIPIN        boolean changePin( int current, int newPin) {

                    }
```

$$\begin{tikzcd}\boxed{\text{PhonePe}} \arrow[r] & \text{YesBankAPI} \arrow[r] & \text{ICICI Bank}\end{tikzcd}$$

```
ICICI BankAPI {

1
checkBalance        double checkBalance( int userId, int pin){

                    }

2
sendmoney           char    transact( int amount, String toUser, String
                                                                  frmUser
                                        ){
                     }
                          I - inprogress, S - success, F - failure

3
changeUPIPIN          boolean   updatePin( int current, int newPin){

                      }
```

This kind of code where Phonepe is completely & directly dependent on YesBank → tightly coupled code [very bad]

i) we are not using startEngine method inside Vehicle because all the children of Vehicle is overriding the method

ii) for upcasting, we want the method to be present

Ans ⇒ Interface
↓
group of abstract method that can be implemented by multiple classes

```
interface IBankAPI {
    double balance(String user, int pin);
    character transfer(double amount, String fromUser, String toUser)
    boolean updatePin(String user, int newPin, int currentPin);
}
```

YesBankAPI

ICICIBankAPI

$$\begin{tikzcd}\text{PhonePe} \arrow[d] \\ \cancel{\text{YesBankAPI}}\end{tikzcd}$$

```
class   PhonePe {

        IBankAPI      iBankAPI;


        public( IBankAPI    iBankAPI) {
                    this.iBankAPI  =  iBankAPI
        }

    .
    .
    .

}


    IBankAPI    bankAPI  =  new   YesBankAPI();

    IBankAPI    bankAPI  =  new   ICICIBankAPI();


  PhonePe    p  =  new  PhonePe( new  YesBankAPI());

  PhonePe    p  =  new  PhonePe( new  ICICBankAPI());
```

* always code loosely coupled

↓

Adapter Design Pattern