HW_Solution#2

May 17, 2023

1 Homework #2

1.0.1 Assignment

Define a set of classes that, starting from a generic 3D shape, derive a rectangular parallelepiped, a sphere and a cone solid definitions and provides their full area and volume. Area must be calculated in the constructor and accessed via getter and setter, whereas Volume must be calculated implementing an abstract method defined in the parent class. Results should be displayed using polymorphism by iterating over the object instances (use the examples provided herein).

1.0.2 A possible solution

[5]:
```
from math import pi

# base class definition
class Shape3D:

    verbose = False

    # constructor for the base class: it simply assigns the name of the shape
    # that is passed by the derived classes constructor with the super object
    def __init__(self, name):
        self.name = name

    def volume(self):
        pass

    # defines a printable version of class information
    def __str__(self):
        return self.name

class RParallelepiped(Shape3D):

    __area = 0.0

    @property
    def area(self):
```

```
        if self.verbose: print('area property value is {:.2f}'.format(self.
 ↪__area))
        return self.__area

    @area.setter
    def area(self,x):
        self.__area = x
        if self.verbose: print('area setter value is {:.2f}'.format(self.
 ↪__area))

    def __init__(self, base_w, base_l, height):
        super().__init__("Rectangular Parallelepiped")
        self.base_w = base_w
        self.base_l = base_l
        self.height = height
        if self.verbose: print('__init__ calling area setter')
        self.area = 2*(base_w*base_l+base_l*height+base_w*height)

    def volume(self):
        return self.base_w*self.base_l*self.height

class Sphere(Shape3D):

    __area = 0.0

    @property
    def area(self):
        return self.__area

    @area.setter
    def area(self,x):
        self.__area = x

    def __init__(self, radius):
        super().__init__("Sphere")
        self.radius = radius
        self.area = 4*pi*radius*radius

    # area is defined according to the circle shape
    def volume(self):
        return 4/3*pi*self.radius**3

class Cone(Shape3D):

    __area = 0.0

    @property
```

```
    def area(self):
        return self.__area

    @area.setter
    def area(self,x):
        self.__area = x

    def __init__(self, radius, height):
        super().__init__("Cone")
        self.radius = radius
        self.height = height
        self.base_area = pi*radius*radius
        self.area=self.base_area+pi*radius*(radius**2+height**2)**0.5
        # import numpy as np
        # self.area=self.base_area+pi*radius*np.sqrt(radius**2+height**2)
    # area is defined according to the circle shape
    def volume(self):
        return self.base_area*self.height/3

# create three instances of our classes
a = RParallelepiped(4.3, 3.2, 5.1)
b = Sphere(5)
c = Cone(5.2,6.4)

# iterate over the objects regardless of their classes and
# obtain the correct results due to polymorphism
print('Printing results')
for obj in (a,b,c):
    print('{} has area {:.2f} and volume {:.2f}'.format(obj,obj.area,obj.
 ↪volume()))
```

```
Printing results
Rectangular Parallelepiped has area 104.02 and volume 70.18
Sphere has area 314.16 and volume 523.60
Cone has area 219.66 and volume 181.22
```

[ ]:

[2]:
```
from math import pi

# base class definition
class Shape3D:

    verbose = False
    __area = 0.0

    @property
    def area(self):
```

```
        if self.verbose: print('area property value is {:.2f}'.format(self.
 ↪__area))
        return self.__area

    @area.setter
    def area(self,x):
        self.__area = x
        if self.verbose: print('area setter value is {:.2f}'.format(self.
 ↪__area))

    # constructor for the base class: it simply assigns the name of the shape
    # that is passed by the derived classes constructor with the super object
    def __init__(self, name):
        self.name = name

    def volume(self):
        pass

    # defines a printable version of class information
    def __str__(self):
        return self.name

class RParallelepiped(Shape3D):

    def __init__(self, base_w, base_l, height):
        super().__init__("Rectangular Parallelepiped")
        self.base_w = base_w
        self.base_l = base_l
        self.height = height
        if self.verbose: print('__init__ calling area setter')
        self.area = 2*(base_w*base_l+base_l*height+base_w*height)

    def volume(self):
        return self.base_w*self.base_l*self.height

class Sphere(Shape3D):

    def __init__(self, radius):
        super().__init__("Sphere")
        self.radius = radius
        self.area = 4*pi*radius*radius

    # area is defined according to the circle shape
    def volume(self):
        return 4/3*pi*self.radius**3

class Cone(Shape3D):
```

```
    def __init__(self, radius, height):
        super().__init__("Cone")
        self.radius = radius
        self.height = height
        self.base_area = pi*radius*radius
        self.area=self.base_area+pi*radius*(radius**2+height**2)**0.5

    # area is defined according to the circle shape
    def volume(self):
        return self.base_area*self.height/3

# create two instances of our classes
a = RParallelepiped(4.3, 3.2, 5.1)
b = Sphere(5)
c = Cone(5.2,6.4)

# iterate over the objects regardless of their classes and
# obtain the correct results due to polymorphism
print('Printing results')
for obj in (a,b,c):
    print('{} has area {:.2f} and volume {:.2f}'.format(obj,obj.area,obj.
 ↪volume()))
```

```
Printing results
Rectangular Parallelepiped has area 104.02 and volume 70.18
Sphere has area 314.16 and volume 523.60
Cone has area 219.66 and volume 181.22
```

[3]:
```
a.area
```

[3]:
```
104.01999999999998
```

[4]:
```
b.area
```

[4]:
```
314.1592653589793
```

[5]:
```
c.area
```

[5]:
```
219.66109612458598
```

[ ]: