

Python -pytest

Prompt

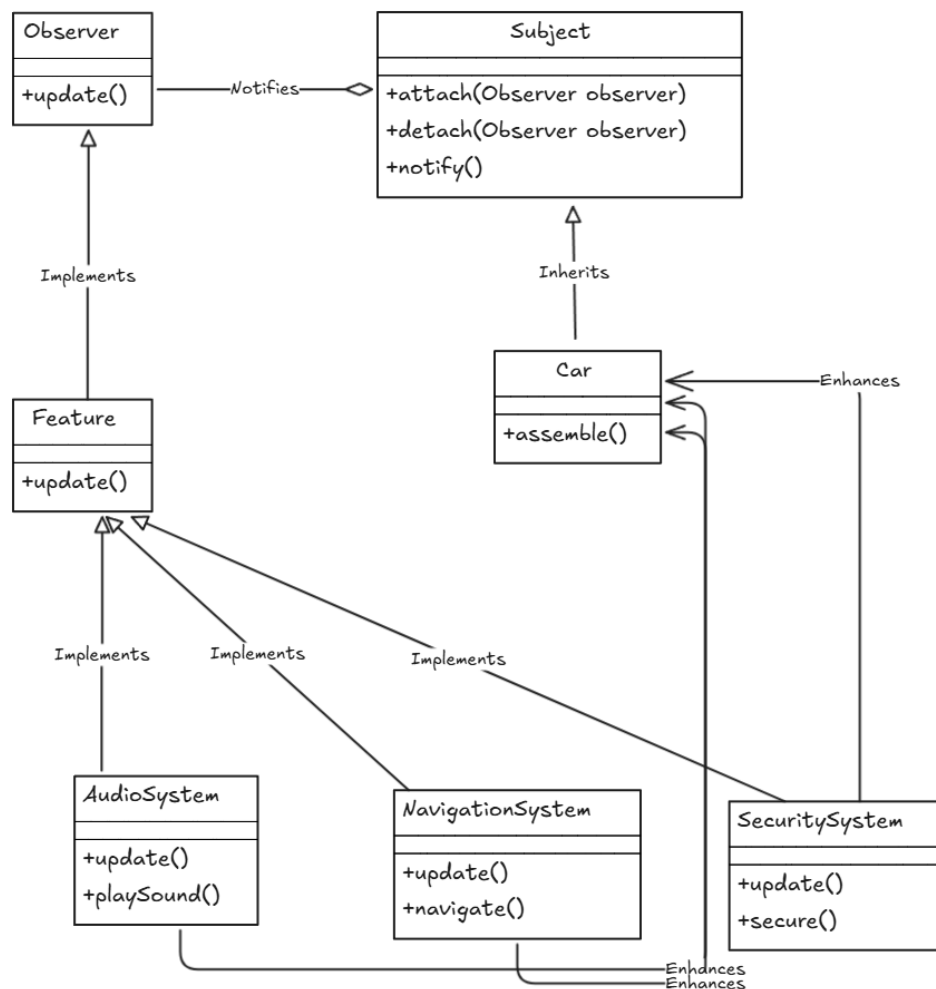
-Observer

Use the Observer pattern to write code in Python programming language and write pytest to test the given code that have 100% statement coverage from the requirements below.

Use Observer pattern to design automotive system development by adding features to cars, such as audio system, navigation system or security systems without changing the main structure of the car.

Diagram

Observer



ChatGPT4o

Observer

รอบที่ 1

Code.py

```
from abc import ABC, abstractmethod

# Subject (Observable)
class Car(ABC):
    def __init__(self):
        self._observers = []

    def add_feature(self, observer):
        self._observers.append(observer)

    def remove_feature(self, observer):
        self._observers.remove(observer)

    def notify(self):
        for observer in self._observers:
            observer.update(self)

# Concrete Subject
class MyCar(Car):
    def __init__(self):
        super().__init__()
        self._state = None

    @property
    def state(self):
        return self._state

    @state.setter
```

```
def state(self, state):
    self._state = state
    self.notify()

# Observer
class Feature(ABC):
    @abstractmethod
    def update(self, car):
        pass

# Concrete Observers
class AudioSystem(Feature):
    def update(self, car):
        if car.state == "audio_on":
            print("Audio System is now ON.")
        elif car.state == "audio_off":
            print("Audio System is now OFF.")

class NavigationSystem(Feature):
    def update(self, car):
        if car.state == "navigation_on":
            print("Navigation System is now ON.")
        elif car.state == "navigation_off":
            print("Navigation System is now OFF.")

class SecuritySystem(Feature):
    def update(self, car):
        if car.state == "security_on":
            print("Security System is now ON.")
        elif car.state == "security_off":
            print("Security System is now OFF.")

# Client code to demonstrate the pattern
if __name__ == "__main__":
```

```
car = MyCar()

audio = AudioSystem()
navigation = NavigationSystem()
security = SecuritySystem()

car.add_feature(audio)
car.add_feature(navigation)
car.add_feature(security)

car.state = "audio_on"
car.state = "navigation_on"
car.state = "security_on"

car.state = "audio_off"
car.state = "security_off"
```

Test.py

```
import pytest
from io import StringIO
import sys

from code import MyCar, AudioSystem, NavigationSystem, SecuritySystem

@pytest.fixture
def car():
    return MyCar()

@pytest.fixture
def audio_system():
    return AudioSystem()
```

```
@pytest.fixture
def navigation_system():
    return NavigationSystem()

@pytest.fixture
def security_system():
    return SecuritySystem()

def test_audio_system_on(car, audio_system, capsys):
    car.add_feature(audio_system)
    car.state = "audio_on"
    captured = capsys.readouterr()
    assert "Audio System is now ON." in captured.out

def test_audio_system_off(car, audio_system, capsys):
    car.add_feature(audio_system)
    car.state = "audio_off"
    captured = capsys.readouterr()
    assert "Audio System is now OFF." in captured.out

def test_navigation_system_on(car, navigation_system, capsys):
    car.add_feature(navigation_system)
    car.state = "navigation_on"
    captured = capsys.readouterr()
    assert "Navigation System is now ON." in captured.out

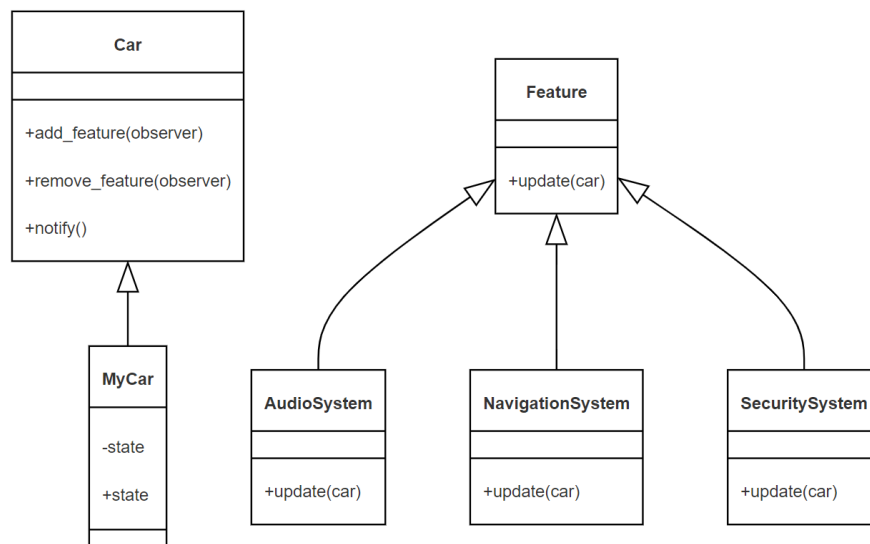
def test_navigation_system_off(car, navigation_system, capsys):
    car.add_feature(navigation_system)
    car.state = "navigation_off"
    captured = capsys.readouterr()
    assert "Navigation System is now OFF." in captured.out

def test_security_system_on(car, security_system, capsys):
    car.add_feature(security_system)
```

```
car.state = "security_on"  
captured = capsys.readouterr()  
assert "Security System is now ON." in captured.out
```

```
def test_security_system_off(car, security_system, capsys):  
    car.add_feature(security_system)  
    car.state = "security_off"  
    captured = capsys.readouterr()  
    assert "Security System is now OFF." in captured.out
```

Diagram From Code:



Result:

```
split class in code.py
```

```
TestCar.py(Edit)
```

```
import pytest
```

```
from Car import MyCar
```

```
from Feature import AudioSystem,NavigationSystem,SecuritySystem
```

```
5 Pass, 1 Error
```