Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Programación Orientada a Objetos

Lista final

nombre_de_la_materia	grupo	clave_unica	nombre_del_alumno	correo_electronico
ALGORITMICA Y PROGRAMACION	001	000158037	CASAS NADER RICARDO	rcasasna@itam.mx
ALGORITMICA Y PROGRAMACION	001	000165280	LUNA VERA JOSE OMAR	jlunaver@itam.mx
ALGORITMICA Y PROGRAMACION	001	000165887	LOPEZ SOLORIO BRANDON ZAYD	blopezso@itam.mx
ALGORITMICA Y PROGRAMACION	001	000172367	CALVILLO SURIANO JOSE MAURICIO	jcalvil6@itam.mx
ALGORITMICA Y PROGRAMACION	001	000174423	RAMIREZ ORTIZ ALONSO DAVID	arami142@itam.mx
ALGORITMICA Y PROGRAMACION	001	000174724	RIVEROS MCKAY AGUILERA EDUARDO	eriveros@itam.mx
ALGORITMICA Y PROGRAMACION	001	000175611	BOBADILLA FRANCO RODRIGO	rbobadil@itam.mx
ALGORITMICA Y PROGRAMACION	001	000177513	HERRERA ABSALON PEDRO DAVID	pherrer1@itam.mx
ALGORITMICA Y PROGRAMACION	001	000179080	OCHOA SAWAYA RENE	rochoasa@itam.mx
ALGORITMICA Y PROGRAMACION	001	000179081	VILLELA FRANYUTTI RODRIGO	rvillel1@itam.mx
ALGORITMICA Y PROGRAMACION	001	000179697	ACOSTA SANCHEZ MANUEL	macost15@itam.mx
ALGORITMICA Y PROGRAMACION	001	000180894	CASTRO ALVAREZ MARIA FERNANDA	mcastr52@itam.mx
ALGORITMICA Y PROGRAMACION	001	000181084	MASETTO HERRERA MARTHA PATRICIA	mmasetto@itam.mx
ALGORITMICA Y PROGRAMACION	001	000181121	LEZAMA JACINTO CARLOS ENRIQUE	clezamaj@itam.mx
ALGORITMICA Y PROGRAMACION	001	000181586	RIVADENEYRA OCAMPO JUAN PABLO	jrivaden@itam.mx
ALGORITMICA Y PROGRAMACION	001	000181592	MAZARIEGOS SALDAÑA ROGELIO	rogelio.mazariegos@itam.mx
ALGORITMICA Y PROGRAMACION	001	000181898	TREVIÑO MONTEMAYOR MAURICIO JESUS	mtrevio2@itam.mx
ALGORITMICA Y PROGRAMACION	001	000182153	RODRIGUEZ MARTINEZ RUBEN	rrodr122@itam.mx
ALGORITMICA Y PROGRAMACION	001	000182163	ALVAREZ GUZMAN JAVIER EMILIANO	jalvar92@itam.mx
ALGORITMICA Y PROGRAMACION	001	000182561	PAREDES PEREZ JOSE MIGUEL	jpared13@itam.mx
ALGORITMICA Y PROGRAMACION	001	000183707	MEDINA HERNANDEZ ULISES JAZHAEL	umedinah@itam.mx
ALGORITMICA Y PROGRAMACION	001	000183829	CHAVEZ ALVARADO HUGO	hchaveza@itam.mx
ALGORITMICA Y PROGRAMACION	001	000183929	BALDERAS CONTRERAS ERICK SALVADOR	ebalder1@itam.mx
ALGORITMICA Y PROGRAMACION	001	000188580	SALINAS GIORDANO LEONARDO	lsalin10@itam.mx

Temas para esta sesión

- **Examen**
- **Repaso**

POO - 40%

En base al siguiente diagrama:

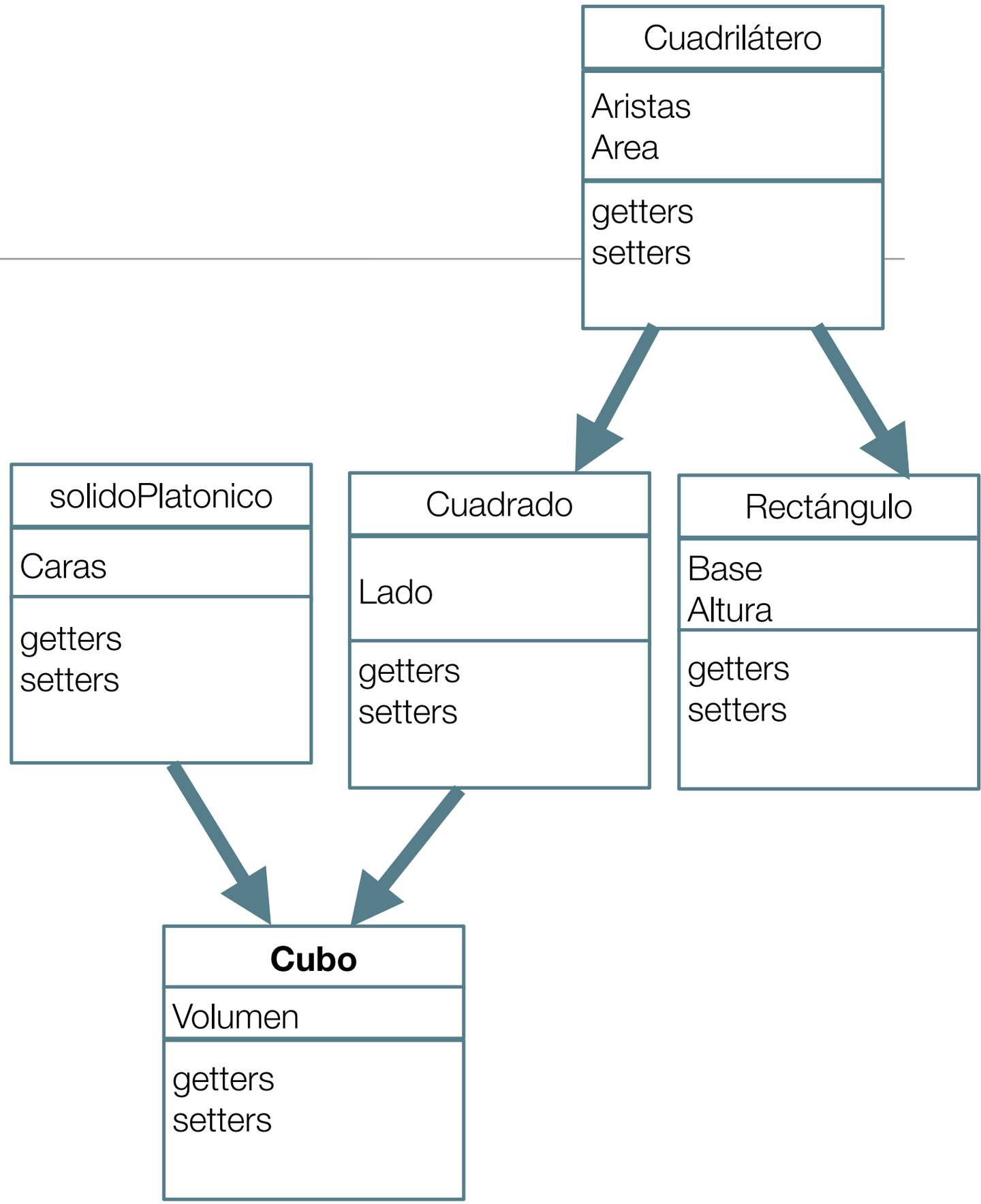
1. Crea las clases correspondientes - 10%

2. Define getters y setters para los atributos de cada clase - 10%

3. Define constructores para inicializar los atributos con valores aleatorios - 10%

4. Define funciones para calcular area y/o volumen según aplique para clase - 10%

5. Adjunta código



```python
class Cubo(solidoP,Cuadrado):
    __Volumen = 0

    def __init__(self,v1):
        self.__Lado = v1
        self.__Volumen = 0

    def getVolumen(self):
        return self.__Volumen

    def setVolumen(self,v1):
        self.__Volumen = v1

    def calculaVolumen(self):
        self.setVolumen(self.__Lado**3)
```

```
In [29]: x = Cubo(4)

In [30]: x.getVolumen()
Out[30]: 0

In [31]: x.__dict__
Out[31]: {'_Cubo__Lado': 4, '_Cubo__Volumen': 0}

In [32]: x.calculaVolumen()

In [33]: x.getVolumen()
Out[33]: 64

In [34]: x.__dict__
Out[34]: {'_Cubo__Lado': 4, '_Cubo__Volumen': 64}
```

```python
class Cubo(solidoP,Cuadrado):
    __Volumen = 0

    def __init__(self,v1):
        super().setLado(v1)
        self.__Volumen = 0

    def getVolumen(self):
        return self.__Volumen

    def setVolumen(self,v1):
        self.__Volumen = v1

    def calculaVolumen(self):
        self.setVolumen(super().getLado()**3)
```

```
In [94]: x = Cubo(4)

In [95]: x.__dict__
Out[95]: {'_Cuadrado__Lado': 4, '_Cubo__Volumen': 0}

In [96]: x.getLado()
Out[96]: 4

In [97]: x.getArea()
Out[97]: 0

In [98]: x.getVolumen()
Out[98]: 0

In [99]: x.calculaArea()

In [100]: x.calculaVolumen()

In [101]: x.getArea()
Out[101]: 16

In [102]: x.getVolumen()
     ...:
Out[102]: 64
```

POO y simulación - 40%

- Usando orientación a objetos con encapsulamiento:

1. Simula el siguiente sistema dinámico para los primeros 50 pasos y grafica los valores de ambas variables

2. El constructor debe permitir inicializar:

    A. las variables y

    B. los parámetros del sistema

3. Adjunta código

$$x_t = x_{t-1} + \alpha(x_{t-1} - \beta x_{t-1} y_{t-1})$$

$$y_t = y_{t-1} + \gamma(y_{t-1} - \epsilon y_{t-1} x_{t-1})$$

$$0 < x < y$$

ITAM

```python
import numpy as np
import matplotlib.pyplot as plt


class dinamico:
    __x = 0
    __y = 0
    __alfa = 0
    __beta = 0
    __gama = 0
    __epsilon = 0

    def __init__(self,v1,v2,v3,v4,v5,v6):
        self.__x = v1
        self.__y = v2
        self.__alfa = v3
        self.__beta = v4
        self.__gamma = v5
        self.__epsilon = v6

        if v1 > v2 or (v1 < 0) or (v2 < 0):
            print('De valores de "x" y "y" que cumplan que 0<x<y')

    def observe(self):
        return self.__x, self.__y

    def actualizar(self):
        nextX = self.__x + (self.__alfa * (self.__x - (self.__beta * self.__x * self.__y)))

        nextY = self.__y + (self.__gamma * (self.__y - (self.__epsilon * self.__y * self.__x)))
        self.__x, self.__y = nextX,nextY

#sistema = dinamico(9,10,5,4,2,8)
sistema = dinamico(0.1,0.2,0.2,2,0.1,1)
valo = []
for i in range(50):
    x,y = sistema.observe()
    valo.append([x,y])
    sistema.actualizar()

arrayV = np.asarray(valo)
plt.plot(arrayV[:,0],'--b')
plt.plot(arrayV[:,1],'--g')
plt.show()
```

ITAM

Linalg - 20%

- Una receta de galletas busca tener una relación de grasa, carbohidratos, proteína y agua al 20:50:10:20 respectivamente.
- Para tal propósito se piensa usar cuatro ingredientes: mantequilla, harina, leche y huevos.
- Una porción de mantequilla tiene una proporción de 85% grasa, 1% proteína y 14% agua. Una porción de harina está compuesta por 86% carbohidratos, 10% proteína y 4% agua. Una porción de leche contiene 4% grasa, 6% carbohidratos, 4% proteína y 86% agua. Finalmente, una porción de huevos tiene 10% grasa, 1% carbohidratos, 35% proteína y 54% agua.
- ¿Cuantas porciones de cada ingrediente se necesitan para la receta?
- Adjunta código

```python
import numpy as np
Ingredientes=np.array([[.85,0.,.04,.1],[0.,.86,.06,.01],[.01,.1,.04,.35],[.14,.04,.86,.54]])
Proporciones=np.array([.20,.50,.10,.20])
Sol=np.linalg.solve(Ingredientes,Proporciones)

print(Sol)
```

Repaso

- Estructuras de datos básicas
- Estructuras de control
- Pandas
    - Análisis de datos
- POO
- Simulación básica de sistemas

Estructuras de datos básicas

- Tuplas
- Listas
- Diccionarios

Estructuras de control

- If then else
- For
- While

Pandas

- Leer archivos CSV
- Análisis con:
  - Series
  - Dataframes
- Exportar a archivos CSV

POO

- Encapsulamiento
- Herencia
  - Simple
  - Multiple
- Polimorfismo

Simulación básica de sistemas

- Inicialización
- Observación
- Actualización
- Graficación