lecture_16

September 27, 2022

1 Lecture 16 ~ SOLID

1.1 Single Responsibility Principle

[ ]:
```python
# overengineering
```

[6]:
```python
class Animal:
    def walk(self):
        print("walking")

    def talk(self, speech):
        print(speech)
```

[7]:
```python
class CanWalkMixin:
    def walk(self):
        print("walking")
```

[8]:
```python
class Animal(CanWalkMixin):
    def talk(self, speech):
        print(speech)
```

[9]:
```python
class Human(CanWalkMixin):
    def talk(self, tone_of_voice, speech):
        print(speech)
```

1.2 Open/Close Principle

[11]:
```python
import math
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        raise NotImplementedError()


class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width


class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return self.radius * math.pi**2
```

[19]:
```
def area_calculator(*args):
    total_area = 0
    for shape_obj in args:
        total_area += shape_obj.area()
    return total_area
```

[20]:
```
a = Rectangle(10, 14)
```

[21]:
```
area_calculator(a)
```

[21]:
```
140
```

[22]:
```
b = Circle(10)
```

[23]:
```
area_calculator(b)
```

[23]:
```
98.69604401089359
```

[24]:
```
area_calculator(a, b)
```

[24]:
```
238.6960440108936
```

[28]:
```
class Object(ABC):
    @abstractmethod
    def area(self):
        raise NotImplementedError()

    @abstractmethod
    def volume(self):
        raise NotImplementedError()
```

```
class Cube:
    def __init__(self, height, width, length):
        self.height = height
        self.width = width
        self.length = length

    def area(self):
        return 2 * (self.length * self.width + self.length * self.height + self.
 ↪height * self.width)

    def volume(self):
        return self.width * self.height * self.length
```

[29]:
```
c = Cube(10, 10, 20)
```

[30]:
```
area_calculator(c)
```

[30]:
```
1000
```

1.3 Liskov Substitution Principle

[37]:
```
class Square(Rectangle):
    def __init__(self, width):
        self.width = width

    def area(self):
        return self.width**2
```

[38]:
```
d = Square(10)
```

[40]:
```
area_calculator(d)
```

[40]:
```
100
```

1.4 Interface Segragation Principle

[41]:
```
class Animal:
    def walk(self):
        raise NotImplementedError

    def swim(self):
        raise NotImplementedError

    def talk(self):
        raise NotImplementedError
```

[42]:
```python
class Human(Animal):
    def walk(self):
        print("walking")

    def swim(self):
        print("swimming")

    def talk(self):
        print("talking")
```

[43]:
```python
class Whale(Animal):
    pass
```

[52]:
```python
class Animal:
    def talk(self):
        print("talking")
```

[53]:
```python
class Walker:
    def walk(self):
        raise NotImplementedError
```

[54]:
```python
class Swimmer:
    def swim(self):
        raise NotImplementedError
```

[55]:
```python
class Human(Animal, Walker, Swimmer):
    def walk(self):
        print("walking")

    def swim(self):
        print("swimming")
```

[56]:
```python
class Whale(Animal, Swimmer):
    def swim(self):
        print("swimming")
```

1.5 Dependency Inversion Principle

[ ]:
```python

```