Name: **Torres, Justine Kurt Q.**
7OOP
Course/Section: **BSCS – C204** **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 parents class's constructor and sets the vocal_range property

• Getter:
- get_vocal_range(self) -> str: Returns the vocal range of the singer. •

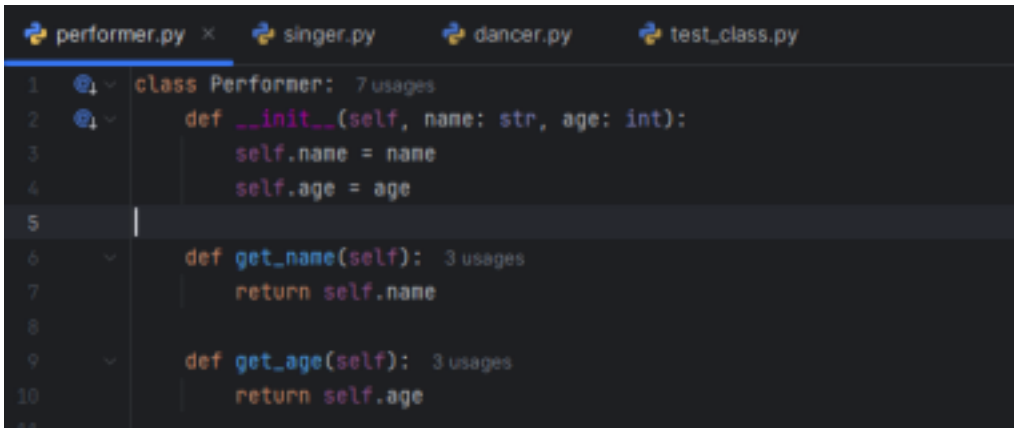
Method:
- sing(self) -> None: Prints "{name} is singing with a {vocal_range} range."

Subclass – **Dancer:**

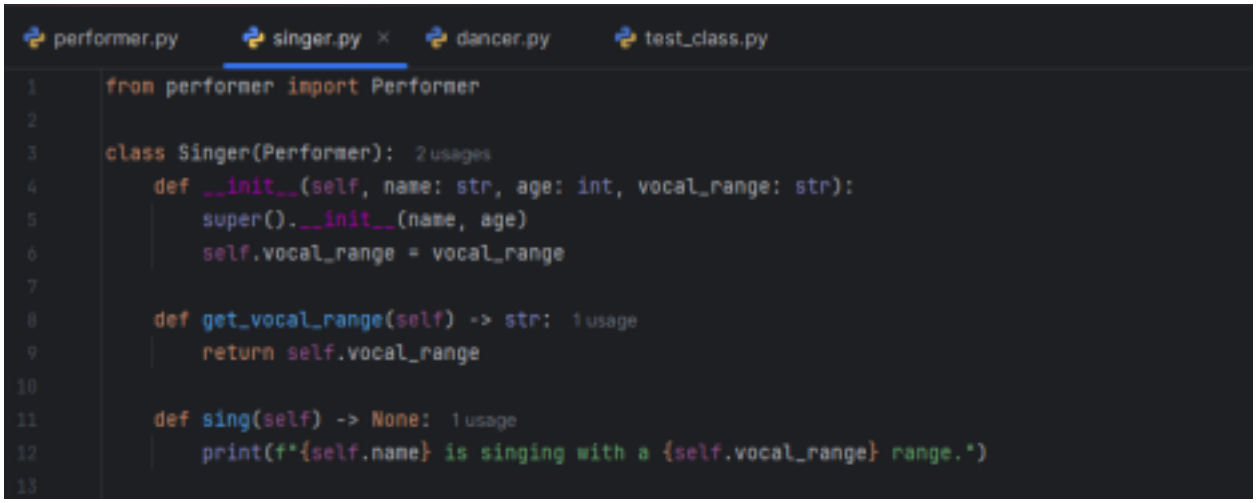
• Inherits From: Performer

• 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 parents 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."

*Base Class – **Performer** (Source Code):*



*Subclass – **Singer** (Source Code):*



Subclass – **Dancer** *(Source Code):*

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

```python
from performer import Performer

class Dancer(Performer):  3 usages
    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:  1 usage
        return self.dance_style

    def dance(self) -> None:  1 usage
        print(f"{self.name} is performing {self.dance_style} dance.")
```

Test Class *(Source Code):*

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

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

print("Test Case 1")
performer1 = Performer( name: "kurt",  age: 20)
print(f"Performer Properties. \n[name:{performer1.get_name()}, age:{performer1.get_age()}]")
print("-" * 40)

print("Test Case 2")
dancer1 = Dancer( name: "dexter",  age: 20,  dance_style: "hiphop")
print(f"Dancer properties. \n[name:{dancer1.get_name()}, age:{dancer1.get_age()}, dance_style:{dancer1.get_dance_style()}]")
print("-" * 40)

print("Test Case 3")
dancer1.dance()
print("-" * 40)

print("Test Case 4")
print(f"Dancer is a subclass of Performer: [{issubclass(Dancer, Performer)}]")
print("-" * 40)

print("Test Case 5")
singer1 = Singer( name: "Cyrile",  age: 21,  vocal_range: "Soprano")
print(f"Singer properties. \n[name:{singer1.get_name()}, age:{singer1.get_age()}, vocal_range:{singer1.get_vocal_range()}]")
print("-" * 40)

print("Test Case 6")
singer1.sing()
print("-" * 40)
```

Test Cases Outputs: