CPSC 250 Test 3

Question 1: Vehicle Collection

The goal of this question is to implement a program to manage an antique car collection.

Step 1. You are given templates for a base Vehicle class and derived Car, Truck classes. You will need to complete the \_\_init\_\_ and display\_info() functions for these classes.

Step 2. Write a main program to create a list called my\_collection.

Step 3. Given some input data (example below), store objects that belong to the appropriate classes in the list created in Step 1.

Step 4. Create a function (called by the main program) with the name display\_list() that uses the display\_info() instance methods defined in the respective classes and prints each element in my\_collection.

In summary, the main program should read vehicles from input (ending with done), add each vehicle to the my\_collection list, and output each element in my\_collection using the display\_info() function.

(See the next page for an input/output example)

Ex. If the input is:

vehicle Vespa 7800

car Chevrolet 27000 BelAir 1957

truck Fargo 22500 Pickup 1948 false

vehicle Scooter 200

done

the output is:

Vehicle 1 Information:

Vehicle Make: Vespa

Value: 7800

Vehicle 2 Information:

Description: 1957 Chevrolet BelAir

Value: 27000

Vehicle 3 Information:

Description: 1948 Fargo Pickup

Value: 22500

All Wheel Drive: false

Vehicle 4 Information:

Vehicle Make: Scooter

Value: 200

Question 2: Shapes

This question is all about creating classes that represent various geometrical shapes.

Step 1: Create a base class called Shape.

It should have one private instance variable of type integer and named number\_of\_sides.

It should have a constructor (i.e. \_\_init\_\_) that takes one argument, the number of sides.

Define the number\_of\_sides instance variable in the constructor body.

Create a method named calculate\_area that takes no arguments and just passes (no implementation).

Create another method named

calculate\_perimeter that takes no arguments and just passes (no implementation).

Create a setter method named set\_number\_of\_sides that takes one argument (the number of sides), and returns nothing.

Create a getter method named get\_number\_of\_sides that takes no arguments, and returns the number of sides.

Step 2: Create a derived class (of the base Shape class) called Rectangle

It should have two private instance variables,

a float named width and a float named height.

Create a constructor (\_\_init) that takes three arguments:

number\_of\_sides, width, and height. Fulfill the constructor requirement of the parent class. Define the width and height instance variables in the constructor body.

Implement the calculate\_area method.

- Area of a rectangle is calculated as: area = width \* height

Implement the calculate\_perimeter method.

- Perimeter of a rectangle is calculated as: perimeter = 2(width) + 2(height)

Create appropriate getters and setters: set\_width, get\_width, set\_height, get\_height

Create an appropriate \_\_eq\_\_ method (one rectangle would be equal to another rectangle if (and only if) BOTH the heights and widths are equal to one another.

Create a \_\_str\_\_ method to return a string in the following format (where x is the appropriate value):

"A rectangle has x sides | Area = x | Perimeter = x"

Step 3. Create a class named Circle that is a child of Shape

It should have one private instance variable, a float

named radius.

Create a constructor that takes two arguments, number\_of\_sides, and radius. Fulfill the

constructor requirement of the parent class. Define the radius instance variable in the constructor body.

Implement the calculate\_area method.

- Area of a circle is calculated as: area = PI \* radius^2

Implement the calculate\_perimeter method.

- Perimeter (circumference) of a circle is calculated as: perimeter = 2 \* PI \* radius

Create appropriate getters and setters: get\_radius, set\_radius

Create an appropriate \_\_eq\_\_ method. One circle is considered equal to another circle if (and only if) the radii are equal to one another.

Create a \_\_str\_\_ method to return a string in the following format (where x is the appropriate value):

"A circle has x sides | Area = x | Perimeter = x"

You may find the following main program helpful to test your classes. With that said, you should think about adding additional code to your main program to test all aspects of the classes (such as “==” for example!)

if \_\_name\_\_ == '\_\_main\_\_':

rectangle = Rectangle(4, 4.1, 2.3)

print(rectangle) # 4 sides, area=9.43, perimeter=12.8

circle = Circle(1, 3.5)

print(circle) # 1 side, area=38.48, perimeter=21.99

print()

shapes = [

Circle(1, 3.2), # 1 side, area=32.17, perimeter=20.11

Rectangle(4, 2, 2.6), # 4 sides, area=5.20, perimeter=9.20

Circle(1, 8.3), # 1 side, area=216.42, perimeter=52.15

]

for shape in shapes:

print(shape)