Python -pytest

Prompt

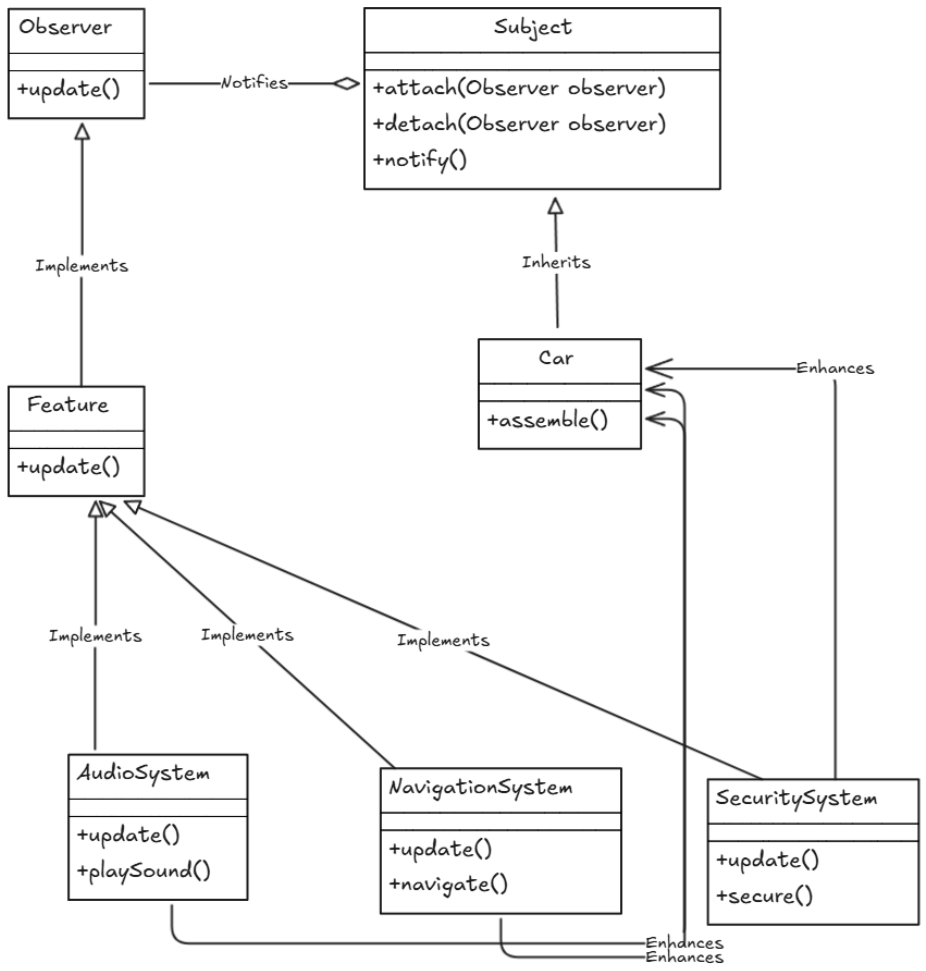
-Observer

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



GitHub Copilot(Education – Student version)

Observer

รอบที่ 2

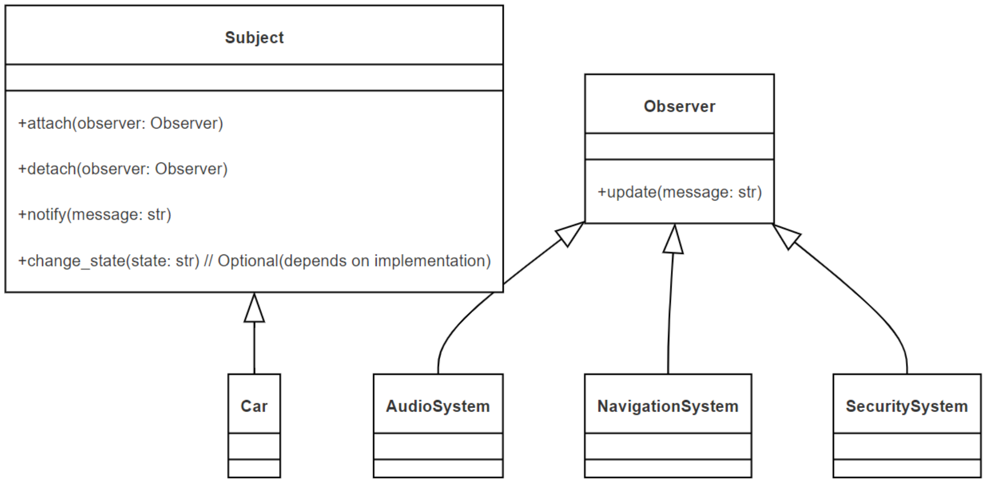
Code.py

```
# observer_pattern.py

from abc import ABC, abstractmethod

class Observer(ABC):
    @abstractmethod
    def update(self, message: str):
        pass

class Car:
    def __init__(self):
        self._observers = []
        self._state = None

    def attach(self, observer: Observer):
        if observer not in self._observers:
            self._observers.append(observer)

    def detach(self, observer: Observer):
        if observer in self._observers:
            self._observers.remove(observer)

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

    def change_state(self, state: str):
        self._state = state
        self.notify(f"Car state changed to {state}")

class AudioSystem(Observer):
    def update(self, message: str):
        print(f"AudioSystem received: {message}")

class NavigationSystem(Observer):
    def update(self, message: str):
        print(f"NavigationSystem received: {message}")

class SecuritySystem(Observer):
    def update(self, message: str):
        print(f"SecuritySystem received: {message}")
```

Test.py

```
# test_observer_pattern.py

import pytest
from code import Car, AudioSystem, NavigationSystem, SecuritySystem

def test_attach_detach_observers():
    car = Car()
    audio = AudioSystem()
    nav = NavigationSystem()
    sec = SecuritySystem()

    car.attach(audio)
    car.attach(nav)
    car.attach(sec)

    assert len(car._observers) == 3

    car.detach(nav)
    assert len(car._observers) == 2

def test_notify_observers(capfd):
    car = Car()
    audio = AudioSystem()
    nav = NavigationSystem()
    sec = SecuritySystem()

    car.attach(audio)
    car.attach(nav)
    car.attach(sec)

    car.change_state("Engine On")

    captured = capfd.readouterr()
    assert "AudioSystem received: Car state changed to Engine On" in captured.out
    assert "NavigationSystem received: Car state changed to Engine On" in captured.out
    assert "SecuritySystem received: Car state changed to Engine On" in captured.out

def test_no_observers_notified(capfd):
    car = Car()
    car.change_state("Engine Off")

    captured = capfd.readouterr()
    assert captured.out == ""
```

Diagram From Code:


Subject
+attach(observer: Observer)
+detach(observer: Observer)
+notify(message: str)
+change_state(state: str) // Optional(depends on implementation)
Observer
+update(message: str)
Car
AudioSystem
NavigationSystem
SecuritySystem

Result:

split class in code.py

TestObserver.py(Edit)

from Observer import Car, AudioSystem, NavigationSystem, SecuritySystem

3 Pass