

Final Task 3. Polymorphism

Sample Problem

Finals Task 3. Simple Polymorphism

Problem. Chirp and Tweet

Create a simple program to demonstrate basic polymorphism with bird sounds.

Class - Bird:

- Methods:
 - `def make_sound(self) -> None`: An abstract method that represents making a sound. It doesn't have a specific implementation in the base class `Bird`.

Class - Sparrow (extends Bird):

- Methods:
 - `def make_sound(self) -> None`: Overrides the `make_sound` method from the base class `Bird`. It prints the sound "Chirp Chirp" when called.

Class - Parrot (extends Bird):

- Methods:
 - `def make_sound(self) -> None`: Overrides the `make_sound` method from the base class `Bird`. It prints the sound "Tweet Tweet" when called.

Class - BirdCage:

- Methods:
 - `def make_bird_sounds(self, birds: List) -> None`: Accepts a list of `Bird` objects as input. Iterates through the list of birds and calls the `make_sound` method on each bird to make its sound.

Note:

- *The test cases are not outputs of your main file but of a hidden test file. Create and implement the classes instructed to test your code.*
- *Each class should be defined in its own file, with the file name following camelCase conventions (e.g., `bankAccount.py`).*

TEST CASES:

Test Cases	
Test case 1	Should return "Chirp-Chirp" when invoking the method make_sound() of Sparrow object returned when invoking the Sparrow() constructor of the Sparrow class.
Test case 2	Should return "Tweet Tweet" when invoking the method make_sound() of Parrot object returned when invoking the Parrot() constructor of the Parrot class.
Test case 3	Should return "Chirp-Chirp" when invoking the method make_sound() of Bird object returned after invoking the Sparrow() constructor of the Sparrow class and return "Tweet Tweet" when invoking the method make_sound() of Bird object returned when invoking the Parrot() constructor of the Parrot class.
Test case 4	Should make Bird class an abstract.
Test case 5	Should return "Chirp-Chirp, Tweet Tweet" when invoking the method make_bird_sound(Sparrow(), Parrot()) of BirdCage object returned when invoking the BirdCage() constructor of the BirdCage class.

Source Code

bird.py

```

1  from abc import ABC, abstractmethod
2
3  class Bird(ABC):
4      @abstractmethod
5      def make_sound(self) -> str:
6          """Abstract method for making a bird sound."""
7          pass
8

```

parrot.py

```

1  from bird import Bird
2
3  class Parrot(Bird):
4      def make_sound(self) -> str:
5          return "Tweet Tweet"
6

```

sparrow.py

```

1  from bird import Bird
2
3  class Sparrow(Bird):
4      def make_sound(self) -> str:
5          return "Chirp Chirp"
6

```

birdCage.py

```
1  from typing import List
2  from bird import Bird
3
4  class BirdCage:
5      def make_bird_sounds(self, birds: List[Bird]) -> List[str]:
6          sounds = []
7          for bird in birds:
8              sounds.append(bird.make_sound())
9          return sounds
10
```

testCase

```
1  from sparrow import Sparrow
2  from parrot import Parrot
3  from birdCage import BirdCage
4
5  # Test Case 1
6  print(Sparrow().make_sound())
7
8  # Test Case 2
9  print(Parrot().make_sound())
10
11 # Test Case 3
12 birds = [Sparrow(), Parrot()]
13 for bird in birds:
14     print(bird.make_sound())
15
16 # Test Case 5
17 cage = BirdCage()
18 for sound in cage.make_bird_sounds([Sparrow(), Parrot()]):
19     print(sound)
20
```

Sample Output

```
Chirp Chirp
Tweet Tweet
Chirp Chirp
Tweet Tweet
Chirp Chirp
Tweet Tweet
PS C:\Users\Liann>
```