

Leonardo de los Rios
CUI: 44.038.085
LSI-21198

Parcial II P00

Práctica

~~Clase AparatoTec~~

ClaseAparatoTec.py
from abc import ABC
import abc

Class AparatoTec(ABC):

-- marca = "

-- modelo = "

-- color = "

-- paisFab = "

-- precioBase = 0.0

-- importeVenta = 0.0

def __init__(self, marca=" ", modelo=" ", color=" ", paisFab=" ", precioBase=0.0):

assert isinstance(marca, str)

assert isinstance(modelo, str)

assert isinstance(color, str)

assert isinstance(paisFab, str)

assert isinstance(precioBase, float)

assert precioBase > 0

self.__marca = marca

self.__modelo = modelo

self.__color = color

self.__paisFab = paisFab

self.__precioBase = precioBase

self.__importe^{venta} = 0.0

@abc.abstractmethod

def calculaImporte():

pass

def getPrecioBase(self):

return self.__precioBase

def setImporte(self, importe):

assert isinstance(importe, float) and importe > 0

self.__importeVenta = importe


```
def calculaImporte(self):  
    self.getPrecioBase()
```

Leonardo de los Rios

Pág ②

```
def getMarca(self):
```

```
    return self.__marca
```

```
def getImporte(self):
```

```
    return self.__importeVenta
```

```
def getPais(self):
```

```
    return self.__paisFab
```


Clase Televisor.py

from clase AparatoTec import AparatoTec

class Televisor(AparatoTec):

-- tipoPantalla = "

-- pulgadas = 0.0

-- tipoDef = "

-- conexInternet = False

def __init__(self, marca="", modelo="", color="", paisFab="", precioBase=0.0,
tipoPantalla="", pulgadas=0.0, tipoDef="", conexInternet=False):

assert isinstance(tipoPantalla, str)

assert isinstance(pulgadas, float)

assert pulgadas > 0

assert isinstance(tipoDef, str)

assert isinstance(conexInternet, bool)

super().__init__(marca, modelo, color, paisFab, precioBase)

self.__tipoPantalla = tipoPantalla

self.__pulgadas = pulgadas

self.__tipoDef = tipoDef

self.__conexInternet = conexInternet

def calculoImporte(self):

~~precioB~~

importe = self.getPrecioBase()

if self.__tipoDef == 'SD':

importe += ~~importe~~ importe * 0.01

elif self.__tipoDef == 'HD':

importe += importe * 0.02

elif self.__tipoDef == 'FULL HD':

importe += importe * 0.03


```

if self.__conexInternet == True:
    importe += self.getPrecioBase() * 0.1
return importe

```

Clase Heladera.py

```

from clase AparatoTec import AparatoTec

```

```

class Heladera(AparatoTec):

```

```

    __capacidad = 0.0

```

```

    __freezer = False

```

```

def __init__(self, marca="", modelo="", color="", paisFob="", precioBase=0.0,
             capacidad=0.0, freezer=False):

```

```

    assert isinstance(capacidad, float)

```

```

    assert capacidad > 0

```

```

    assert isinstance(freezer, bool)

```

```

    super().__init__(marca, modelo, color, paisFob, precioBase)

```

```

    self.__capacidad = capacidad

```

```

    self.__freezer = freezer

```



```
def calculaImporte(self):  
    importe = self.getPrecioBase()  
    if self._freezer == False:  
        importe += importe * 0,01  
    else:  
        importe += importe * 0,05  
    return importe
```


else:

Leonardo de los Rios

Leonardo de los Rios

Pag 6

ClaseLavarropas.py

from claseAparatoTec import AparatoTec

Class Lavarropa(AparatoTec):

--Capacidad = 0

--centrifugado = 0

--cantidadProg = 0

--tipoCarga = "

def __init__(self, marca="", modelo="", color="", paisFab="", precioBase=0, capacidad=0, centrifugado=0, cantidadProg=0, tipoCarga=""):

assert isinstance(capacidad, int) and capacidad > 0

assert isinstance(centrifugado, int) and centrifugado > 0

assert isinstance(cantidadProg, int) and cantidadProg > 0

assert isinstance(tipoCarga, str) and (tipoCarga == 'Superior' or tipoCarga == 'Inferior')

super().__init__(marca, modelo, color, paisFab, precioBase)

self.__capacidad = capacidad

self.__centrifugado = centrifugado

self.__cantidadProg = cantidadProg

self.__tipoCarga = tipoCarga

def calculaImporte(self):

importe = self.getPrecioBase()

if self.__capacidad <= 5:

importe += importe * 0,03

else:

importe += importe * 0,03

def getCarga(self):

return self.__tipoCarga

ClaseNodo.py

from claseAparatoTec import AparatoTec

Class Nodo:

--dato = None

--siguiente = None

def __init__(self, dato):

assert isinstance(dato, AparatoTec)

self.__dato = dato

self.__siguiente = None

def setSiguiente(self, siguiente):

~~basculador~~

self.__siguiente = siguiente

def getSiguiente(self):

return self.__siguiente

def getDato(self):

return self.__dato

~~def~~

else:

aux = min

Leonardo de los Rios

Pag 5

Clase Lista.py

from claseTelevisor import Televisor

from claseHeladera import Heladera

from claseLavarropa import Lavarropa

from claseNodo import Nodo

Class Lista:

-- comienzo = None

-- actual = None

-- indice = 0

-- tope = 0

def __init__(self):

self.__comienzo = None

self.__actual = None

self.__indice = 0

self.__tope = 0

def __iter__(self):

return self

def __next__(self):

if self.__indice == self.__tope:

self.__actual = self.__comienzo

self.__indice = 0

~~raise~~ raise StopIteration

else:

self.__indice += 1

dato = self.__actual.getdato()

self.__actual = self.__actual.getsiguiente()

return dato

def __len__(self):

return self.__tope

def insertar(self, elemento, posicion):

if posicion == 0:

self.agregar(elemento)

assert isinstance(elemento, AparatoTec)
(assert isinstance(posicion, int) and posicion >= 0)

else:

aux = self.--comienzo

i = 0

elemento = Nodo(elemento)

while ~~aux~~ i < posicion and aux != None:

anterior = aux

aux = aux.getSiguiente()

i += 1

if i == posicion:

raise IndexError

else:

elemento.setSiguiente(aux)

anterior.setSiguiente(elemento)

self.--tope += 1

def agregar(self, elemento):

assert isinstance(elemento, AparatoTec)

nodo = Nodo(elemento)

nodo.setSiguiente(self.--comienzo)

self.--comienzo = nodo

self.--actual = nodo

self.--tope += 1

def opc3(self):

~~140~~ i = 0

actual = self.--comienzo

cantidad = 0

while i < len(~~self~~ ^{self}) ~~and~~:

if actual.getDato().getMarca() == 'Philips':

cantidad += 1

actual = actual.getSiguiente()

i += 1

return cantidad


```
def opc4(self):
```

```
    Marcas = []
```

```
    i = 0
```

```
    actual = self._comienzo
```

```
    while i < len(self):
```

```
        if actual is instance(Carro):
```

```
            if actual.getDatos().getCarga() == 'superior':
```

```
                Marcas.append(actual.getDatos().getMarca())
```

```
                actual = actual.getSiguiente()
```

```
            i += 1
```

```
    for marca in Marcas:
```

```
        print(marca)
```

```
def opc5(self):
```

```
    for dato in self:
```

```
        print(dato.getMarca() + '\n' + dato.getPais() + '\n')
```

```
        dato.calculaImporte()
```

```
        print(dato.getImporte())
```

and actual.getDatos().getMarca() not in Marcas:

[Faint, mostly illegible handwritten text, possibly bleed-through from the reverse side of the page. The text appears to be organized into several paragraphs or sections.]


```
ClasMenuOpciones.py
from claseLista import Lista
from claseHeladera import Heladera
from claseTelevisor import Televisor
from claseLavadora import Lavadora
```

Class Menu:

-- opc=0

def __init__(self):

self.__opc=0

def menu():

lista=Lista()

continuar=True

while continuar:

Print(f" Menú opciones:

1) Insertar.

2) Agregar.

3) Mostrar marca Philips. 4) Lavadora carga superior. 5) Mostrar. 6) Salir

~~opc = int~~ (input
self.__opc = int('Ingrese una opción: '))

if self.__opc == 1:

aparatos = ['Televisor', 'Heladera', 'Lavadora'] ==> Va en solicitarDatos()

posicion = int(input('Ingrese pos: '))

self.solicitarDatos(lista, aparato)

aparato = input('Ingrese aparato: ')

if aparato == aparatos[0]:

~~cantidad~~ = int(input('Cantidad: '))

tipoPant = input('Tipo Pant: ')

pulgadas = float(input('Pulgadas: '))

tipoDef = input('Tipo def: ')

conexInternet = bool(input('Conex Internet: '))

unTelevisor = Televisor(marca, modelo, color, paisFab, precioBase,

tipoPant, pulgadas, tipoDef, conexInternet)

if self.__opc == 1:
lista.insertar(unTelevisor, posicion)

else:
lista.agregar(unTelevisor)

Si el usuario
elige la op
de insertar,
se solicita la
posición

Todo esto
va en
def solicitarDatos

elif aparato == aparatos[5]:

capacidad = float(input('Capacidad: '))

freezer = bool(input('Freezer: '))

unHeladera = Heladera(marca, modelo, color, paisFab, precioBase, capacidad, freezer)

if self._opc == 1:

lista.insertar(unHeladera, posicion)

else: lista.agregar(unHeladera)

elif aparato == aparatos[2]:

capacidad = int(input('Capacidad: '))

centrifugado = int(input('Centrifugado: '))

cantidadProg = int(input('Cantidad Prog: '))

tipoCarga = input('Tipo carga: ')

unLavadora = Lavadora(marca, modelo, color, paisFab, precioBase, capacidad, centrifugado, cantidadProg, tipoCarga)

if self._opc == 1:

lista.insertar(unLavadora, posicion)

else: lista.agregar(unLavadora)

elif self._opc == 2:

~~aparatos = [Heladera, Lavadora, Lavadora]~~

self.solicitaDatos(lista, aparato)

aparato = input('Ingrese aparato: ')

elif self._opc == 3:

cantidad = lista.opc3()

print(cantidad)

elif self._opc == 4:

lista.opc4()

elif self._opc == 5:

lista.opc5()

elif self._opc == 0:

continuar = False

else:

print('Opción no válida.')

def solicitaDatos()
se corrigió
en pag 55 y 56

def solicitaDatos (self, lista, aparato):

ya se escribió el contenido en las págs. 12 y 13

Se corrigió
en pág 16

main.py

```
from claseMenuOpciones import menuOpciones
```

```
if __name__ == '__main__':
```

```
    UnMenu = MenuOpciones()
```

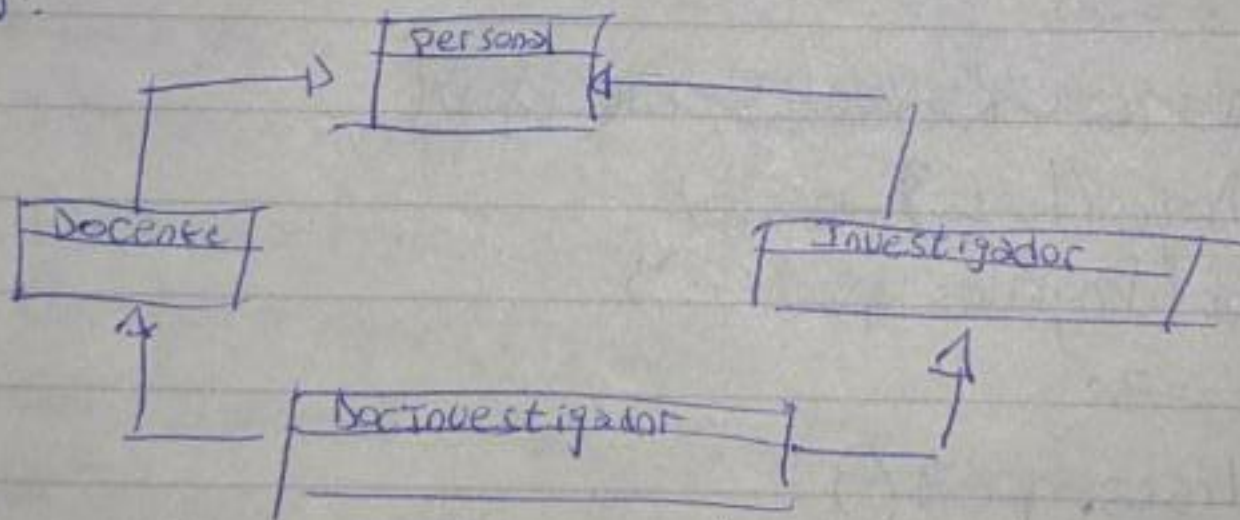
```
    UnMenu.Menu
```

Teoría.

1) La diferencia es que las clases que implementen los métodos en una interfaz no deben estar relacionadas por la herencia. En cambio, las clases hijas ~~que~~ que se derivan de una clase Padre, heredan los métodos de la clase Padre.

2) La regla del diamante funciona de derecha a izquierda.

Ejemplo:



personal: recibe sus atributos + los de docente + los de investigador.

Docente: recibe los de personal + los de investigador + sus atributos.

Investigador: recibe los de personal + los suyos + los de docente.

Docente Inves: recibe los de personal + los de investigador + los de docente.

En la clase Docente Inves. se puede apreciar su funcionamiento.

3) Los bloques ^{mencionados} se utilizan para no frenar la ejecución del programa y evitar de esa forma tener que volver a ejecutarlo.

Corrección en ClaseMenuOpciones.py :

```
if self.__opc == 1:
    aparatos = ['Televisor', 'Heladera', 'Lavadora']
    aparato = input('Aparato: ')
    if aparato in aparatos:
        self.solicitaDatos(lista, aparato)
```

```
elif self.__opc == 2:
    aparatos = ['Televisor', 'Heladera', 'Lavadora']
    aparato = input('Aparato: ')
    if aparato in aparatos:
        self.solicitaDatos(lista, aparato)
```

```
def solicitaDatos(self, lista, aparato):
```

```
    Marca = input('Marca: ')
    Modelo = input('Modelo: ')
    color = input('Color: ')
    PrecioBase = float(input('Precio Base: '))
    if aparato == 'Televisor':
```

```
        tipoPan = input('Tipo Pantalla: ')
        pulgadas = float(input('Pulgadas: '))
        tipoDef = input('Tipo def: ')
        conexInternet = bool(input('Conex Internet: '))
        unTelevisor = Televisor(Marca, Modelo, color, PrecioBase, tipoPan, pulgadas,
```

```
                                tipoDef, conexInternet)
```

```
    elif self.__opc == 1:
```

```
        posicion = int(input('Posición: '))
        lista.insertar(unTelevisor, posicion)
```

```
    elif self.__opc == 2:
        lista.agregar(unTelevisor)
```

```
    elif aparato == 'Heladera':
        capacidad = float(input('Capacidad: '))
```



```
freezer = bool(input('Freezer: '))
unaHeladera = Heladera(marca, modelo, color, paisFab, precioBase,
                        capacidad, freezer)
```

```
if self._opc == 1:
    posicion = int(input('Posicion: '))
    lista.insertar(unaHeladera, posicion)
```

```
elif self._opc == 2:
    lista.agregar(unaHeladera)
```

else:

```
capacidad = int(input('Capacidad: '))
centrifugado = int(input('Centrifugado: '))
cantidadProg = int(input('Cantidad Prog: '))
tipoCarga = input('Tipo carga: ')
unLavavropa = Lavavropa(marca, modelo, color, paisFab, precioBase,
                        capacidad, centrifugado, cantidadProg, tipoCarga)
```

```
if self._opc == 1:
    posicion = int(input('Posicion: '))
    lista.insertar(unLavavropa, posicion)
```

```
elif self self._opc == 2:
    lista.agregar(unLavavropa)
```