

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

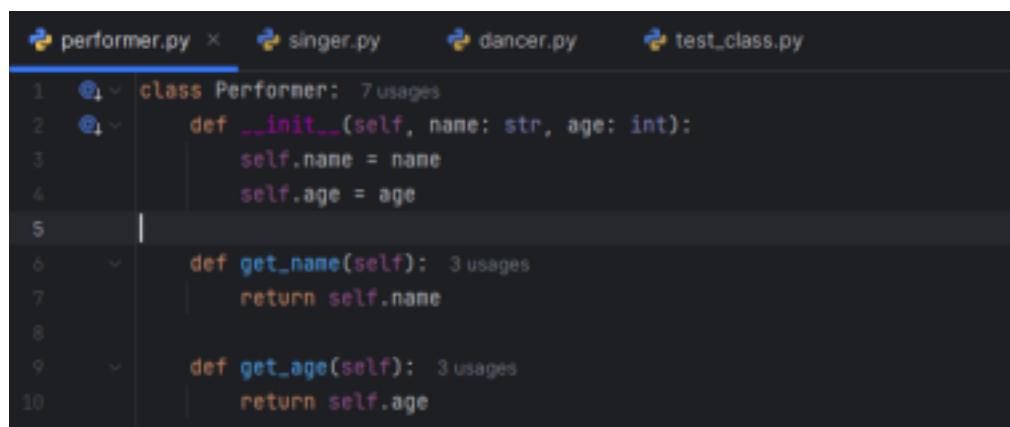
• Getter:

- `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):

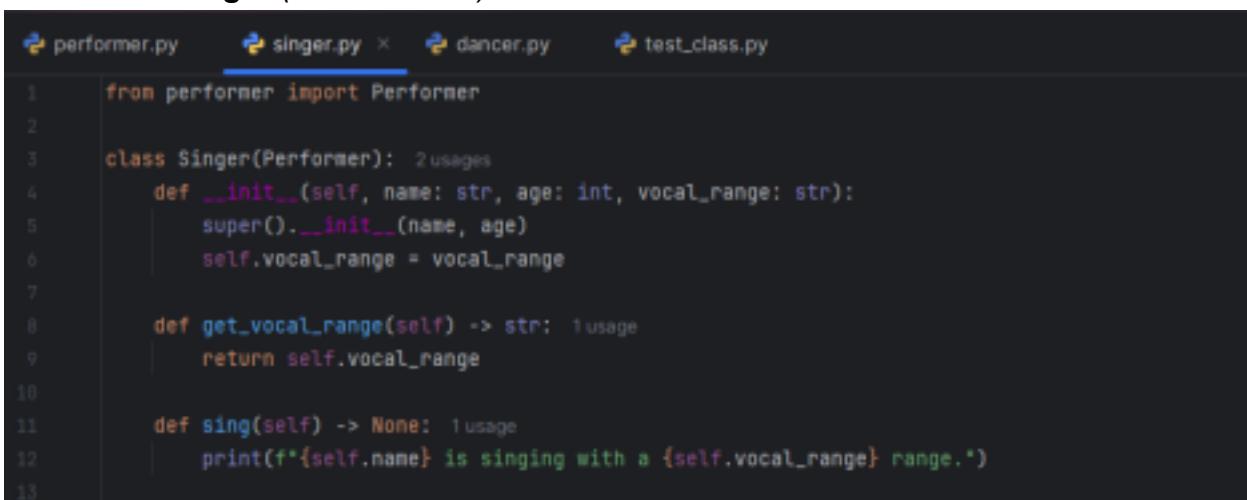


```

1  @4 class Performer: 7 usages
2  @4 def __init__(self, name: str, age: int):
3      self.name = name
4      self.age = age
5
6      def get_name(self): 3 usages
7          return self.name
8
9      def get_age(self): 3 usages
10         return self.age

```

Subclass – Singer (Source Code):



```

1  from performer import Performer
2
3  class Singer(Performer): 2 usages
4      def __init__(self, name: str, age: int, vocal_range: str):
5          super().__init__(name, age)
6          self.vocal_range = vocal_range
7
8      def get_vocal_range(self) -> str: 1 usage
9          return self.vocal_range
10
11     def sing(self) -> None: 1 usage
12         print(f"{self.name} is singing with a {self.vocal_range} range.")

```

Subclass – Dancer (Source Code):

```
performer.py singer.py dancer.py test_class.py
1 from performer import Performer
2
3 class Dancer(Performer): 3 usages
4     def __init__(self, name: str, age: int, dance_style: str):
5         super().__init__(name, age)
6         self.dance_style = dance_style
7
8     def get_dance_style(self) -> str: 1 usage
9         return self.dance_style
10
11    def dance(self) -> None: 1 usage
12        print(f"{self.name} is performing {self.dance_style} dance.")
```

Test Class (Source Code):

```
performer.py singer.py dancer.py test_class.py
1 from performer import Performer
2 from singer import Singer
3 from dancer import Dancer
4
5 print("Test Case 1")
6 performer1 = Performer(name="kurt", age=20)
7 print(f"Performer Properties. \n{name:{performer1.get_name()}}, age:{performer1.get_age()}")
8 print("-" * 40)
9
10 print("Test Case 2")
11 dancer1 = Dancer(name="dexter", age=20, dance_style="hiphop")
12 print(f"Dancer properties. \n{name:{dancer1.get_name()}}, age:{dancer1.get_age()}, dance_style:{dancer1.get_dance_style()}")
13 print("-" * 40)
14
15 print("Test Case 3")
16 dancer1.dance()
17 print("-" * 40)
18
19 print("Test Case 4")
20 print(f"Dancer is a subclass of Performer: [{issubclass(Dancer, Performer)}]")
21 print("-" * 40)
22
23 print("Test Case 5")
24 singer1 = Singer(name="Cyrille", age=21, vocal_range="Soprano")
25 print(f"Singer properties. \n{name:{singer1.get_name()}}, age:{singer1.get_age()}, vocal_range:{singer1.get_vocal_range()}")
26 print("-" * 40)
27
28 print("Test Case 6")
29 singer1.sing()
30 print("-" * 40)
```

Test Cases Outputs:

```
performer.py      singer.py      dancer.py      test_class.py ×
Run   test_class ×

C:\Users\Windows\PycharmProjects\ItemStorage\.venv\Scripts\python.exe C:\Users\Windows\PycharmProjects\ItemStorage\test_class.py
Test Case 1
Performer Properties.
[name:kurt, age:20]
-----
Test Case 2
Dancer properties.
[name:dexter, age:20, dance_style:hiphop]
-----
Test Case 3
dexter is performing hiphop dance.
-----
Test Case 4
Dancer is a subclass of Performer: [True]
-----
Test Case 5
Singer properties.
[name:Cyrile, age:21, vocal_range:Soprano]
-----
Test Case 6
Cyrile is singing with a Soprano range.
-----
Process finished with exit code 0
```