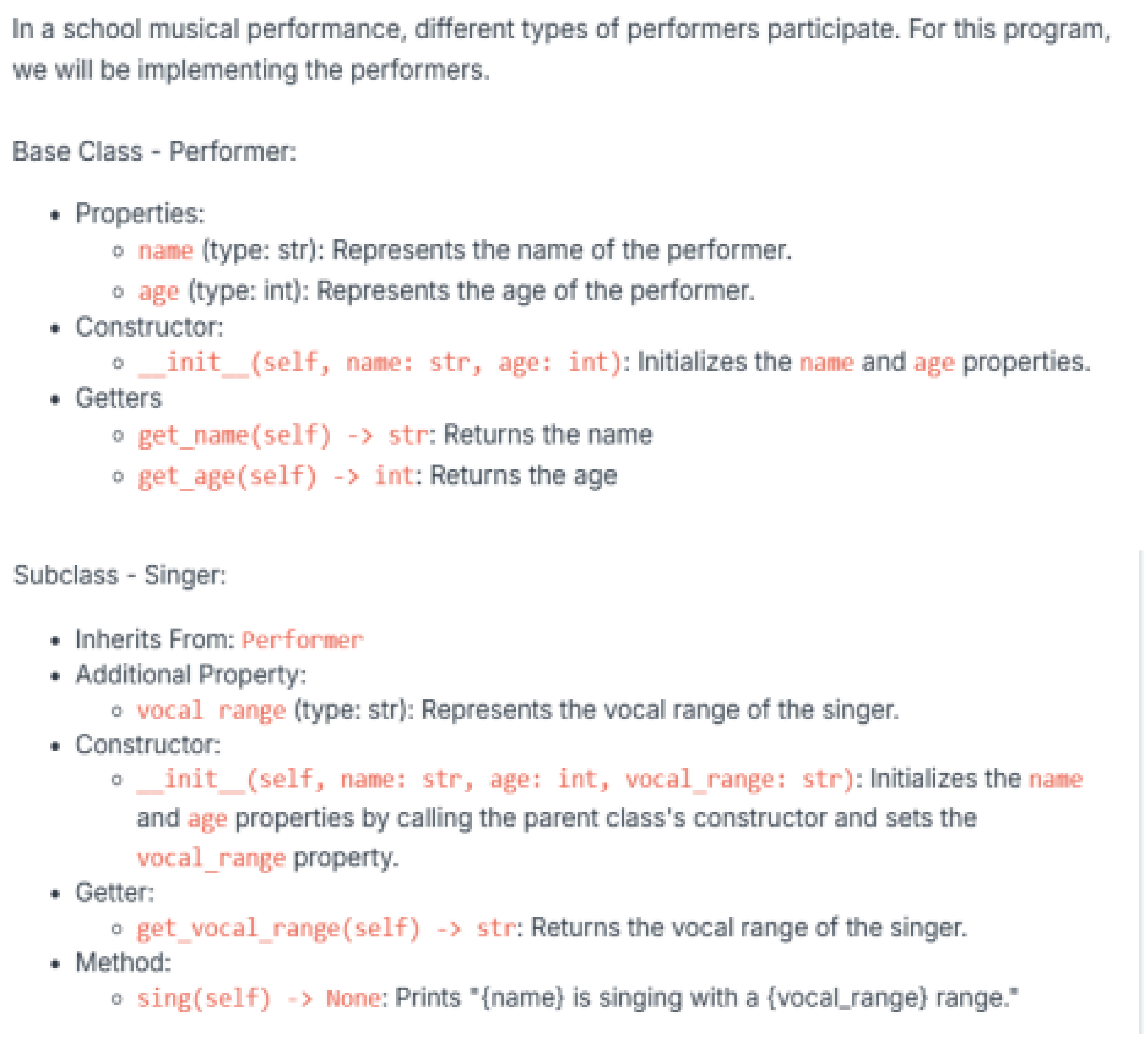
Finals Task 2. Inheritance

Problem School Performance

Note: You are to create 4 separate python files for this task:

- **performer.py(base class)**
- **singer.py(sub class)**
- **dancer.py(sub class)**
- **test_class.py – following the required test cases**



In a school musical performance, different types of performers participate. For this program, we will be implementing the performers.

Base Class - Performer:

- Properties:
  - `name` (type: str): Represents the name of the performer.
  - `age` (type: int): Represents the age of the performer.
- Constructor:
  - `__init__(self, name: str, age: int)`: Initializes the `name` and `age` properties.
- Getters
  - `get_name(self) -> str`: Returns the name
  - `get_age(self) -> int`: Returns the age

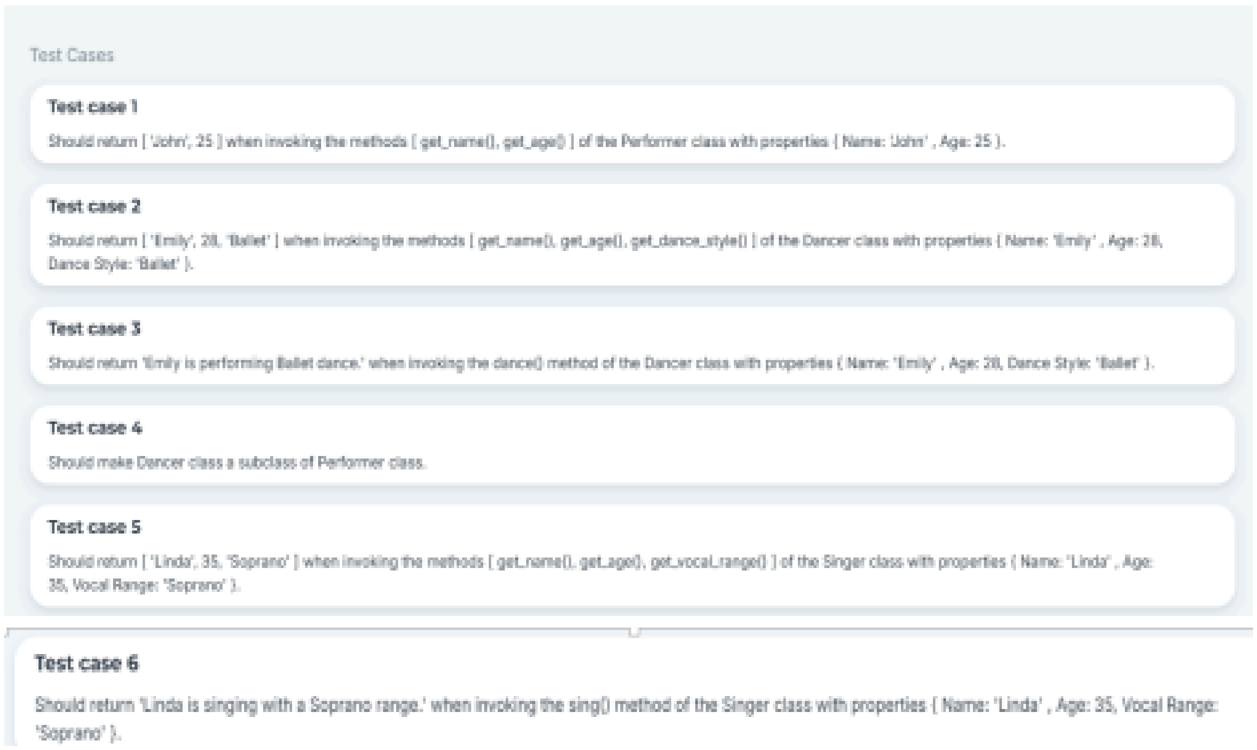
Subclass - Singer:

- Inherits From: `Performer`
- Additional Property:
  - `vocal_range` (type: str): Represents the vocal range of the singer.
- Constructor:
  - `__init__(self, name: str, age: int, vocal_range: str)`: Initializes the `name` and `age` properties by calling the parent class's constructor and sets the `vocal_range` property.
- Getter:
  - `get_vocal_range(self) -> str`: Returns the vocal range of the singer.
- Method:
  - `sing(self) -> None`: Prints "{name} is singing with a {vocal_range} range."

Subclass - Dancer:

- Inherits From: `Performer`
- Additional Property:
  - `dance_style` (type: str): Represents the dance style of the dancer.
- Constructor:
  - `__init__(self, name: str, age: int, dance_style: str)`: Initializes the `name` and `age` properties by calling the parent class's constructor and sets the `dance_style` property.
- Getter:
  - `get_dance_style(self) -> str`: Returns the dance style of the dancer.
- Method:
  - `dance(self) -> None`: Prints "{name} is performing {dance_style} dance."

Sample output for the Test Class



Test Cases

Test case 1

Should return ['John', 25] when invoking the methods [get_name(), get_age()] of the Performer class with properties { Name: 'John' , Age: 25 }.

Test case 2

Should return ['Emily', 28, 'Ballet'] when invoking the methods [get_name(), get_age(), get_dance_style()] of the Dancer class with properties { Name: 'Emily' , Age: 28, Dance Style: 'Ballet' }.

Test case 3

Should return 'Emily is performing Ballet dance.' when invoking the dance() method of the Dancer class with properties { Name: 'Emily' , Age: 28, Dance Style: 'Ballet' }.

Test case 4

Should make Dancer class a subclass of Performer class.

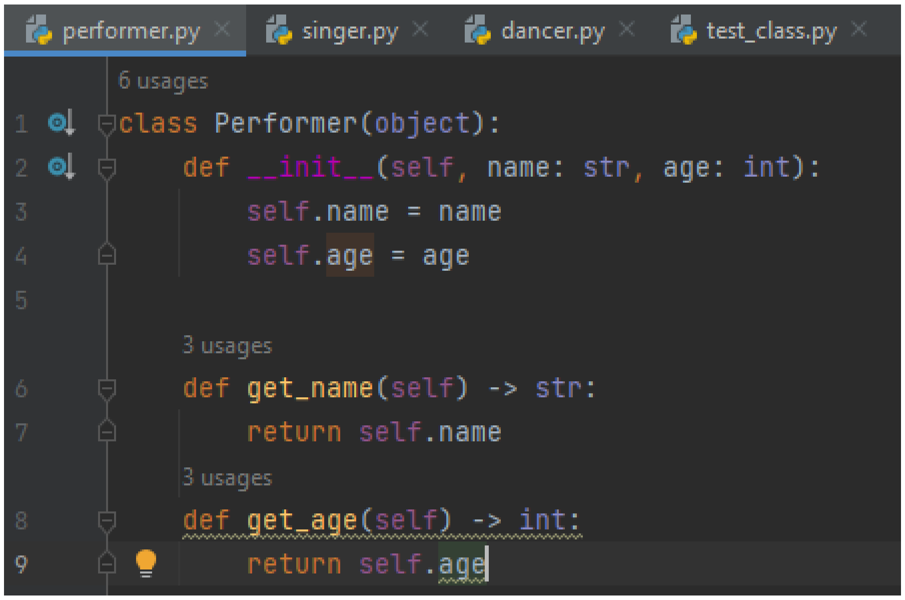
Test case 5

Should return ['Linda', 35, 'Soprano'] when invoking the methods [get_name(), get_age(), get_vocal_range()] of the Singer class with properties { Name: 'Linda' , Age: 35, Vocal Range: 'Soprano' }.

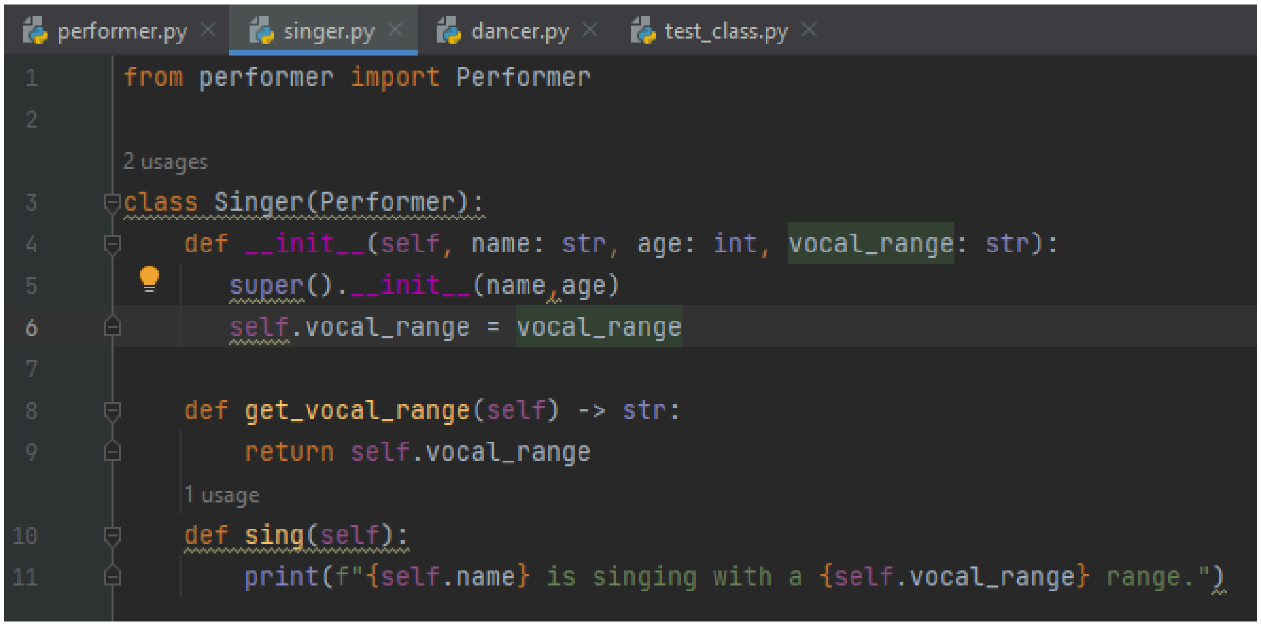
Test case 6

Should return 'Linda is singing with a Soprano range.' when invoking the sing() method of the Singer class with properties { Name: 'Linda' , Age: 35, Vocal Range: 'Soprano' }.

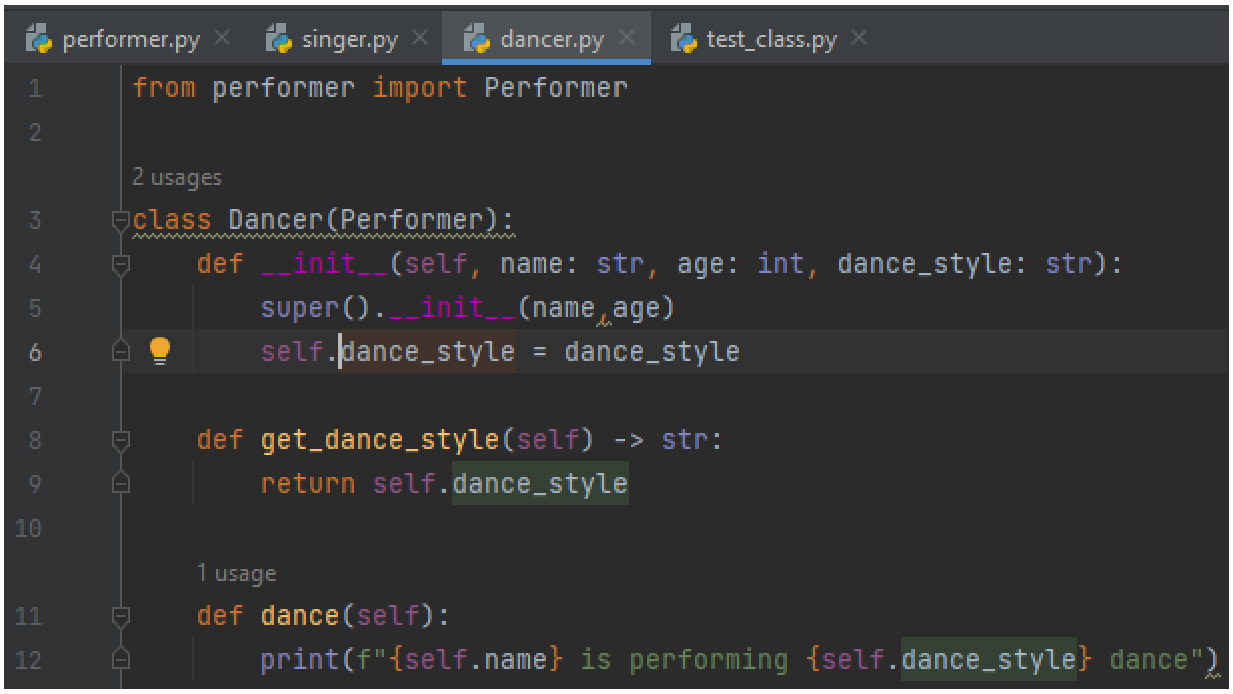
CODE


performer.py   singer.py   dancer.py   test_class.py

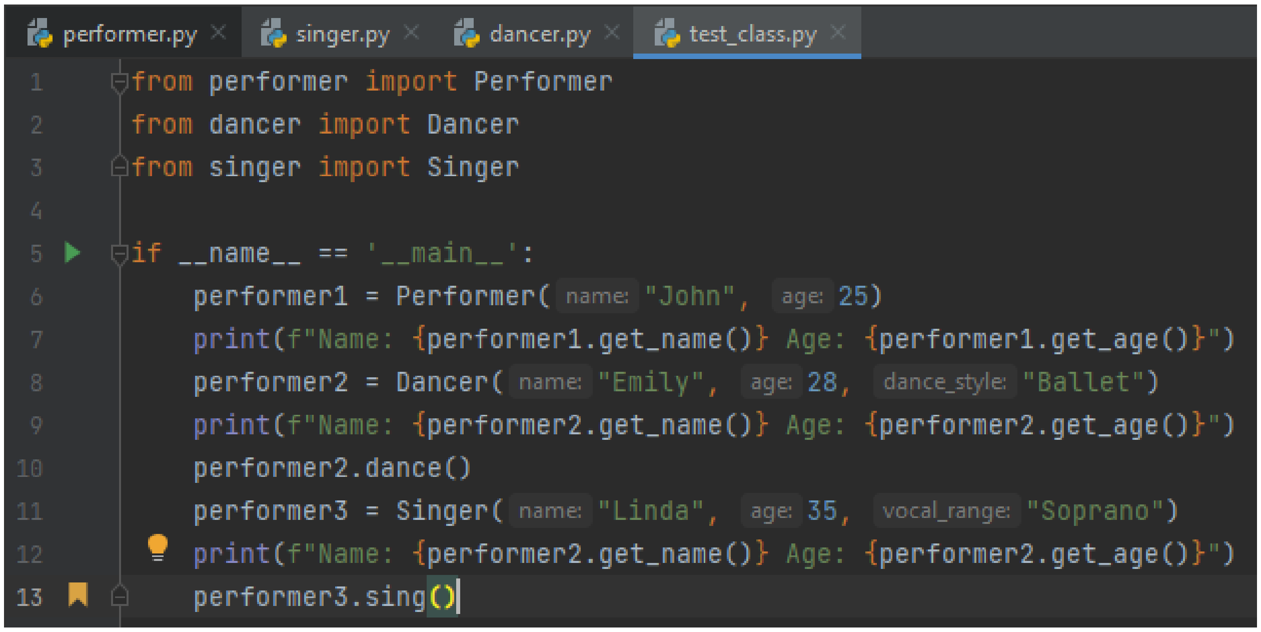
```
6 usages
class Performer(object):
    def __init__(self, name: str, age: int):
        self.name = name
        self.age = age

    3 usages
    def get_name(self) -> str:
        return self.name
    3 usages
    def get_age(self) -> int:
        return self.age
```


performer.py   singer.py   dancer.py   test_class.py

```
from performer import Performer

2 usages
class Singer(Performer):
    def __init__(self, name: str, age: int, vocal_range: str):
        super().__init__(name,age)
        self.vocal_range = vocal_range

    def get_vocal_range(self) -> str:
        return self.vocal_range
    1 usage
    def sing(self):
        print(f"{self.name} is singing with a {self.vocal_range} range.")
```

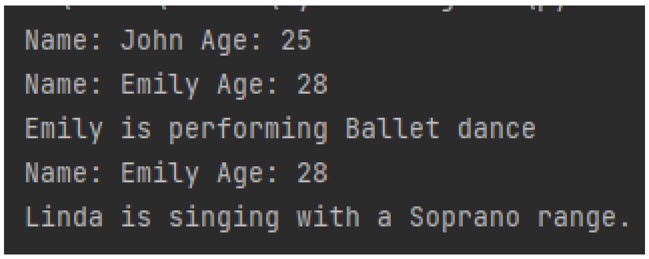

performer.py × singer.py × dancer.py × test_class.py ×

```
from performer import Performer

2 usages
class Dancer(Performer):
    def __init__(self, name: str, age: int, dance_style: str):
        super().__init__(name,age)
        self.dance_style = dance_style

    def get_dance_style(self) -> str:
        return self.dance_style

    1 usage
    def dance(self):
        print(f"{self.name} is performing {self.dance_style} dance")
```


performer.py × singer.py × dancer.py × test_class.py ×

```
from performer import Performer
from dancer import Dancer
from singer import Singer

if __name__ == '__main__':
    performer1 = Performer( name: "John",  age: 25)
    print(f"Name: {performer1.get_name()} Age: {performer1.get_age()}")
    performer2 = Dancer( name: "Emily",  age: 28,  dance_style: "Ballet")
    print(f"Name: {performer2.get_name()} Age: {performer2.get_age()}")
    performer2.dance()
    performer3 = Singer( name: "Linda",  age: 35,  vocal_range: "Soprano")
    print(f"Name: {performer2.get_name()} Age: {performer2.get_age()}")
    performer3.sing()
```

SAMPLE OUTPUT



```
Name: John Age: 25
Name: Emily Age: 28
Emily is performing Ballet dance
Name: Emily Age: 28
Linda is singing with a Soprano range.
```