Untitled3

May 6, 2023

1 Encapsulation

[14]:
```
## this is the eg of not using the encapsulation! that leads to modification
```

[2]:
```
class test:
    def __init__(self, a,b):
        self.a = a
        self.b = b
```

[4]:
```
t = test(45,55)
```

[6]:
```
t.a
```

[6]: 45

[7]:
```
t.b
```

[7]: 55

[8]:
```
t.a = 546
```

[9]:
```
t.a
```

[9]: 546

[13]:
```
### u can access and also perform the function
## any one can make any kind of  modification to avoid that we use encapsulation
```

[12]:
```
### Encapsulation is an function to prevent any direct modification of data in␣
↪OOPS
```

[ ]:
```

```

[32]:
```
class car:

    def __init__(self,year,make,model,speed):
        self.__year = year
```

```
        self.__make = make
        self.__model = model
        self.__speed = 0
```

[50]:
```
vechicle_name = car(2002,"TATA","curvv",100)
```

[69]:
```
vechicle_name._car__year
```

[69]:
```
2002
```

[ ]:
```
vechicle_name._car__make
```

[ ]:
```

```

[41]:
```
## if someone else wants to modify the data it is not possible to do it !
## because he/she will not be knowing the name of class and __keep it has␣
 ↪private
```

[53]:
```
vechicle_name.make
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[53], line 1
----> 1 vechicle_name.make

AttributeError: 'car' object has no attribute 'make'
```

[54]:
```
## this is what the other person faces the problem during the modification
```

[55]:
```
### and if i want to allow to modify only the speed of the car
```

[88]:
```
class car:

    def __init__(self,year,make,model,speed):
        self.__year = year
        self.__make = make
        self.__model = model
        self.__speed = 100

    def set_speed(self,speed):
        self.__speed = 0 if speed > 0 else speed

    def get_speed(self):
        return self.__speed
```

[ ]:

[122]:
```
### no can change the balance because of constructor
```

[115]:
```
class bank_account:
    def __init__(self,balance):
        self.__balance = balance

    def deposit(self, amount):
        self.__balance = self.__balance + amount

    def withdraw(self, amount):
        if self.__balance >= amount:
            self.__balance = self.__balance - amount
            return True
        else :
            return False

    def get_balance(self):
        return self.__balance
```

[116]:
```
account = bank_account(2000)
```

[117]:
```
account.get_balance()
```

[117]:
```
2000
```

[118]:
```
account.deposit(35000000)
```

[119]:
```
account.get_balance()
```

[119]:
```
35002000
```

[120]:
```
account.withdraw(1)
```

[120]:
```
True
```

[121]:
```
account.get_balance()
```

[121]:
```
35001999
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: