Python -pytest

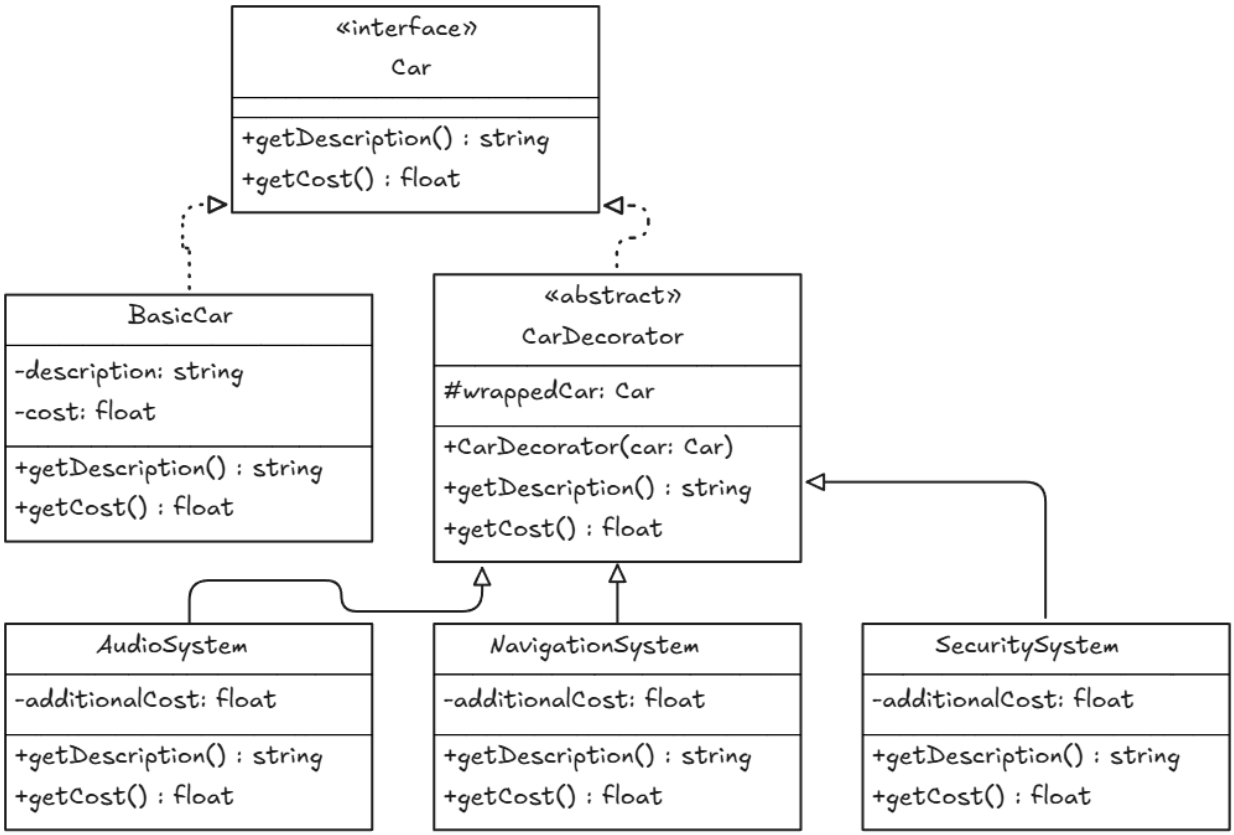
Prompt

-Decorator

Use the Decorator 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 Decorator 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

Decorator



GitHub Copilot(Education – Student version)

Decorator

รอบที่ 3

Code.py

```
# car.py
from abc import ABC, abstractmethod

class Car(ABC):
    @abstractmethod
    def get_description(self) -> str:
        pass

    @abstractmethod
    def get_cost(self) -> float:
        pass

class BasicCar(Car):
    def get_description(self) -> str:
        return "Basic Car"

    def get_cost(self) -> float:
        return 10000.0
# decorators.py



class CarDecorator(Car):
    def __init__(self, car: Car):
        self._car = car

    def get_description(self) -> str:
        return self._car.get_description()
```

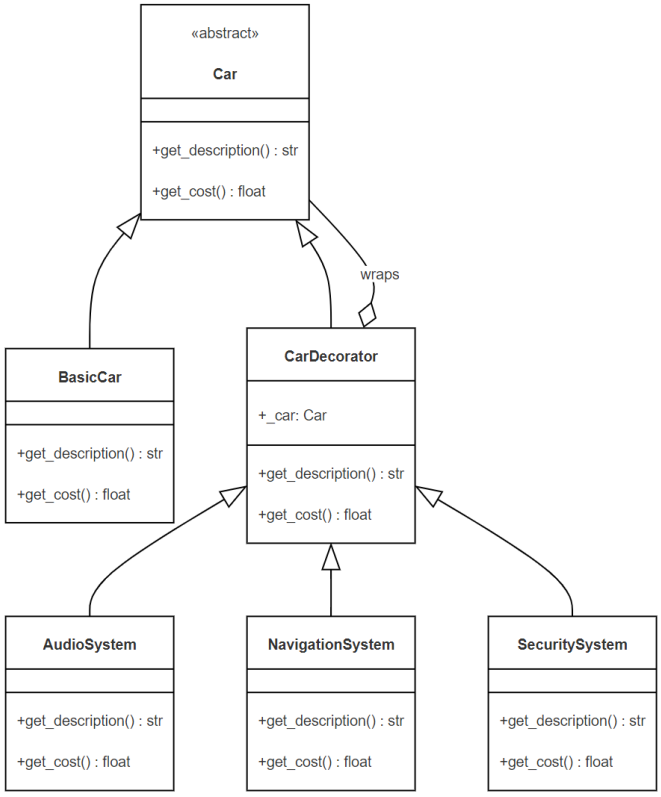
```
    def get_cost(self) -> float:
        return self._car.get_cost()

class AudioSystem(CarDecorator):
    def get_description(self) -> str:
        return f"{self._car.get_description()} + Audio System"

    def get_cost(self) -> float:
        return self._car.get_cost() + 1500.0

class NavigationSystem(CarDecorator):
    def get_description(self) -> str:
        return f"{self._car.get_description()} + Navigation System"

    def get_cost(self) -> float:
        return self._car.get_cost() + 2000.0

class SecuritySystem(CarDecorator):
    def get_description(self) -> str:
        return f"{self._car.get_description()} + Security System"

    def get_cost(self) -> float:
        return self._car.get_cost() + 2500.0
```

Test.py

```
# test_decorators.py
import pytest
from code import BasicCar
from code import AudioSystem, NavigationSystem, SecuritySystem

def test_basic_car():
    car = BasicCar()
    assert car.get_description() == "Basic Car"
```

```
    assert car.get_cost() == 10000.0


def test_audio_system():
    car = AudioSystem(BasicCar())
    assert car.get_description() == "Basic Car + Audio System"
    assert car.get_cost() == 11500.0


def test_navigation_system():
    car = NavigationSystem(BasicCar())
    assert car.get_description() == "Basic Car + Navigation System"
    assert car.get_cost() == 12000.0


def test_security_system():
    car = SecuritySystem(BasicCar())
    assert car.get_description() == "Basic Car + Security System"
    assert car.get_cost() == 12500.0


def test_all_features():
    car = SecuritySystem(NavigationSystem(AudioSystem(BasicCar())))
    assert car.get_description() == "Basic Car + Audio System + Navigation System + Security System"
    assert car.get_cost() == 16000.0
```

Diagram From Code:



Result:

split class in code.py

TestDecorator.py(Edit)

from Car import BasicCar

from Decorator import AudioSystem, NavigationSystem, SecuritySystem

5 Pass