Chapter 9- End of Chapter assignments

Chapter 9 Review Questions

1. Hiding the implementation of a class is referred to as _____________.
2. The data and methods of a class are called ____________.
3. Program statements and other objects access an object's attributes through the ________________.
4. An object is an ___________ of a class.
5. The ________________ declares the data and methods for a class.
6. The method that creates and initializes an object is called the _____________.
7. The values stored in an object's data attributes are referred to as the object's __________.
8. Two underscores preceding a class data attribute indicate that the element is __________.
9. Methods that access an object's data attributes are called ______________.
10. Methods that set or change an object's data attributes (state) are called __________.
11. The ________ function provides a way to output an objects state.
12. _________ refers to the degree to which an object represents a single abstraction without external dependencies.
13. _________ refers to the degree to which an object is dependent upon another.
14. A way in Python to serialize objects into byte streams is called __________.
15. A __________ diagram can be used to document the data and method attributes of a class.
16. ______________ allows a base (super) class to contain common elements for derived (sub) classes.
17. ______________ is the ability to take on many forms.
18. The __________ function can be used to determine if an object is an instance of or derived class of another class.

Chapter 9 Short Answer Exercises

1. What is the difference between a class and an object?

2. What is an instance of a class?

3. What is the object reference in the following statement?

   account.get_balance( )

4. What parameters need to be passed to the following constructor?

   def __init__(self, name, age)

5. What change is needed to indicate that the following data attributes are private?

   name
   account
   phone

6. What is the __str__ method used for and how is it called?

7. In the following statement, what is the base (super) class? What is the derived (sub) class?

   class Car(Vehicle)

Chapter 9 Programming Exercises

1. Write a class definition for a Circle Class that has a data attribute for radius, a constructor that accepts a radius and initializes the instance attribute, and the following methods:

   get_circumference( )

   get_area( )

2. Using the Circle Class from #1, write a program that creates a Circle object with a radius of 6, and displays the circumference and area of the circle.

3. Implement a Product class that has data attributes for description, price, and inventory. Write a constructor that accepts parameters for the attributes and initializes them, and a method to display a product's information. Write a program to create the product objects below and display them.

Product	Price	Inventory
Mug	8.50	23
T-shirt	12.95	45
Towel	18.50	36

4. Implement a Gas Pump class that has data attributes for gallons pumped, price per gallon, and total sale. The constructor will accept a dollar amount for the gas purchased. The class will have a method to allow setting the price per gallon, and a method to display the gallons pumped and the sale.

   Write a main program that will prompt for the price per gallon and the amount in dollars to pump. It will then create a Gas Pump object and display the results. The program will have a loop to create a new pump without restarting the program. Sample main and output shown below.

```
def main():

     keep_going = 'y'

     while keep_going == 'y':
         ppg = float(input('Enter price per gallon: '))
         dollars = float(input('Enter amount: $'))

         g1 = GasPump(dollars)
         g1.set_price_per_gallon(ppg)
         g1.display_result()
         keep_going = input('Enter y for another.')

main()
```

```
Enter price per gallon: 2.95
Enter amount: $20.00
GALLONS PUMPED:  6.67 Price: $20.00
Enter y for another.
```

5. Write a program using the class definition below that creates a Date object, and sets the day, month, and year, then displays the date in mm/dd/yyyy format.

```
class Date:
    def __init__(self):
        self.__day = 1

    def set_day(self, day):
        self.__day = day

    def set_month(self,month):
        self.__month = month

    def set_year(self, year):
        self.__year = year


    def get_date(self):
        return self.__month, self.__day, self.__year
```

6. Create a UML diagram for the Date Class in #5.

7. Implement an Oven Class for a microwave oven that accepts a time to cook in minutes that is greater than 0 and less than 12, three power levels (1, 2, 3 – for low, medium, high), and start. The Class methods will validate the input, and use default values when invalid data is entered and display an error. Write a program that creates an object and prompts for the input. The program will have a loop to create a new object without restarting the program. Sample output shown below.

```
Enter minutes: 3
Enter power level: 2

Enter "y" to start: y
Cooked:  3 minutes  at  MED  power

Enter "y" for another:
```

8. Implement the *Business* Class from Ex. 9.9 with the data attributes for name and employees and methods for accessing them. Then implement a derived class *Restaurant* that inherits from the *Business* Class and has data elements for tables, and seats. Write a program that creates an instance of the *Restaurant* Class named Sally's with 14 employees, 15 tables and 65 seats. Add a method to the program that displays all of the information for Sally's Restaurant.

Chapter 9 Programming Challenges

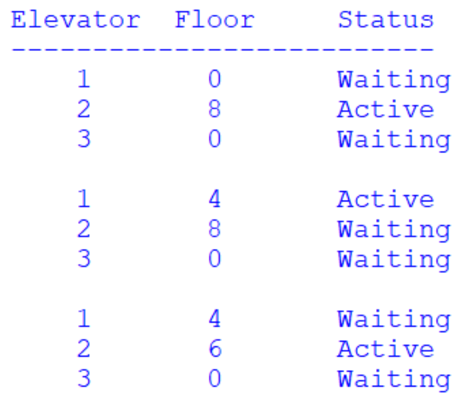
#1 – Product Pickling

Using the Product Class from #3, create the three products, and *dump* them (serialize them and write them to a file), and close the file. Open the file and *load* the products (retrieve and de-serialize) them into object references (names) that are different from the names they were given when they were created. Then display the information for the products.

#2 – Elevator Class

Implement an Elevator Class for elevators that can travel to floors 1 - 8, “know” what elevator they are (1, 2, 3), “know” what floor they are on, and “know” whether they are active or waiting.

Write a program that creates three (3) elevators, and starts them each at the ground floor. Using a loop with 10 iterations, randomly select one of the elevators to move (this one is active and the others are waiting) and send it to a random floor. The same elevator cannot be moved two times in a row. Display the current state for each of the elevators at each execution of the loop in columns and sets of three as shown. Only one elevator should be active at each interval and the others should remain on their current floors.



```
Elevator  Floor     Status
--------------------------
    1       0       Waiting
    2       8       Active
    3       0       Waiting

    1       4       Active
    2       8       Waiting
    3       0       Waiting

    1       4       Waiting
    2       6       Active
    3       0       Waiting
```