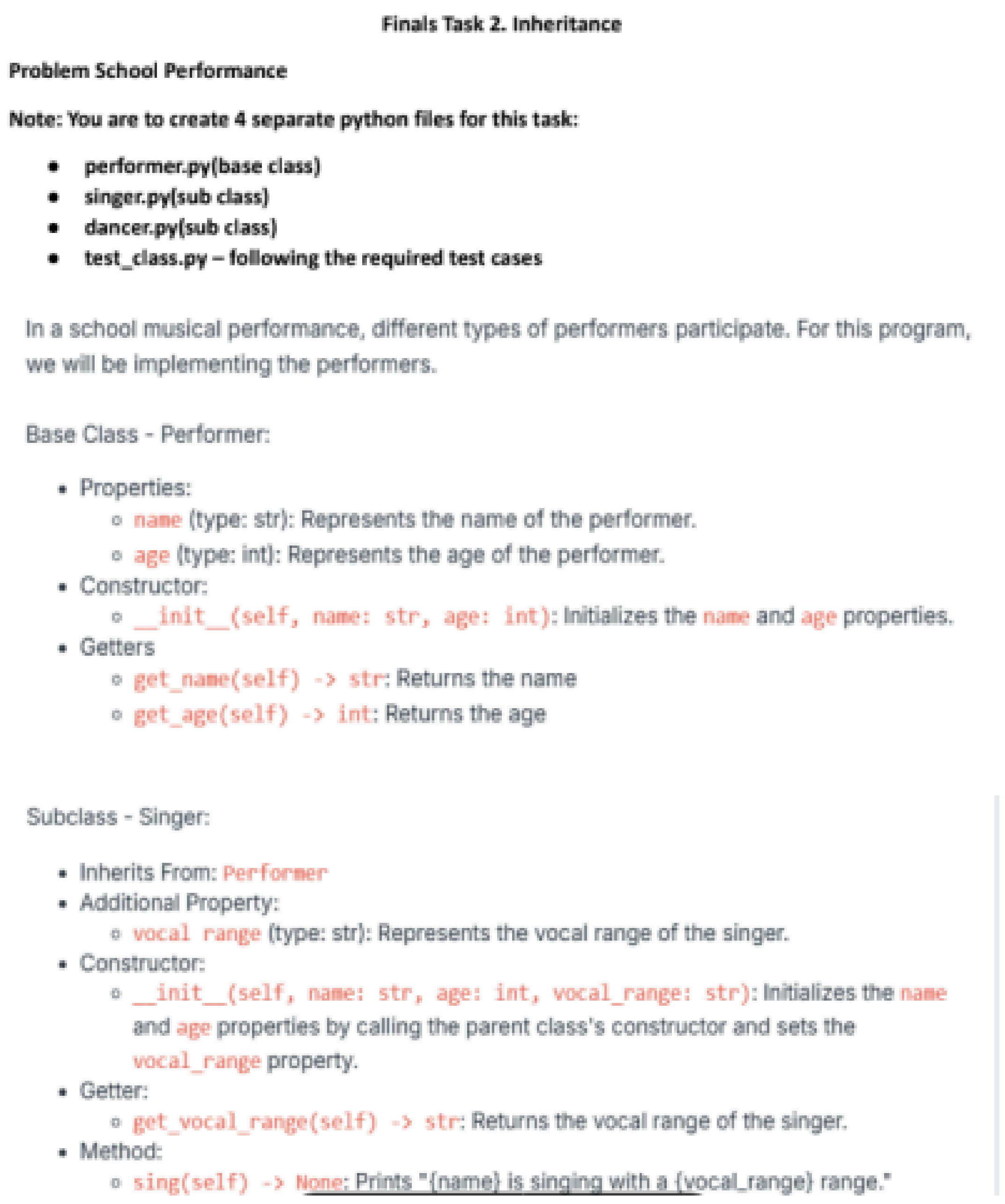
Name: Sicheco, Vincent Rhian S.
Section: C203

Final Task 2. Inheritance.

Requirements.



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**

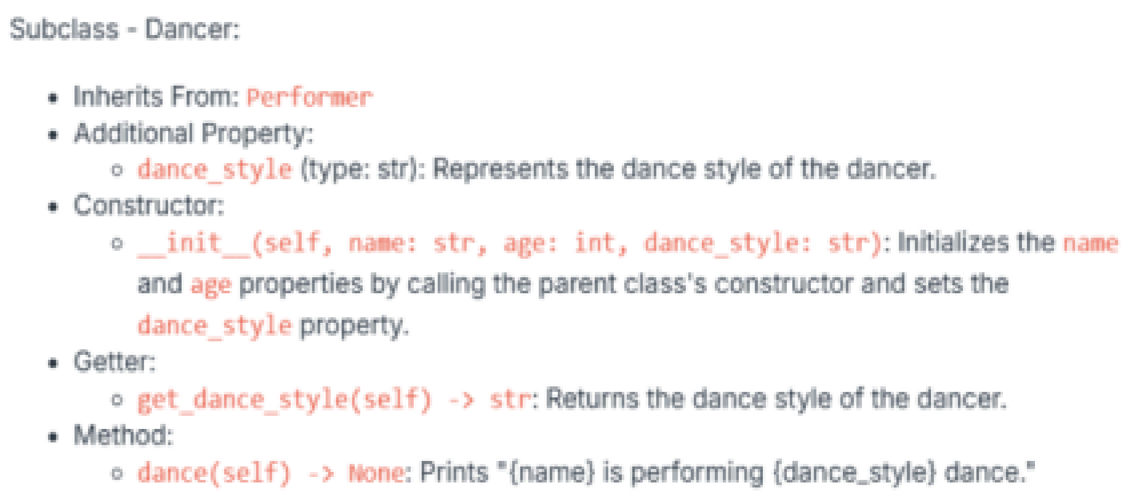
In a school musical performance, different types of performers participate. For this program, we will be implementing the performers.

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."

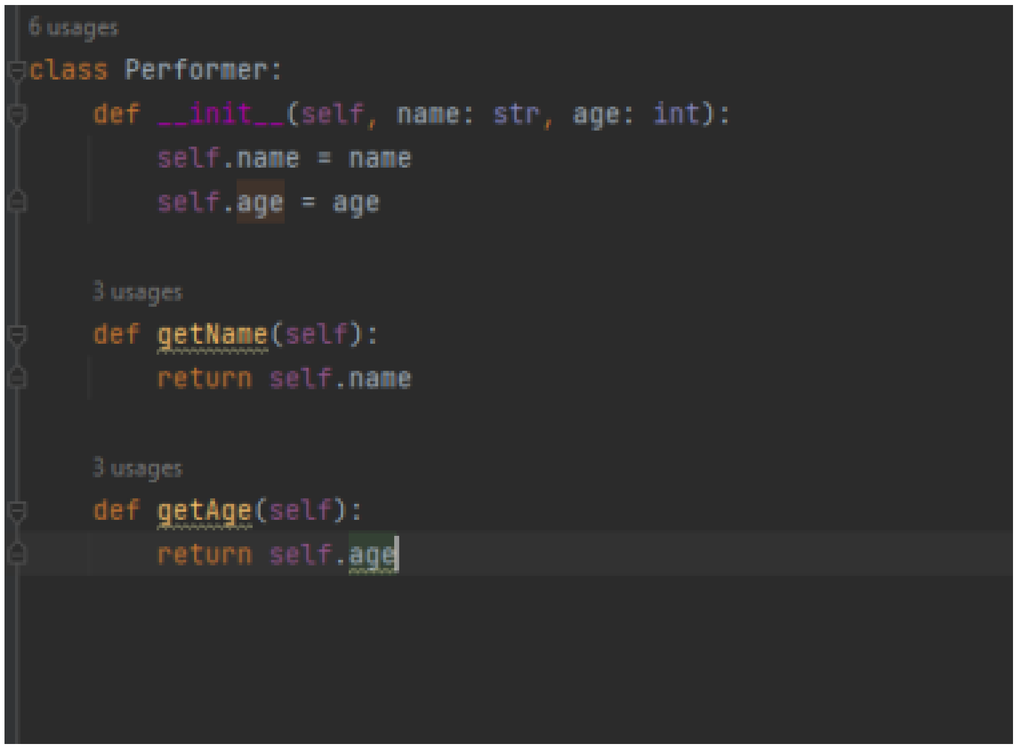


Subclass - Dancer:

- 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."

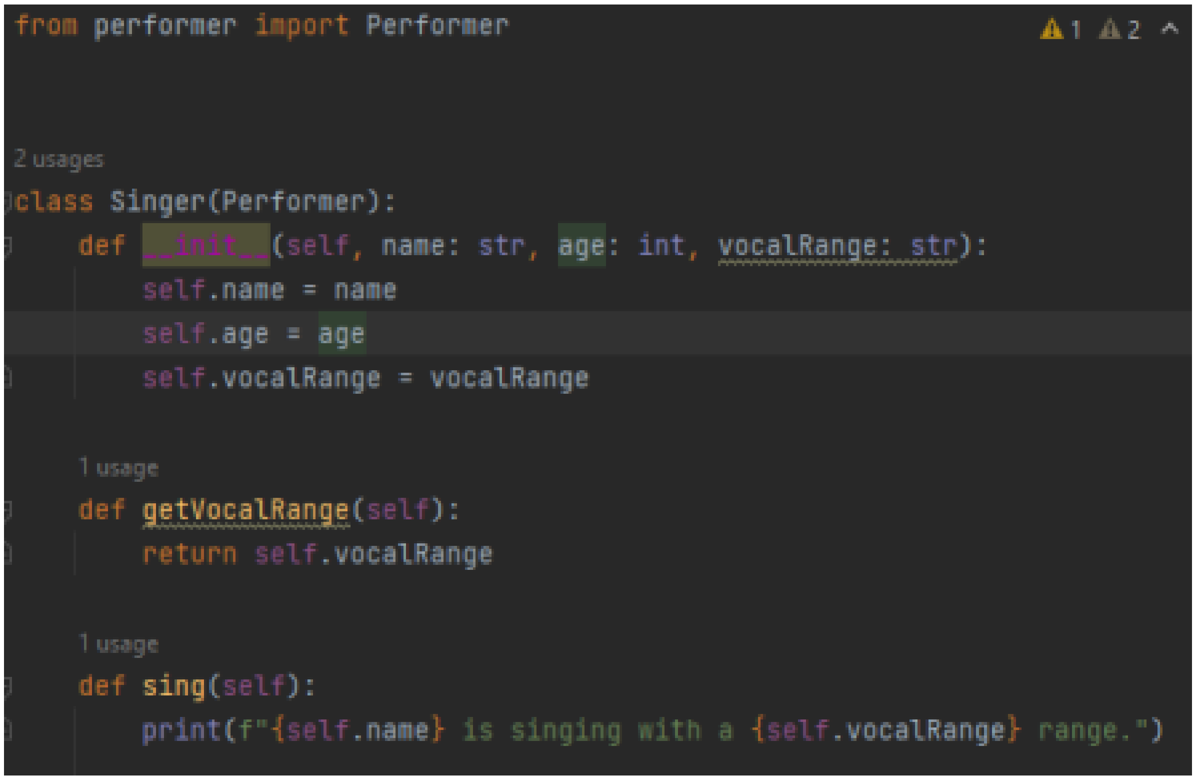
Code.

Performer.



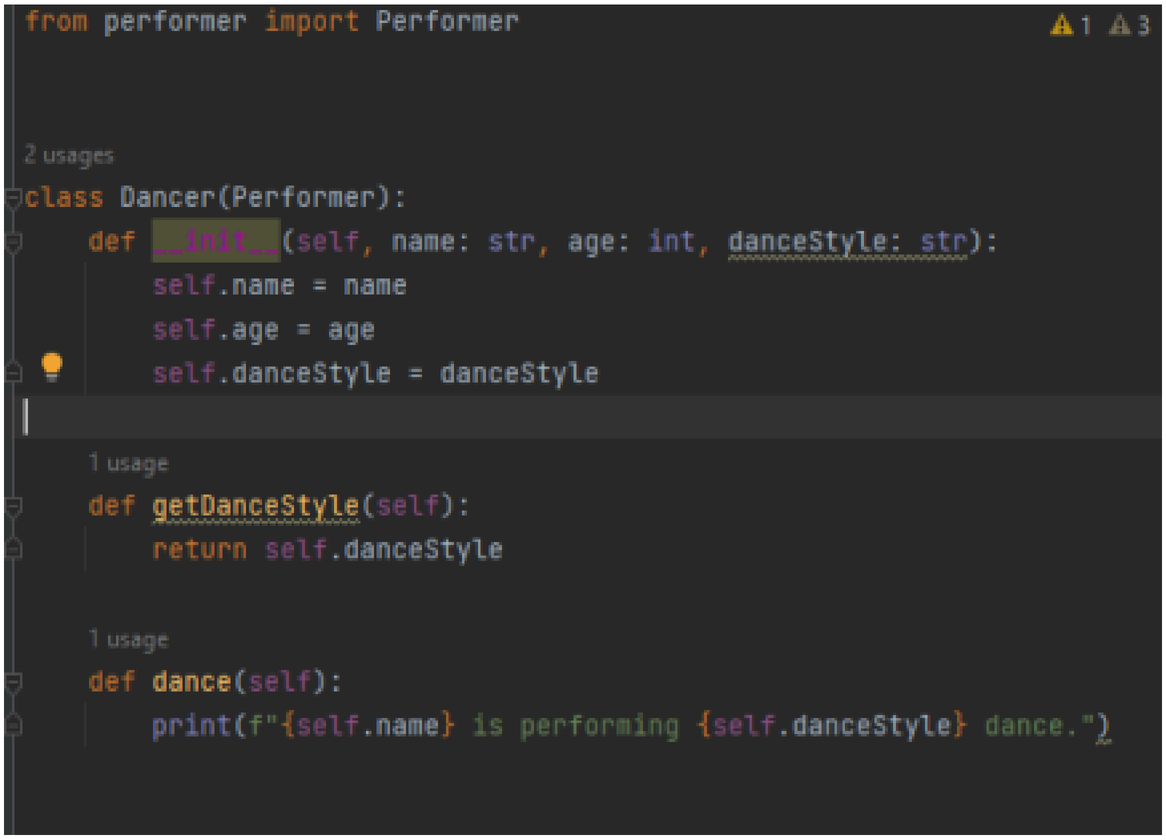
```python
6 usages
class Performer:
    def __init__(self, name: str, age: int):
        self.name = name
        self.age = age

    3 usages
    def getName(self):
        return self.name

    3 usages
    def getAge(self):
        return self.age
```

Singer.



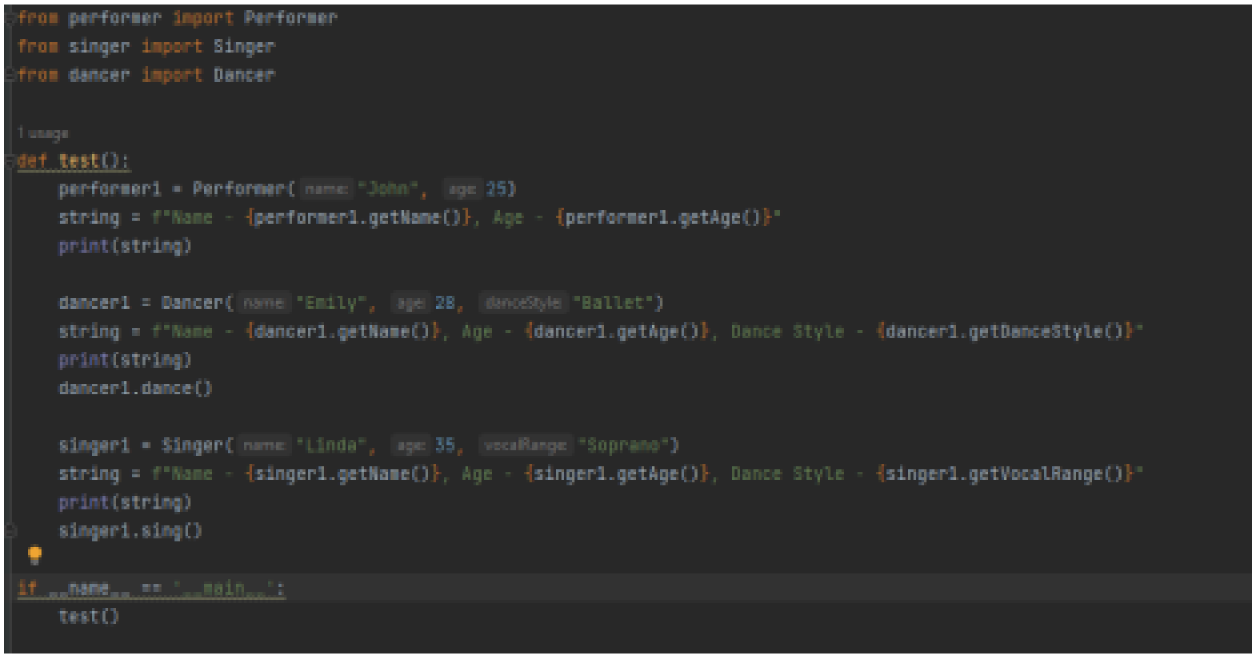
```
from performer import Performer


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

    1 usage
    def getVocalRange(self):
        return self.vocalRange

    1 usage
    def sing(self):
        print(f"{self.name} is singing with a {self.vocalRange} range.")
```

Dancer.



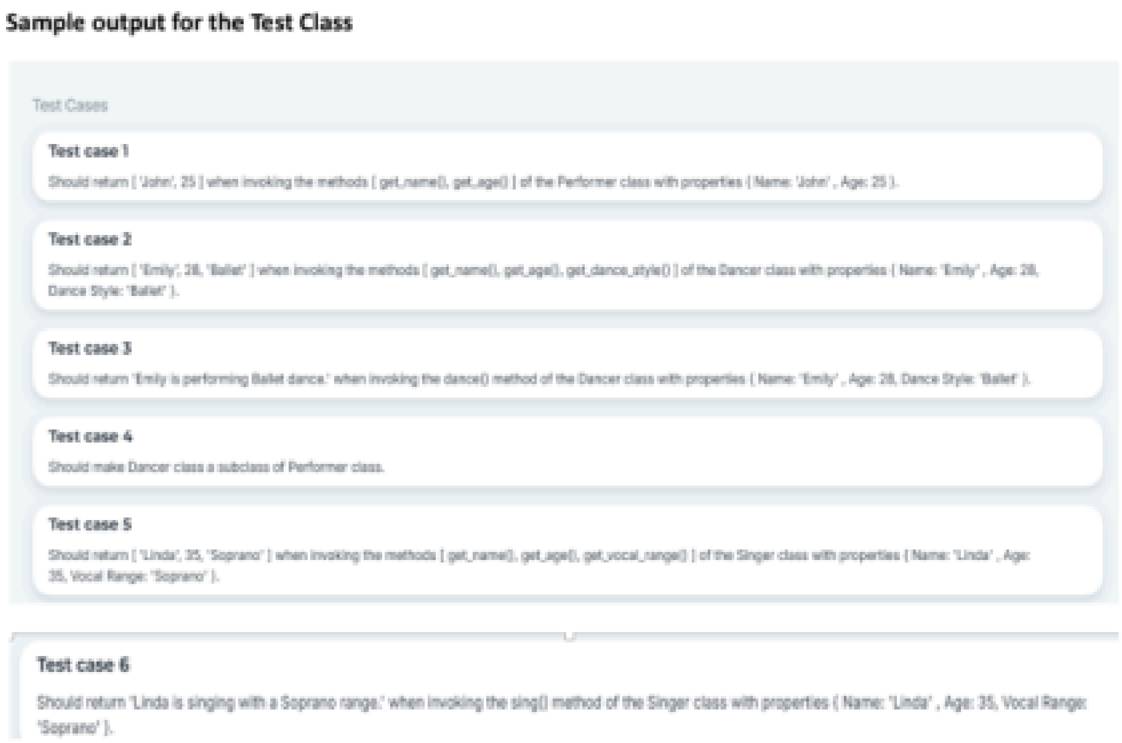
```
from performer import Performer


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

    1 usage
    def getDanceStyle(self):
        return self.danceStyle

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

Test.



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

1 usage
def test():
    performer1 = Performer( name: "John",  age: 25)
    string = f"Name - {performer1.getName()}, Age - {performer1.getAge()}"
    print(string)

    dancer1 = Dancer( name: "Emily",  age: 28,  danceStyle: "Ballet")
    string = f"Name - {dancer1.getName()}, Age - {dancer1.getAge()}, Dance Style - {dancer1.getDanceStyle()}"
    print(string)
    dancer1.dance()

    singer1 = Singer( name: "Linda",  age: 35,  vocalRange: "Soprano")
    string = f"Name - {singer1.getName()}, Age - {singer1.getAge()}, Dance Style - {singer1.getVocalRange()}"
    print(string)
    singer1.sing()

if __name__ == '__main__':
    test()
```

Test Cases.



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.

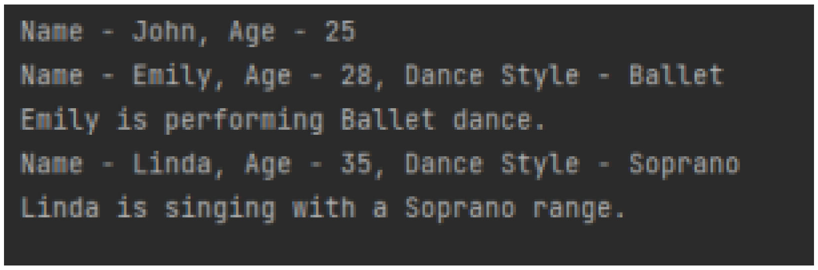
Test case 5

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' }.

Output.



```
Name - John, Age - 25
Name - Emily, Age - 28, Dance Style - Ballet
Emily is performing Ballet dance.
Name - Linda, Age - 35, Dance Style - Soprano
Linda is singing with a Soprano range.
```