INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Programación avanzada

Práctica 3: Herencia y Polimorfismo

Adalberto Cabrera Vazquez

Boleta: 2023640791

Ingeniería Mecatrónica, Grupo 2MV7

13 de Marzo del 2024

1 Objetivo

Desarrollar programas utilizando la herencia y polimorfismo.

2 Introducción

La herencia es un proceso mediante el cual se puede crear una clase hija que hereda de una clase padre, compartiendo sus métodos y atributos. Además de ello, una clase hija puede sobreescribir los métodos o atributos, o incluso definir unos nuevos.

¿Y para que queremos la herencia? Dado que una clase hija hereda los atributos y métodos de la padre, nos puede ser muy útil cuando tengamos clases que se parecen entre sí pero tienen ciertas particularidades.

El polimorfismo, por otro lado, es una característica que nos permite utilizar objetos de diferentes clases de manera intercambiable. Esto significa que el mismo método o función puede ser utilizado en diferentes tipos de objetos, sin preocuparnos por conocer los detalles exactos de cada uno de ellos. En Python, el polimorfismo está estrechamente relacionado con la herencia y la superposición de métodos.

3 Desarrollo

3.1 Parte 1: Computadoras

Cree un aplicativo donde defina las siguientes clases:

1. Computadora. Clase abstracta. Deberá declarar los métodos *Getters* y *Setters* abstractos para los atributos: Memoria, Procesador, Almacenamiento, GPU.
2. ComputadoraPortatil. Hereda de Computadora. Deberá tener adicionalmente al menos 1 atributo y 1 método.
3. ComputadoraEscritorio. Hereda de Computadora. Deberá tener adicionalmente al menos 1 atributo y 1 método.
4. TelefonoInteligente. Hereda de Computadora. Deberá tener adicionalmente al menos 1 atributo y 1 método.
5. Tablet. Hereda de Computadora. Deberá tener adicionalmente al menos 1 atributo y 1 método.

En su programa principal pruebe sus clases diseñadas creando al menos 3 instancias de cada una.

Primero debemos crear la super clase Computadora, la clase de las que los demás dispositivos van a heredar atributos. Esta es una clase abstracta. Vamos a usar los getter y setters para poder o meter la informacion de los dispositivos y para poder igual imprimirlas ya que son atributos protegidos.

```python
class Computadora(metaclass=ABCMeta):
    @abstractmethod
    def __init__(self, memoria:str, procesador: str, almacenamiento:str, gpu:str) -> None:
        self.__memoria= memoria
        self.__procesador= procesador
        self.__almacenamiento= almacenamiento
        self.__gpu= gpu

        #Memoria
    @property
    def memoria(self):
        return self.__memoria

    @memoria.setter
    def memoria(self, nuevamemoria:str):
        self.__memoria= nuevamemoria
    # Procesador
    @property
    def procesador(self):
        return self.__procesador

    @procesador.setter
    def procesador(self, nuevoproce:str):
        self.__procesador= nuevoproce
    #Almacenamiento
    @property
    def almacenamiento(self):
        return self.__almacenamiento
```

```python
@almacenamiento.setter
def almacenamiento(self, nuevoalma:str):
    self.__almacenamiento= nuevoalma
#GPU
@property
def gpu(self):
    return self.__gpu

@gpu.setter
def gpu(self, nuevogpu:str):
    self.__gpu=nuevogpu
```

Figure 1: Creacion de la super-clase

Ahora vamos creando las sub-clases de computadora portatil, computadora de escritorio, telefono inteligente y tablet. En computadora portatil le agregamos un atributo de 'marca' y un metodo getter para poder imprimirla (figura 2). En la computadora de escritorio le agregamos el atributo 'tamaño' al igual que un metodo getter (figura 3). Para el telefono inteligente le agregamos el atributo 'numero' y el metodo getter (figura 4). Y en la tablet le agregué el atributo 'peso' al igual que un metodo getter (figura 5).

```python
class ComputadoraPortatil(Computadora):
    def __init__(self, memoria, procesador,almacenamiento, gpu, marca:str) -> None:
        super().__init__(memoria, procesador, almacenamiento, gpu)
        self.__marca = marca

    @property
    def marca(self):
        return self.__marca
```

Figure 2: Clase Portatil

```python
class ComputadoraEscritorio(Computadora):
    def __init__(self, memoria, procesador, almacenamiento, gpu, tamaño:str) ->None:
        super().__init__(memoria, procesador, almacenamiento, gpu)
        self.__tamaño = tamaño

    @property
    def tamaño(self):
        return self.__tamaño
```

Figure 3: Clase Escritorio

```python
class Tablet(Computadora):
    def __init__(self,memoria, procesador, almacenamiento, gpu, peso:str) ->None:
        super().__init__(memoria,procesador, almacenamiento, gpu)
        self.__peso= peso

    @property
    def peso(self):
        return self.__peso
```

Figure 4: Clase Telefono

```python
class Tablet(Computadora):
    def __init__(self,memoria, procesador, almacenamiento, gpu, peso:str) ->None:
        super().__init__(memoria,procesador, almacenamiento, gpu)
        self.__peso= peso

    @property
    def peso(self):
        return self.__peso
```

Figure 5: Clase Tablet

Ahora le tenemos que meter informacion a los objetos, asi que creamos 3 objetos por cada sub-clase.

```python
portatil_1=ComputadoraPortatil('16', 'Intel® Celeron® G','1T', 'Zotac(20)','Samsung')
portatil_2=ComputadoraPortatil('8', 'AMD Ryzen™ 9', '256 Gb', 'SAPPHIRE(11)', 'Apple')
portatil_3=ComputadoraPortatil('4', 'Intel® Core™ i5-10xxx', '512 Gb', 'ASRock(9)','LENOVO')

escritorio_1=ComputadoraEscritorio('4', 'Intel® Core™ i5-12xxx', '256 Gb','PowerColor(4)', '22000')
escritorio_2=ComputadoraEscritorio('8','AMD Ryzen™ 5', '512 Gb', 'ASUS(60)', '23000')
escritorio_3=ComputadoraEscritorio('8', 'Intel® Core™ i3-10xxx', '1T', 'MSI(26)','21000')

telefono_1=TelefonoInteligente('8', 'Qualcomm Snapdragon 8 Gen 2', '512 Gb', 'Adreno 750', '9984037473')
telefono_2=TelefonoInteligente('4','Apple A16 Bionic', '256 Gb', 'Apple M2 10-Core GPU', '9981007887')
telefono_3=TelefonoInteligente('8' ,'Samsung Exynos 2200', '64 Gb', 'Mali-G615 MP6','9981000105')

tablet_1=Tablet('32',' 1.5GHz Dual-Core Cortex-A9, 1.6GHz', '512 Gb','ARM Mali-T764','500')
tablet_2=Tablet('16','Octa-Core Cortex-A7, 1.6GHz','256 Gb','ARM Mali-450MP6', '460')
tablet_3=Tablet('8','Quad-Core 64Bit Cortex-A53, 2.0GHz','256 Gb','Mali400MP2', '520')
```

Figure 6: Ingreso datos

Despues para concatenar la informacion y mostrarla de forma ordenada lo que hice fue meterla a una tabla usando la biblioteca tabulate.

```
data_portatil=[
    ["Dispositivo",'Memoria [Gb]', 'Procesador', 'Almacenamiento','GPU', 'Marca'],
    ['Portatil 1', portatil_1.memoria, portatil_1.procesador, portatil_1.almacenamiento, portatil_1.gpu, portati
    ['Portatil 2', portatil_2.memoria, portatil_2.procesador, portatil_2.almacenamiento, portatil_2.gpu, portati
    ['Portatil 3', portatil_3.memoria, portatil_3.procesador, portatil_3.almacenamiento, portatil_3.gpu, portati
]

data_escritorio=[
    ["Dispositivo",'Memoria [Gb]', 'Procesador', 'Almacenamiento','GPU', 'Tamaño [cm3]'],
    ['Escritorio 1', escritorio_1.memoria, escritorio_1.procesador, escritorio_1.almacenamiento, escritorio_1.gp
    ['Escritorio 2', escritorio_2.memoria, escritorio_2.procesador, escritorio_2.almacenamiento, escritorio_2.gp
    ['Escritorio 3', escritorio_3.memoria, escritorio_3.procesador, escritorio_3.almacenamiento, escritorio_3.gp
]

data_telefono=[
    ["Dispositivo",'Memoria [Gb]', 'Procesador', 'Almacenamiento','GPU', 'Numero'],
    ['Telefono 1', telefono_1.memoria, telefono_1.procesador, telefono_1.almacenamiento, telefono_1.gpu, telefor
    ['Telefono 2', telefono_2.memoria, telefono_2.procesador, telefono_2.almacenamiento, telefono_2.gpu, telefor
    ['Telefono 3', telefono_3.memoria, telefono_3.procesador, telefono_3.almacenamiento, telefono_3.gpu, telefor
]

data_tablet=[
    ["Dispositivo",'Memoria [Gb]', 'Procesador', 'Almacenamiento','GPU', 'Peso [gr]'],
    ['Tablet 1', tablet_1.memoria, tablet_1.procesador, tablet_1.almacenamiento, tablet_1.gpu, tablet_1.peso],
    ['Tablet 2', tablet_2.memoria, tablet_2.procesador, tablet_2.almacenamiento, tablet_2.gpu, tablet_2.peso],
    ['Tablet 3', tablet_3.memoria, tablet_3.procesador, tablet_3.almacenamiento, tablet_3.gpu, tablet_3.peso]
]
```

Figure 7: Creacion de la tabla

Finalente solo imprimimos las tablas. en orden.

```
print(tabulate(data_portatil, headers="firstrow", tablefmt="grid"))
print('')
print(tabulate(data_escritorio, headers="firstrow", tablefmt="grid")
print('')
print(tabulate(data_telefono, headers="firstrow", tablefmt="grid"))
print('')
print(tabulate(data_tablet, headers="firstrow", tablefmt="grid"))
```

Figure 8: Impresion de la informacion.

3.2 Ejecucion Parte 1

Dispositivo	Memoria [Gb]	Procesador	Almacenamiento	GPU	Marca
Portatil 1	16	Intel® Celeron® G	1T	Zotac(20)	Samsung
Portatil 2	8	AMD Ryzen™ 9	256 Gb	SAPPHIRE(11)	Apple
Portatil 3	4	Intel® Core™ i5-10xxx	512 Gb	ASRock(9)	LENOVO

Dispositivo	Memoria [Gb]	Procesador	Almacenamiento	GPU	Tamaño [cm3]
Escritorio 1	4	Intel® Core™ i5-12xxx	256 Gb	PowerColor(4)	22000
Escritorio 2	8	AMD Ryzen™ 5	512 Gb	ASUS(60)	23000
Escritorio 3	8	Intel® Core™ i3-10xxx	1T	MSI(26)	21000

Dispositivo	Memoria [Gb]	Procesador	Almacenamiento	GPU	Numero
Telefono 1	8	Qualcomm Snapdragon 8 Gen 2	512 Gb	Adreno 750	9984037473
Telefono 2	4	Apple A16 Bionic	256 Gb	Apple M2 10-Core GPU	9981007887
Telefono 3	8	Samsung Exynos 2200	64 Gb	Mali-G615 MP6	9981000105

Dispositivo	Memoria [Gb]	Procesador	Almacenamiento	GPU	Peso [gr]
Tablet 1	32	1.5GHz Dual-Core Cortex-A9, 1.6GHz	512 Gb	ARM Mali-T764	500
Tablet 2	16	Octa-Core Cortex-A7, 1.6GHz	256 Gb	ARM Mali-450MP6	460
Tablet 3	8	Quad-Core 64Bit Cortex-A53, 2.0GHz	256 Gb	Mali400MP2	520

Figure 9: Ejecucion parte 1

3.3 Parte 2: Mascotas

Dado el siguiente diagrama de clases, escriba un programa que contenga todas las clases y herencia indicadas. En su programa principal, muestre un menú con el cual permita al usuario adquirir tantas mascotas como desee, y le permita visualizar en pantalla todas sus mascotas.

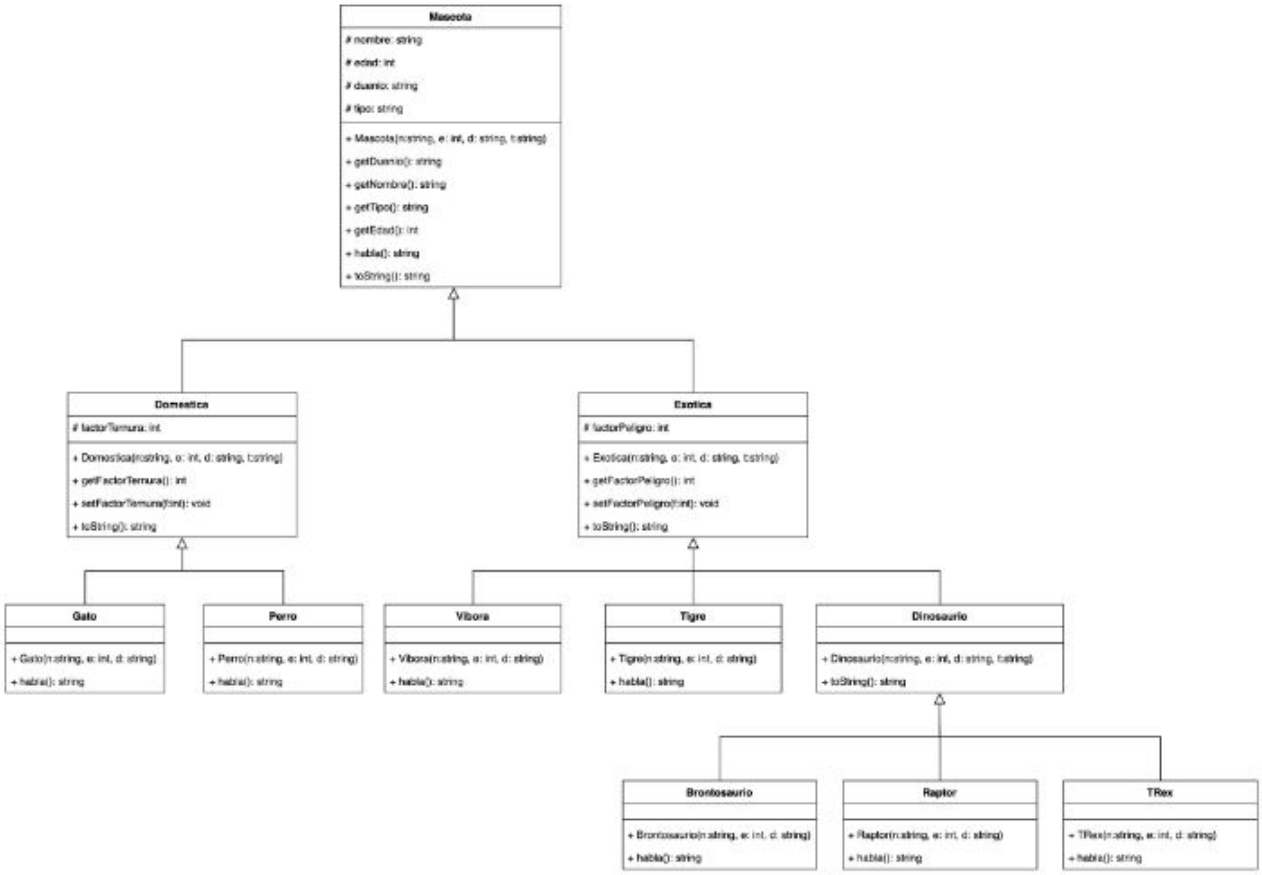

Figure 10: Diagrama de clases

Primero debemos de crear nuestra super-clase principal 'Mascotas' con sus debidos getters y setters y atributos. Esta tiene un metodo abstracto llamado 'habla()' que hace que todos los animales tengan que hablar.

```python
class Mascota:
    def __init__(self, nombre:str, edad: int, dueño:str, tipo:str) -> None:
        self.__nombre= nombre
        self.__edad= edad
        self.__dueño= dueño
        self.__tipo= tipo

    @property
    def dueño(self):
        return self.__dueño

    @property
    def nombre(self):
        return self.__nombre

    @property
    def edad(self):
        return self.__edad

    @property
    def tipo(self):
        return self.__tipo

    @abstractmethod
    def habla(self, msj):
        pass
```

Figure 11: Super-Clase Mascota

Ahora creamos la clase de Domestica que va a ser una super-clase para perro y gato, pero que a la vez es una sub-clase de 'Mascota'. En esta clase le agregamos el atributo factor de ternura. Ahora

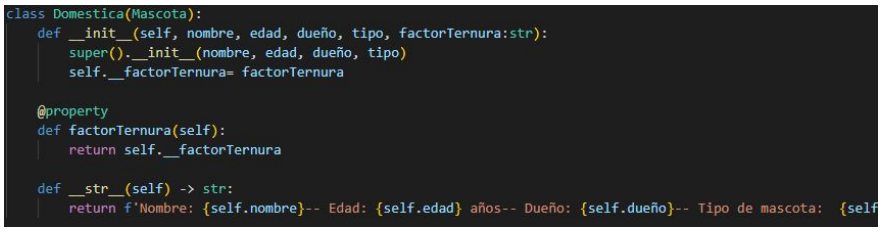

Figure 12: Clase Domestica

creamos las clases 'Gato' y 'Perro' que son sub-clases de Domestica en donde ahi le vamos a decir como hablan los gatos y los perros.

```python
class Gato(Domestica):
    def __init__(self, nombre, edad, dueño, tipo, factorTernura: str):
        super().__init__(nombre, edad, dueño, tipo, factorTernura)

    def habla(self, msj):
        return (f'\nLos gatos dicen {msj}')

class Perro(Domestica):
    def __init__(self, nombre, edad, dueño, tipo, factorTernura: str):
        super().__init__(nombre, edad, dueño, tipo, factorTernura)

    def habla(self, msj):
        return (f'\nLos perros dicen {msj}')
```

Figure 13: Clase Gato y Perro

Ahora creamos la clase de Exotica que va a ser una super-clase para Tigre, Vibora y Dinosaurio, pero que a la vez es ena sub-clase de Mascota. En esta clase agregamos el factor de Peligro. Ahora

```python
class Exotica(Mascota):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro:str) -> None:
        super().__init__(nombre, edad, dueño, tipo)
        self.__factorPeligro=factorPeligro

    @property
    def factorPeligro(self):
        return self.__factorPeligro

    def __str__(self):
        return f'Nombre: {self.nombre}-- Edad: {self.edad} años---Dueño: {self.dueño}---Tipo de mascota: {self.t
```

Figure 14: Clase Exotica

creamos las clase 'Vibora' y 'Tigre' que son sub-clases de la Domestica en donde ahi le vamos a decir como hablan las viboras y los tigres.

```python
class Vibora(Exotica):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)

    def habla(self, msj):
        return (f'\nLas viboras dicen {msj}')

class Tigre(Exotica):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)

    def habla(self, msj):
        return (f'\nLos tigres dicen {msj}')
```

Figure 15: Clase vibora y tigre

Despues creamos la clase de Dinosaurio que es una subclase de Exotica pero que es una super-clase de TRex, Raptor y Brontosaurio. Despues agregamos las clases 'TRex', 'Raptor' y 'Brontosaurio' y en esas clases especificamos como habla cada uno.

```python
class Dinosaurio(Exotica):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)
```

Figure 16: Clase dinosaurio

```python
class Brontosaurio(Dinosaurio):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)

    def habla(self, msj):
        return (f'\nLos brontosaurios dicen {msj}')

class Raptor(Dinosaurio):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)

    def habla(self, msj):
        return (f'\nLos raptors dicen {msj}')

class TRex(Dinosaurio):
    def __init__(self, nombre: str, edad: int, dueño: str, tipo: str, factorPeligro: str) -> None:
        super().__init__(nombre, edad, dueño, tipo, factorPeligro)

    def habla(self, msj):
        return (f'\nLos TRex dicen {msj}')
```

Figure 17: Clases 'TRex', 'Brontosaurio' y 'Raptor'

Ahora empezamos con el menu en donde tendremos la opcion de inscribir cada tipo de animal, ver los animales que llevamos y salir. Para eso, cada vez que agregemos a una mascota, el objeto creado lo agregaremos a una lista de mascotas.
Usamos un while para que se ponga la lista en un bucle hasta que decidamos salir del programa.

```
while flag==False:
    os.system('cls')
    print('BIENVENIDO A LA TIENDA DE MASCOTAS\n¿Que desea comprar el dia de hoy?:\n1. Domestica\n   1.1 Gato\n
    print('2. Exótica\n   2.1 Vivora\n   2.2 Tigre\n   2.3 Dinosaurio\n      2.3.1 Brontosaurio\n      2.3.2 Ra
    print('3. Mostrar a mis animales\n4. Salir')
    opcion=str(input('Ingrese la opcion que desee:'))
```

Figure 18: Menu

En cada opcion que pongas de ingresar animales te hará ingresar los datos de mismo, asi como te dirá como habla cada tipo de mascota. En la opcion 1.1 esocges el gato, la 1.2 escoges el perro, la 2.1 escoges la vibora, la 2.2 escoges el tigre, la 2.3.1 escoges brontosaurio, la 2.3.2 escoges el raptor, la 2.3.3 el TRex, la 3 escoges visualizar a las mascotas ingresadas usando un for para imprimir los elementos de la lista de animales, y la opcion 4 es salir del menu.

```
if opcion=='1.1':
    os.system('cls')
    print('Escogiste registrar un gato.\n')
    gato=Gato(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '), 'Dome
    lista_animales.append(gato)
    print(gato.habla('MIAU MIAU'))


    input('\nPresiona cualquier tecla para continuar')
elif opcion=='1.2':
    os.system('cls')
    print('Escogiste registrar un Perro.\n')
    perro= Perro(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '), 'D
    lista_animales.append(perro)
    print(perro.habla('GUAU GUAU'))

    input('\nPresiona cualquier tecla para continuar')
elif opcion=='2.1':
    os.system('cls')
    print('Escogiste registrar una Vivora.\n')
    vivora= Vibora(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '),
    lista_animales.append(vivora)
    print(vivora.habla('TSSSS'))
    input('\nPresiona cualquier tecla para continuar')
```

```
elif opcion=='2.2':
    os.system('cls')
    print('Escogiste registrar una Tigre.\n')
    tigre= Tigre(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '), '
    lista_animales.append(tigre)
    print(tigre.habla('RRRRRAR'))
    input('\nPresiona cualquier tecla para continuar')
elif opcion=='2.3.1':
    os.system('cls')
    print('Escogiste registrar un dinosaurio (Brontosaurio).\n')
    brontosaurio= Brontosaurio(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese s
    lista_animales.append(brontosaurio)
    print(brontosaurio.habla('LIROLELIROLE'))
    input('\nPresiona cualquier tecla para continuar')
elif opcion=='2.3.2':
    os.system('cls')
    print('Escogiste registrar un dinosaurio (Raptor).\n')
    raptor= Raptor(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '),
    lista_animales.append(raptor)
    print(raptor.habla('RIRIRIR'))
    input('\nPresiona cualquier tecla para continuar')
elif opcion=='2.3.3':
    os.system('cls')
    print('Escogiste registrar un dinosaurio (TRex).\n')
    trex= TRex(str(input('Ingrese el nombre: ')), input('Ingrese edad: '), input('Ingrese su dueño: '), 'Ex
    lista_animales.append(trex)
    print(trex.habla('RAAAAUR'))
    input('\nPresiona cualquier tecla para continuar')
```

```
elif opcion=='3':
    print('Estas son tus mascotas')
    numero_animales=len(lista_animales)

    for i in range(numero_animales):
        print(str(i+1)+'. ', end='')
        print(lista_animales[i])
        i+=1

    input('\nPresiona cualquier tecla para continuar')
elif opcion=='4':
    flag=True
```

Si metes una opcion que no este en la lista del menu te sale un mensjaje de que la opcion ingresada es invalida.

```
else:
    print('\nFavor de ingresar un numero valido del menu')
    input('\nPresiona cualquier tecla para continuar')
    os.system('cls')
```

Figure 19: Opcion invalida

3.4 Ejecucion Parte 2

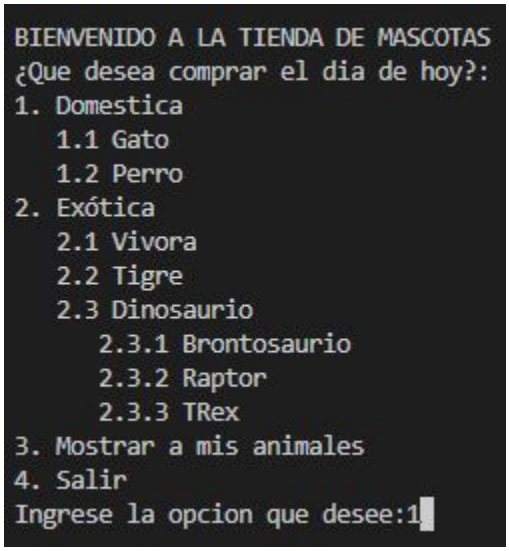

Figure 20: Imagen del menu

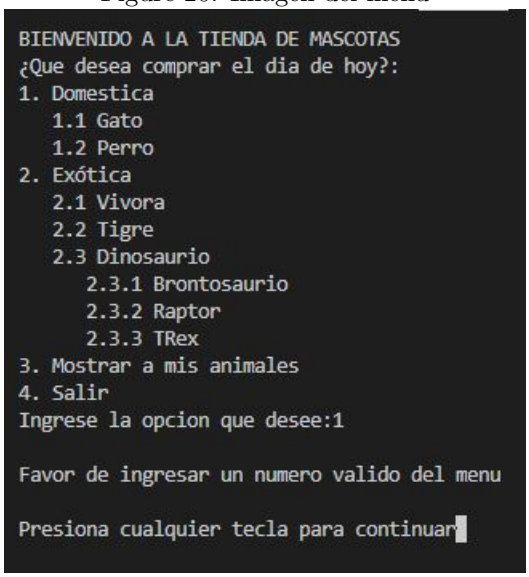

Figure 21: Menu invalido

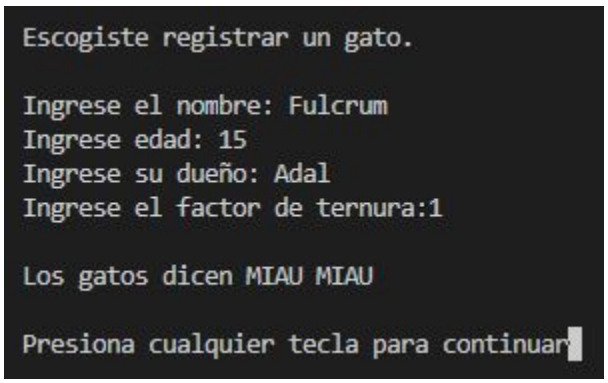

Figure 22: Registro de la mascota 1

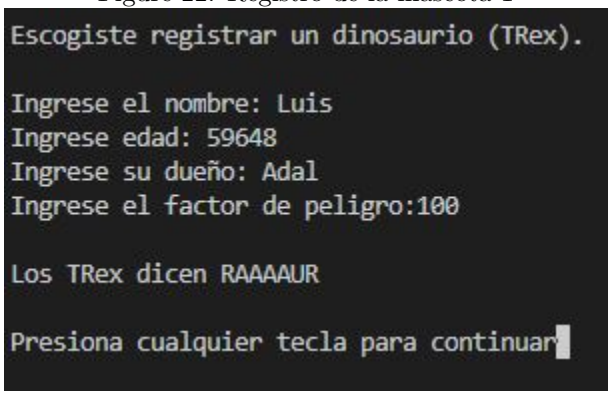

Figure 23: Registro mascota 2

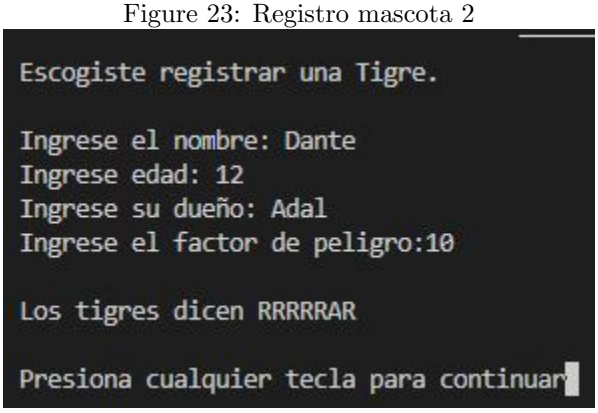

Figure 24: Registro mascota 3

Figure 25: Se visualizan las mascotas

4 Conclusión

En esta práctica desarrolle programas utlizando herencia asi como reforzando conocimientos de getters, setter, polimorfismo, etc. Hice nuevamente uso de bucles como for y otros metodos para realizar listas.

5 Bibliografia

- Herencia en Python. (n.d.). El Libro De Python. https://ellibrodepython.com/herencia-en-python
- Losapuntesde. (2024, February 4). Herencia y polimorfismo en Python: Amplía la flexibilidad de tus clases. Apuntes De Programador. https://apuntes.de/python/herencia-y-polimorfismo-en-python-ampli#:~:text=El%20polimorfismo%2C%20por%20otro%20lado,de%20cada%20uno%20de%20ellos.