

Python -pytest

Prompt

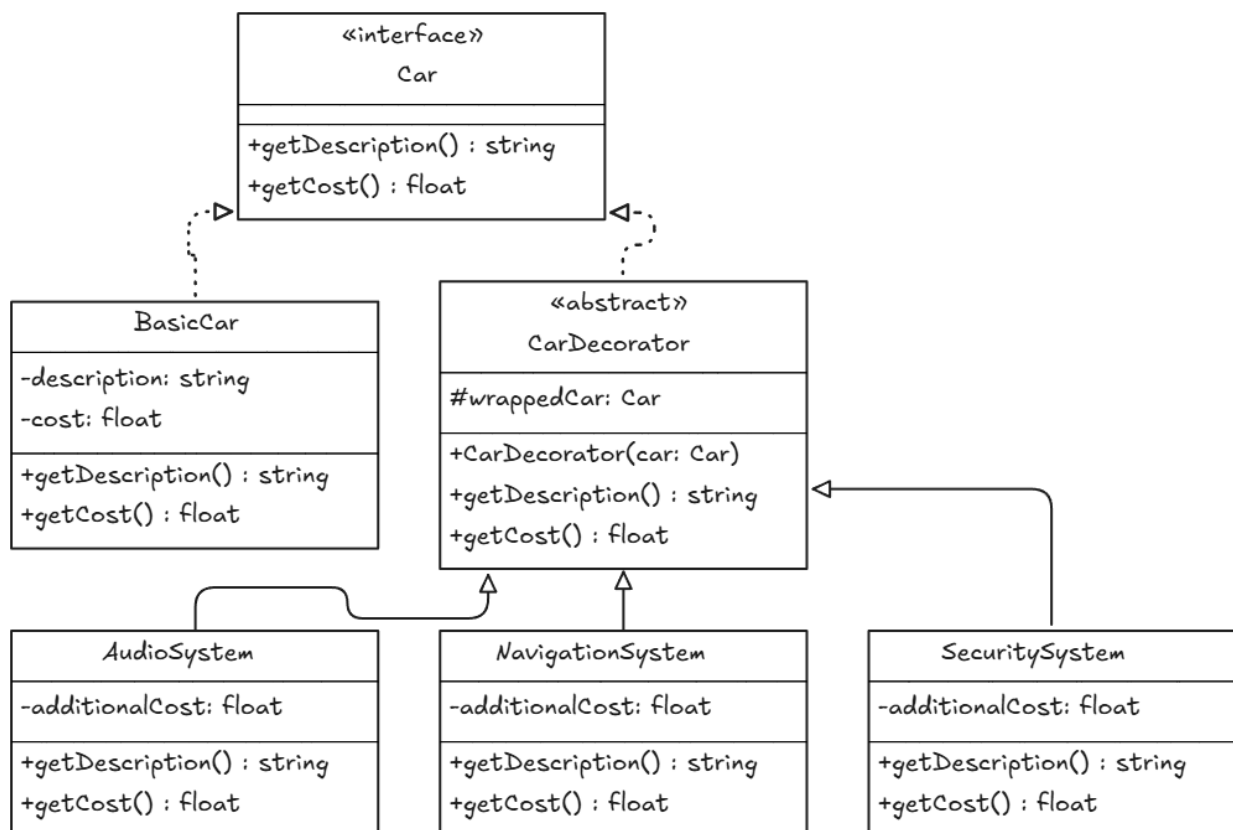
-Decorator

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



ChatGPT4o

Decorator

รอบที่ 3

Code.py

```
from abc import ABC, abstractmethod

class Car(ABC):
    @abstractmethod
    def description(self):
        pass

    @abstractmethod
    def cost(self):
        pass

class BasicCar(Car):
    def description(self):
        return "Basic Car"

    def cost(self):
        return 10000

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

    def description(self):
        return self._car.description()

    def cost(self):
        return self._car.cost()

class AudioSystem(CarDecorator):
    def description(self):
        return self._car.description() + " + Audio System"
```

```
def cost(self):
    return self._car.cost() + 1500

class NavigationSystem(CarDecorator):
    def description(self):
        return self._car.description() + " + Navigation System"

    def cost(self):
        return self._car.cost() + 2500

class SecuritySystem(CarDecorator):
    def description(self):
        return self._car.description() + " + Security System"

    def cost(self):
        return self._car.cost() + 2000
```

Test.py

```
import pytest
from code import BasicCar, AudioSystem, NavigationSystem, SecuritySystem

def test_basic_car():
    car = BasicCar()
    assert car.description() == "Basic Car"
    assert car.cost() == 10000

def test_car_with_audio_system():
    car = AudioSystem(BasicCar())
    assert car.description() == "Basic Car + Audio System"
    assert car.cost() == 11500
```

```

def test_car_with_navigation_system():
    car = NavigationSystem(BasicCar())
    assert car.description() == "Basic Car + Navigation System"
    assert car.cost() == 12500

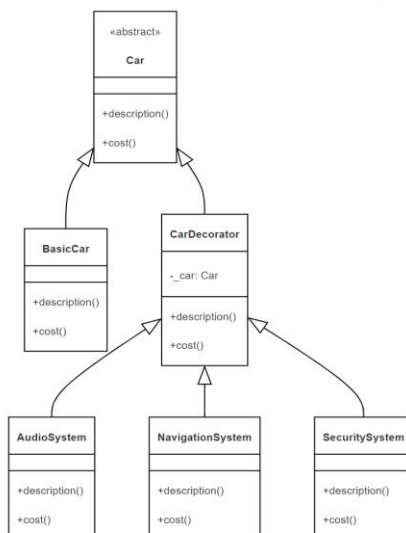
def test_car_with_security_system():
    car = SecuritySystem(BasicCar())
    assert car.description() == "Basic Car + Security System"
    assert car.cost() == 12000

def test_car_with_audio_and_navigation_systems():
    car = NavigationSystem(AudioSystem(BasicCar()))
    assert car.description() == "Basic Car + Audio System + Navigation System"
    assert car.cost() == 14000

def test_car_with_all_features():
    car = SecuritySystem(NavigationSystem(AudioSystem(BasicCar())))
    assert car.description() == "Basic Car + Audio System + Navigation System + Security System"
    assert car.cost() == 16000

```

Diagram From Code:



Result:

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

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

```
from BasicCar import BasicCar
```

```
from AudioSystem import AudioSystem
```

```
from NavigationSystem import NavigationSystem
```

```
from SecuritySystem import SecuritySystem
```

```
5 Fail, 1 Pass
```