

Finals Task 2. Inheritance

Sample Problem

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' }.

Source Code

performer.py

```
1 class Performer:
2     def __init__(self, name: str, age: int):
3         self.name = name
4         self.age = age
5
6     def get_name(self) -> str:
7         return self.name
8
9     def get_age(self) -> int:
10        return self.age
11
12    def introduce(self) -> None:
13        print(f"Hello! My name is {self.name} and I am {self.age} years old.")
14
```

singer.py

```
from performer import Performer

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

    def sing(self) -> None:
print(f"{self.get_name()} is singing with a {self.vocal_range}
range.")
dancer.py
```

```
1  from performer import Performer
2
3  class Dancer(Performer):
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:
9      return self.dance_style
10
11 def dance(self) -> None:
12     print(f"{self.get_name()} is dancing {self.dance_style}.")
13
```

test_class.py

```

1  ✓ from performer import Performer
2    from singer import Singer
3    from dancer import Dancer
4
5    # Test Case 1
6    performer = Performer("John", 25)
7    print(performer.get_name(), performer.get_age())
8
9    # Test Case 2
10   dancer = Dancer("Emily", 28, "Ballet")
11   print(dancer.get_name(), dancer.get_age(), dancer.get_dance_style())
12
13   # Test Case 3
14   dancer.dance()
15
16   # Test Case 4
17   print(isinstance(dancer, Performer))
18
19   # Test Case 5
20   singer = Singer("Linda", 35, "Soprano")
21   print(singer.get_name(), singer.get_age(), singer.get_vocal_range())
22
23   # Test Case 6
24   singer.sing()

```

Sample Output

```

John 25
Emily 28 Ballet
Emily is dancing Ballet.
True
Linda 35 Soprano
Linda is singing with a Soprano range.
PS C:\Users\Liann>

```