PROGRAMACIÒN ORIENTADA A OBJETOS

TALLER No. 2 PYTHON	PROGRAMACIÓN ORIENTADA A OBJETOS

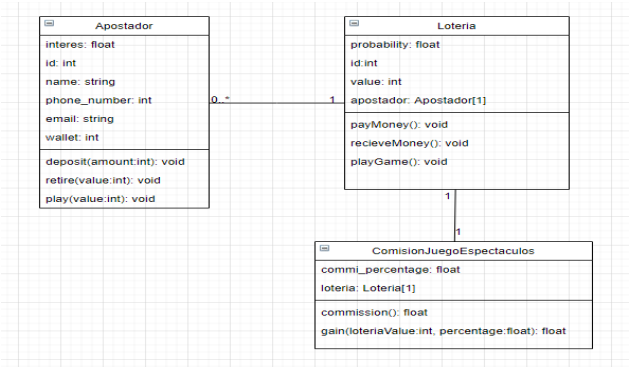
Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1. Clases y objetos
2. Atributos y Métodos

Ejercicio 1 – Clases y Objetos.

a) Crear una nueva carpeta llamada Taller1 y ábrela en VSCode.
b) Analiza el siguiente diagrama de clases.



c) Crea un archivo “main.py” dentro de la carpeta y agrega las siguientes clases y metodos

```
1  import random
2  class Apostador:
3      interes = 0.15
4      def __init__(self, id, name, phone_number, email):
5          self.id = id
6          self.name = name
7          self.phone_number = phone_number
8          self.email = email
9          self.wallet = 0
10
11     def deposit(self, amount):
12         self.wallet += amount
13
14     def retire(self, value):
15        if(value <= self.wallet):
16            self.wallet -= value
17            realValue = value-(value*self.interes)
18            print("Has retirado "+ str(value)+ " que despues de interes
19            quedo en " + str(realValue))
20        else:
21            print("No tienes fondos suficientes para retirar")
```

```
21      def play(self, value):
22          if(self.wallet >= value):
23              loteria = Loteria(value, self)
24              loteria.playGame()
25          else:
26              print("Necesitas poner mas dinero en tu wallet")
27
28  class ComisionJuegoEspectaculos:
29      COMMIPERCENTAJE = 0.20
30      def __init__(self, loteria):
31          self.loteria = loteria
32
33      def commission(self):
34          loteriaValue = self.loteria.value
35          commission = self.gain(loteriaValue, self.COMMIPERCENTAJE)
36          return commission
37
38      @staticmethod
39      def  gain(loteriaValue, percentage):
40          gain = loteriaValue-(loteriaValue*percentage)
41          return gain
42
43  class Loteria:
44      probability = 0.5
45      def __init__(self, value, apostador):
46          self.value = value
47          self.apostador = apostador
48
49      def payMoney(self, gain):
50          self.apostador.wallet += gain
51
52      def recieveMoney(self):
53          self.apostador.wallet -= self.value
54
55      def playGame(self):
56          a = random.randint(0,1)
57          if (a < self.probability):
58              commi = ComisionJuegoEspectaculos(self)
59              gain = commi.commission()
60              total = gain + self.value
61              print("Has ganado "+ str(total))
62              self.payMoney(gain)
63          else:
64              print("Has perdido lo que apostaste")
65              self.recieveMoney()
66
67
68
69
70
```

```
71  if __name__ == "__main__":
72      apostador1 = Apostador(1, "Juan", 302, "j@gmail.com")
73      apostador1.deposit(500)
74      print(apostador1.wallet)
75      apostador1.play(400)
76      print(apostador1.wallet)
77      apostador1.retire(400)
78      print(apostador1.wallet)
79
80
81      apostador2 = Apostador(2, "Ricardo", 548, "r@gmail.com")
82      apostador2.deposit(500)
83      print(apostador2.wallet)
84      apostador2.play(400)
85      print(apostador2.wallet)
86      apostador2.retire(400)
87      print(apostador2.wallet)
88
```

Luego de esto corre el archivo main.py y mira lo que imprime.

Responder:

a) ¿Cuántas clases se están definiendo en este ejercicio?

b) ¿Cuántos objetos de la clase `Apostador` se están creando?

c) ¿Cuáles objetos de la clase `Apostador` se están creando?

d) ¿A quién está haciendo referencia la variable `self` de la línea 23 de la clase `Apostador` cuandose ejecuta el programa principal?

e) ¿Cuántos objetos de la clase `Loteria` se están creando?
- En la línea 73 del main.py cambiar el apostador1.deposit(500) por apostador1.deposit(300)

f) ¿Qué imprimiría el código por parte del apostador1?
- En la línea 82 del main.py cambiar el apostador2.deposit(500) por apostador2.deposit(400)

g) ¿Qué imprimiría el código por parte del apostador2?

h) ¿ Cuáles atributos de la clase `Lotería` están haciendo referencia a objetos?

i) ¿ Cuáles atributos de la clase `Lotería` están haciendo referencia a tipos primitivos?

j) ¿Complete las siguientes líneas para que en la clase Loteria, se implemente el método de clase changeProbability?

```
- _________________
- def changeProbability(___, nprobability):
-         _______.probability = nprobability
```

k) ¿Cómo seria la línea de código para llamar el metodo changeProbability?

l) ¿Es correcto si en el método changeProbability que se creo, cambiar lo siguiente? Explique:

Linea Original

```
- cls.probability  = nprobability
```

Linea Nueva

```
- Loteria.probability  = nprobability
```

m) ¿Cuántos métodos tiene la clase `Loteria` después de agregarle el nuevo metodo?

n) ¿Si el apostador1 gana el apostador2 también? Explique porque pasa en caso de ser sí o no

o) Cambiar el atributo de clase probability a una constante y ejecuta el método changeProbability ¿Es correcto el uso del método changeProbability teniendo en cuenta lo visto en clase?

p) ¿Cuál es el tipo de retorno de los métodos `gain()` y `commission()` de la clase `ComisionJuegoEspectaculos`?

q) ¿A quién está haciendo referencia la variable `self` de la línea 58 de la clase `Loteria` cuandose ejecuta el programa principal? ¿Podría omitirse el uso de la variable `self` en este caso?

r) ¿En la línea 23 de la clase apostador vemos como la clase recibe dos parámetros(value, self) especificar cual de estos pasa por valor y cual por referencia y por qué?

-Realiza el siguiente cambio en el código:

Línea original

```
apostador1 = Apostador(1, "Juan", 302, "j@gmail.com")
apostador1.deposit(500)
print(apostador1.wallet)
apostador1.play(400)
print(apostador1.wallet)
apostador1.retire(400)
print(apostador1.wallet)
```

Línea nueva

```
apostador1 = Apostador(1, "Juan", 302, "j@gmail.com")
apostador1.deposit(500)
print(apostador1.wallet)
apostador1.play(400)
print(apostador1.wallet)
apostador1.interes = 0.10
apostador1.retire(400)
print(apostador1.wallet)
```

s) ¿Qué valor tiene el atributo interés en el objeto apostador2?

t) ¿Después de realizar los cambios en el código el interés de ambos apostadores es el mismo? Explique que ocurre con el atributo de clase interés

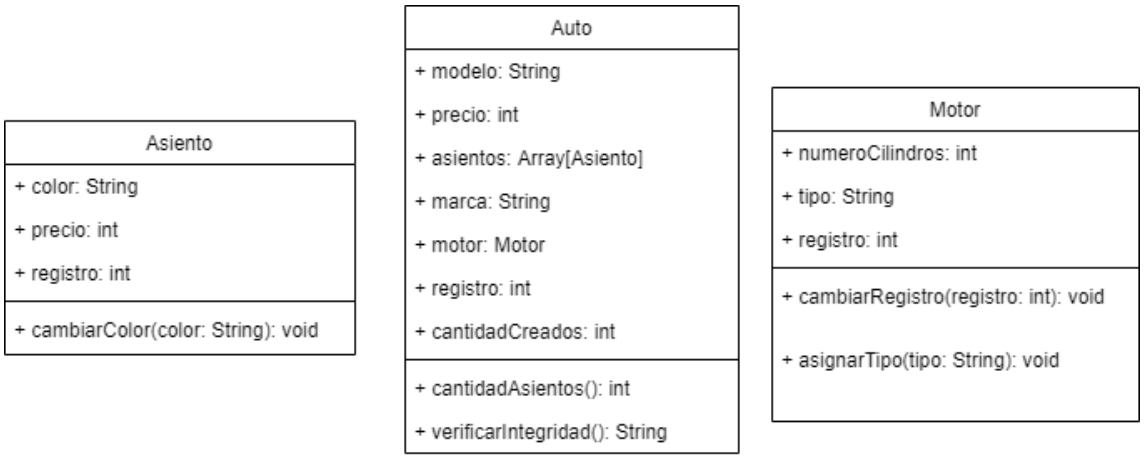
PROGRAMACIÒN ORIENTADA A OBJETOS

Ejercicio 2 – GitHub

Enlace entrega: https://classroom.github.com/a/ePVUHajC

PROBLEMA – Componentes de un Auto

Se le ha contratado para el diseño de la estructura de componentes de un Auto, y se le entrego un diagrama que mostraba un poco las Clases, los atributos y métodos, además se le proporciono unas ciertas instrucciones.



Cree las clases Auto, Asiento y Motor, con sus respectivos atributos y métodos…

Tener en cuenta para la Clase Asiento que:

- El método de instancia cambiarColor(), recibirá un argumento String que será el valor a asignar al atributo color del objeto, tenga en cuenta que los únicos valores permitidos para cambiar el color serán rojo, verde, amarillo, negro y blanco, cualquier otro color no cambiara el color del Asiento.

Tener en cuenta para la Clase Auto que:

- cantidadCreados es un atributo de clase
- El método de instancia cantidadAsientos() retornara la cantidad de asientos que efectivamente sean objetos Asiento en la lista del objeto Auto.
- El método verificarIntegridad(), se encargara de revisar que el atributo registro de Motor, Auto y Cada Asiento sean el mismo, esto para ir en contra de la piratería de piezas. En caso de encontrar que un Asiento, el Auto o el Motor tiene un registro diferente al de los demás retornara el mensaje “Las piezas no son originales” en caso contrario, retornara “Auto original”

Tener en cuenta que para la Clase Motor que:

- El método cambiarRegistro(), recibirá como argumento un int que cambiara el numero del registro del objeto
- El método asignarTipo(), recibirá como argumento un String que cambiara el tipo de motor, este valor solo podrá ser cambiado por el valor electrico o gasolina, en caso contrario no modificara el valor

IMPORTANTE: cree un archivo main.py y agregue alli todas las clases y métodos

requeridos. Y pegue este archivo en el directorio principal del repositorio.