

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | | |

**Design patterns:**

**In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design.**

**Structural pattern:**

Facade pattern: **Facade** is a structural design pattern that provides a simplified interface to a library, a framework, or any other complex set of classes.

Problem**:**Ordinarily, you’d need to initialize all of those objects, keep track of dependencies, execute methods in the correct order, and so on.

## Solution: A facade is a class that provides a simple interface to a complex subsystem which contains lots of moving parts. A facade might provide limited functionality in comparison to working with the subsystem directly. However, it includes only those features that clients really care about.



class Tyre():

def \_\_init\_\_(self,name):

self.pressure = 50

class Tank():

def \_\_init\_\_(self, name):

self.level = 100

class Car():

def \_\_init\_\_(self):

self.\_tyres = [Tyre('front\_left'),

Tyre('front\_right'),

Tyre('rear\_left'),

Tyre('rear\_right'), ]

self.\_tank = Tank(70)

def tyres\_pressure(self):

return [tyre.pressure for tyre in self.\_tyres]

def fuel\_level(self):

return self.\_tank.level

c = Car()

print(c.tyres\_pressure())

print(c.fuel\_level())

output:

[50, 50, 50, 50]

100