MANUAL TÉCNICO

Gestor de Películas

26 DE FEB. DE 24

ANGEL ENRIQUE ALVARADO RUIZ - 202209714

Práctica Única de Lenguajes Formales y de Programación

Contenido

Paradigma utilizado 2
Clases Principales 2
La Clase App 2
Clase Main 9

Paradigma utilizado

El paradigma utilizado durante la practica única fue el paradigma orientado a objetos POO.

Clases Principales

```python
class Pelicula:
    def __init__(self, titulo, estreno, genero):
        self.titulo = titulo
        self.artistas = []
        self.estreno = estreno
        self.genero = genero

    def verArtistas(self) -> list:
        listadoAux = []
        for artista in self.artistas:
            listadoAux.append(artista.nombre)
        return listadoAux

class Artista:
    def __init__(self, nombre):
        self.nombre = nombre
        self.peliculas = []

    def verPeliculas(self) -> list:
        listadoAux = []
        for pelicula in self.peliculas:
            listadoAux.append(pelicula.titulo)
        return listadoAux
```

Por cada instancia de la clase Película, se asociará un título, un listado de artistas, un año de estreno y una lista de géneros. También tendrán un método que nos devolverá un listado con los nombres de los artístas que participan en la película.

Mientras que para cada instancia de la clase Artista, se asocia un nombre y un listado de películas en las que participa. Además cuenta con un método que devolverá el titulo de las películas en las que participa el artísta.

La razón principal por la que se decidió utilizar este tipo de de paradigma fue porque simplifica y optimiza la gestión de la información relacionada entre sí, ya que al poder almacenar referencias en los listados de cada clase, al actualizarse el estado de una instancia como tal, todas sus referencias manifestarán el cambio.

La Clase App

La clase app es la más importante, pues es la que hace posible gestionar la información de las películas ingresadas por medio de un archivo.lfp y de hecho, es la que se encarga de que podamos realizar, la carga, la gestión de información, el filtrado de información y el grafico de relaciones entre películas y artístas.

```python
# constructor de la clase
def __init__(self):
    self.listadoPeliculas = []
    self.listadoArtistas = []
    self.listadoGeneros = []
    self.listadoEstrenos = []

# apartado de carga de archivos
def configurarListados(self):
    for pelicula in self.listadoPeliculas:
        if pelicula.estreno not in self.listadoEstrenos:
            self.listadoEstrenos.append(pelicula.estreno)
    for pelicula in self.listadoPeliculas:
        aux = pelicula.genero
        for genero in aux:
            aux2 = genero.lower().strip()
            if aux2 not in self.listadoGeneros:
                self.listadoGeneros.append(aux2)
    print("Configuración de base de datos realizada con éxito")
```

Al final del documento se podrán ver capuras del código completo ya que es muy extenso, pero a priori, las funciones principales son el constructor y la configuración de listados, pues una se encarga de crear listas que almacenarán los datos (de películas, artístas, años de estreno y géneros de películas) y la otra se encarga de verificar la información que ya me ingresó y

se almacenó únicamente en el listado de películas; para que posteriormente se llenen los listados de generos y estrenos con los años de estreno asociados a las películas y sus categorías de géneros.

```
def carga(self,archivoRuta):
        try:
            # en caso de generarse un error el with open se encarga de cerrar el archivo
            with open(archivoRuta, "r") as archivo:
                lineas = archivo.readlines()
                for linea in lineas:
                    partes = linea.strip().split(";")
                    if len(partes) == 4:
                        titulo = partes[0].strip()
                        existePelicula = self.busqueda(titulo, "ninguno")
                        if existePelicula is False:
                            actores = partes[1].strip().split(",")
                            actores2 = [x.strip() for x in actores]
                            estreno = partes[2].strip()
                            genero = partes[3].strip().split(",")
                            pelicula = Pelicula(titulo, estreno, genero)
                            for actor in actores2:
                                existeArtista = self.busqueda("ninguno", actor)
                                if existeArtista is not True:
                                    artista = Artista(actor)
                                    artista.peliculas.append(pelicula)
                                    pelicula.artistas.append(artista)
                                    self.listadoArtistas.append(artista)
                                else:
                                    artista = self.retorno("ninguno", actor)
                                    artista.peliculas.append(pelicula)
                                    pelicula.artistas.append(artista)
                            self.listadoPeliculas.append(pelicula)
                        else:
                            print(f"La película:")
                            print(pelicula.titulo, pelicula.estreno, pelicula.genero)
                            print("con los artístas: ", pelicula.verArtistas())
                            while True:
                                respuesta = input("desea sobreescribir los datos de la pelicula? (s/n)")
                                if respuesta.lower() == "s" or respuesta.lower() == "n":
                                    break
                                print("Ingrese únicamente s o n")
                            if respuesta.lower() == "s":
                                self.listadoPeliculas.remove(self.retorno(titulo, "ninguno"))
                                actores = partes[1].strip().split(",")
                                estreno = partes[2]
                                genero = partes[3]
                                pelicula = Pelicula(titulo, estreno, genero)
                                for actor in actores:
                                    existeArtista = self.busqueda("ninguno", actor)
                                    if existeArtista is not True:
                                        artista = Artista(actor)
                                        artista.peliculas.append(pelicula)
                                        pelicula.artistas.append(artista)
                                        self.listadoArtistas.append(artista)
                                    else:
                                        artista = self.retorno("ninguno", actor)
                                        artista.peliculas.append(pelicula)
                                        pelicula.artistas.append(artista)
                                self.listadoPeliculas.append(pelicula)
                            else:
                                continue
            print("Carga realizada con éxito")
        except Exception as e:
            print(f"Error: {e}")
            print()
```

La carga de archivos es la parte más importante pues es donde estamos recibiendo toda la información y es donde, si no leemos bien y separamos correcamtente los datos, pueden existir los errores.

Al abrir el documento en la vista de lectura nos asegurarémos de que no vamos a modificar nada del archivo, además utilizamos “with open” porque de esta manera aunque exista un error o no, el archivo se cerrará automáticamente luego de finalizar. Utilizando el método readLines() podremos obtener cada linea del archivo, pero solamente accederemos individualmente a cada una sí realizamos el ciclo “Linea in Lineas” esto nos permitirá profundizar más y crear un listado con cada cadena de texto obtenida antes y después del separador “;”, para realizar este listado nos ayudarémos del método split(“;“), el strip() es únicamente para ignorar los espacio es en blanco al principio y al final de cada parte de la linea.

```
def busqueda(self, titulo="ninguno", nombre="ninguno") -> bool:
    if titulo != "ninguno":
        for pelicula in self.listadoPeliculas:
            if pelicula.titulo == titulo:
            # si existe entonces me devuelve un true
                return True
        return False
    else:
        for artista in self.listadoArtistas:
            artistaAux = artista.nombre
            if artistaAux == nombre:
                return True
        return False
```

Luego vamos a verficar que tengamos exactamente 4 partes por linea (acorde al formato de los datos de ingreso), luego nos ayudamos de una función llamada “busqueda” para verificar si la película ya se encuentra dentro del sistema.

Si no existe entonces el listado de actores almacenado en la posición 1 de nuestra lista, vamos a separarlos y a ignorar los espacios al principio y al final de cada cadena de caracteres, por ello volvemos a ayudarnos de los métodos strip() y el método split(“,”) ahora con una coma pues los artístas están separados por comas.

Luego realizamos un ajuste a los actores obtenidos, pues el strip() anterior solamente toma a todo el listado como un elemento, es decir, no elimina los espacios que hay al inicio y al final de cada elemento, solamente del primero y del último.

Por ello vamos a crear “actores2”, pues aquí ya vamos a decirle que a cada elemento x de actores, le vamos a aplicar el método strip() y esto resuelve el problema de tener espacios al incio y al final de elementos intermedios de la lista de actores.

Luegos se crea el objeto películas (sin actor), para verificar si existe el artísta y si este no existe lo va a crear, va a agregar a la película a su listado de películas, agregarémos al artísta al listado de artístas de la película y al listado creado a través del constructor de clase vamos a agregarle al artista nuevo.

A pesar de que parece faltar una vinculación luego de crear el artísta , no es necesario ya que lo que agregamos al listado de películas es la referencia del artísta, de la misma forma con la película, cada cambia que se haga desde cualquier otra parte del código, se verá reflejada en todas las referencias.

```python
def retorno(self, titulo="ninguno", nombre="ninguno") -> object:
    if titulo != "ninguno":
        for pelicula in self.listadoPeliculas:
            if pelicula.titulo == titulo:
            # si existe entonces me devuelve un true
                return pelicula
        return None
    else:
        for artista in self.listadoArtistas:
            if artista.nombre == nombre:
                return artista
        return None
```

En caso contrario de que el artísta ya existe entonces vamos a buscar el artísta con la función “retorno” y vamos a vincular la película al listado de películas del artísta y el artísta al listado de artístas de la película. Finalmente vamos a agregar la película al listado creado en el constructor de clase.

La gestión de información solamente consta de un mini menú de dos opciones que brindarán la posibilidad de mostrar el listado de películas por su nombre y luego podrá ver la información relacionada a cada película ingresando un número asociado al titulo, en el menú interactivo.

```python
def gestion(self):
    while True:
        print("-"*15, " Gestión de películas ", "-"*15)
        print(" 1. Mostrar listado de películas por nombre ")
        print(" 2. Mostrar listado de películas por artístas")
        print(" 3. Salir ")
        print("-"*67)
        print()
        option = ""
        while True:
            try:
                option = int(input("Ingrese el número de la opción a la que desea ingresar: "))
                if option >= 1 and option <= 3:
                    break
            except:
                print()
                print("Ingrese el número correspondiente a la opción que desea seleccionar")
        if option == 1:
            self.gestionInformacion("nombre")
            print()
        elif option == 2:
            self.gestionInformacion("artistas")
            print()
        elif option == 3:
            break
        else:
            print("Ingrese el número correspondiente a la opción que desea seleccionar")
            print()
```

Fue creado para mostrar el nombre de los artistas o de las películas, por ello al realizar interaciones del ciclo for, se imprime en consola un i+1 asociado a un título o a un nombre, este par se almacena en un diccionario que a su vez se almacena en una lista de diccionarios, para tener un registro local y no global del número que se puede ingresar y si es valido o no, luego tenemos la verificación de que se ingresen números desde 0 hasta el

```python
def gestionInformacion(self,clave) -> None:
        listadoAuxDiccionarios = []
        while True:
            print("-"*30, f"mostrar películas según {clave}", "-"*30)
            for i in range(len(self.listadoPeliculas)):
                if clave == "nombre":
                    print(f" {i+1}. {self.listadoPeliculas[i].titulo}")
                    diccionarioAux = {"pelicula": self.listadoPeliculas[i], "indice": i+1}
                    listadoAuxDiccionarios.append(diccionarioAux)
                else:
                    print(f" {i+1}. {self.listadoArtistas[i].nombre}")
                    diccionarioAux = {"artista": self.listadoArtistas[i], "indice": i+1}
                    listadoAuxDiccionarios.append(diccionarioAux)
            print(" 0. Salir ")
            print("-"*30)
            print()
            option = ""
            while True:
                try:
                    if clave == "nombre":
                        option = int(input("Ingrese el número de la película para ver los artístas que participan: "))
                        print()
                        if option <= (len(self.listadoPeliculas)+1) and option >= 0:
                            break
                    else:
                        option = int(input("Ingrese el número del artísta, para ver las películas en las que participa: "))
                        print()
                        if option <= (len(self.listadoArtistas)+1) and option >= 0:
                            break
                except:
                    print()
                    print("Ingrese el número correspondiente a la opción que desea seleccionar")
            if option == 0:
                break
            for elemento in range(len(listadoAuxDiccionarios)):
                if clave == "nombre":
                    peliculaAux = listadoAuxDiccionarios[elemento]["pelicula"]
                    indicePelicula = listadoAuxDiccionarios[elemento]["indice"]
                    if option == indicePelicula:
                        print("la película: ", peliculaAux.titulo, "cuenta con los artístas: ", peliculaAux.verArtistas())
                        break
                else:
                    artistaAux = listadoAuxDiccionarios[elemento]["artista"]
                    indiceArtista = listadoAuxDiccionarios[elemento]["indice"]
                    if option == indiceArtista:
                        print("el artísta: ", artistaAux.nombre, "participa en las películas: ", artistaAux.verPeliculas())
                        break
            break
```

número de iteraciones que se realizaron o en otras palabras la cantidad de elementos existentes en el listado de diccionarios auxiliares.

Luego para que nos muestre la información volvemos a iterar con un for, pero ahora vamos a buscar el número ingresado en la variable opción con el almacenado en el valor de la clave "indice", de esta forma podremos obtener el indice asociado al nombre del artísta o título de la película y retornarlo como una impresión en consola.

En la opción de filtrado solamente tenemos un menú que nos mostrará que tipo de filtrado queremos realizar.

```
def filtrado(self):
    while True:
        print("-"*15," Filtrado de películas ","-"*15)
        print("1. filtrar por artista")
        print("2. filtrar por estreno")
        print("3. filtrar por genero")
        print("4. Salir")
        print()
        option = None
        while True:
            try:
                option = int(input("ingrese el numero de la opcion que desea: "))
                print()
                if option >=1 and option <=4:
                    break
            except:
                print("error al ingresar la opcion intentelo nuevamente.")
                print("-"*30)
                print()
        if option == 1:
            self.gestionFiltros("artista")
            print()
        elif option == 2:
            self.gestionFiltros("estreno")
            print()
        elif option == 3:
            self.gestionFiltros("genero")
            print()
        elif option == 4:
            break
        else:
            print("Ingrese el número correspondiente a la opción que desea seleccionar")
            print()
```

Es decir, si deseamos filtrar por artísta, por genero o por año de estreno, es aquí donde comenzamos a ver la importancia de tener listas creadas a partir del constructor del clase y haberlas configurado previamente con el método de configurarlistados, ya que esto facilita el filtrado y lo optimiza en cuanto a tiempo, evitando realizar más instrucciones de las necesarias.

La opción de filtrado tiene que ver con estos 3 tipos, por lo que el código tiene tres condicionales al momento de imprimir y almacenar los datos, así como en el apartado de gestión, de hecho, funcionan de la misma forma las parte iniciales en las que creamos un diccionario por cada nombre, año de estreno, género y su respectivo i+1 con el cual podemos simular una selección después.

Para la selección como tal, realizamos un for que irá iterando entre los diferentes elementos del listado de diccionarios y estará obteniendo el valor del diccionario en la clave índice y en la clave artista, estreno o género. Sin embargo, la diferencia es entre artista y los otros dos filtros, pues artista ya cuenta con un listado propio y dinámico de las películas en las cuales participa, por eso solamente llamamos al método verPeliculas() que me devolverá un listado de nombres de las películas en las que aparece el actor.

```
def filtro(self, llave, tipo) -> list:
    listaAux = []
    if tipo == "estreno":
        # for que busca las peliculas que coincidan con el año de estreno
        for pelicula in self.listadoPeliculas:
            if pelicula.estreno == llave:
                listaAux.append(pelicula)
        return listaAux
    else:
        # for que busca las peliculas que coincidan con el género
        for pelicula in self.listadoPeliculas:
            aux = pelicula.genero
            for genero in aux:
                aux2 = genero.lower().strip()
                if aux2 == llave and aux2 not in listaAux:
                    listaAux.append(pelicula)
        return listaAux
```

En estreno y en género, necesitamos una función llamada "filtro" que acorde al tipo ingresado, se buscará esa llave (ese dato puntual) en los datos que tenemos en los listados previamente configurados y me devolverá un listado de películas acordes al filtro que se requiere. Al devolver el listado entonces se imprimirá en consola el listado como tal y así finalizaría el filtro.

El Código completo del filtro es el siguiente:

```
def gestionFiltros(self, tipo) -> None:
        while True:
            listadoAuxDiccionarios = []
            print("-"*15, f" Filtrado por {tipo} ", "-"*15)
            if tipo == "artista":
                for i in range(len(self.listadoArtistas)):
                    artistaAux = self.listadoArtistas[i]
                    print(f" {i+1}. {artistaAux.nombre}")
                    diccionarioAux = {"artista": artistaAux, "indice": i+1}
                    listadoAuxDiccionarios.append(diccionarioAux)
            elif tipo == "estreno":
                for i in range(len(self.listadoEstrenos)):
                    estreno = self.listadoEstrenos[i]
                    print(f" {i+1}. peliculas estrenadas en {estreno}")
                    diccionarioAux = {"estreno": estreno, "indice": i+1}
                    listadoAuxDiccionarios.append(diccionarioAux)
            else:
                for i in range(len(self.listadoGeneros)):
                    genero = self.listadoGeneros[i]
                    print(f" {i+1}. peliculas del genero {genero}")
                    diccionarioAux = {"genero": genero, "indice": i+1}
                    listadoAuxDiccionarios.append(diccionarioAux)
            print(" 0. Salir ")
            print("-"*30)
            print()
            option = None
            while True:
                try:
                    option = int(input("Ingrese el número del artísta por el que desea filtrar: "))
                    print()
                    if option <= (len(listadoAuxDiccionarios)+1) and option >= 0:
                        break
                except:
                    print()
                    print("Ingrese el número correspondiente a la opción que desea seleccionar")
            if option == 0:
                    break
            peliculasAux = []
            filtro = None
            if tipo == "artista":
                for i in range(len(listadoAuxDiccionarios)+1):
                    indiceAux = listadoAuxDiccionarios[i]["indice"]
                    artistaAux = listadoAuxDiccionarios[i]["artista"]
                    if option == indiceAux:
                        peliculasAux = artistaAux.verPeliculas()
                        filtro = artistaAux.nombre
                        break
                print(" * "*5,f"peliculas en las que participa {filtro}"," * "*5)
                for x in peliculasAux:
                    print(x)
                print(" * "*25)
                break
            elif tipo == "estreno":
                for i in range(len(listadoAuxDiccionarios)+1):
                    indiceAux = listadoAuxDiccionarios[i]["indice"]
                    estrenoAux = listadoAuxDiccionarios[i]["estreno"]
                    if option == indiceAux:
                        peliculasAux = self.filtro(estrenoAux, "estreno")
                        filtro = estrenoAux
                        break
                print(" * "*5,f"peliculas estrenadas en el año {filtro}"," * "*5)
                for x in peliculasAux:
                    print(x.titulo)
                print(" * "*23)
                break
            else:
                for i in range(len(listadoAuxDiccionarios)+1):
                    indiceAux = listadoAuxDiccionarios[i]["indice"]
                    generoAux = listadoAuxDiccionarios[i]["genero"]
                    if option == indiceAux:
                        peliculasAux = self.filtro(generoAux, "genero")
                        filtro = generoAux
                        break
                print(" * "*5,f"peliculas del genero {filtro}", " * "*5)
                for x in peliculasAux:
                    print(x.titulo)
                print(" * "*23)
                break
```

En cuanto al apartado para la realización de un diagrama visual por medio de graphviz, no es necesario realizar ningún tipo de procedimiento más que presionar la opción correspondiente

al de Grafico, ya que el código toma el listado de películas y trabaja únicamente sobre dicho listado para evitar errores.

```
def crear_grafo(self):
        # creando el objeto grafo dirigido
        grafo = Digraph()

        # diccionario de nodos
        nodos = {}

        # Agregar nodos de peliculas
        for pelicula in self.listadoPeliculas:
            #grafo.node(identificador de nodo,etiquetas , atributos)
            grafo.node(pelicula.titulo, f"Titulo: {pelicula.titulo}\nEstreno: {pelicula.estreno}\nGenero: {pelicula.genero}", shape="box", style="filled",
color="lightgreen")
            # verificando la lista de artistas
            for artista in pelicula.artistas:
                if artista.nombre not in nodos:
                    # si no existe el artista en el diccionario, se crea
                    nodos[artista.nombre] = []
                nodos[artista.nombre].append(pelicula.titulo)

        #conectando los nodos de artistas con las peliculas
        for artista, peliculas in nodos.items():
            grafo.node(artista, shape="box", style="filled", color="skyblue")
            for pelicula in peliculas:
                # relacionando los nodos
                # grafo.edge(nodoPrincipal, nodoSecundario, atributos)
                # el formato anterior es para que la flecha vaya de la pelicula al artista/s
                grafo.edge(pelicula, artista, color="black")

        # configuracion del grafo
        grafo.attr(rankdir="LR", splines="ortho")
        #guardando el grafo como jpg y abriendolo en el visor de imagenes
        grafo.render("digrama de relaciones Pelicula-Artista", format="jpg", view=True)
        # confirmando que el grafo fue exitoso
        print(" * "*10)
        print("Grafico creado correctamente")
        print(" * "*10)
```

Primeramente, hay que tomar en cuenta que hay que importar la librería de Graphviz al principio del archivo con la línea “from graphviz import *” de esta forma todo estará listo para comenzar. Primero mandamos a llamar al método de grafo dirigido Digraph(), luego creamos un diccionario para almacenar los nodos, luego vamos a iterar en el listado base de películas para crear nodos con el título como identificador, el texto que queremos que salga y los atributos que son , nodo tipo caja lleno y color verde claro.

Posteriormente vamos a iterar dentro del for para obtener los nombres de los artistas y si no existe el artista (creado como un valor dentro del diccionario) entonces inicializamos un artista PERO COMO UN VALOR EN EL DICCIONARIO, para luego agregar el título de la película al nodo de artista.

Esto no servirá ya que luego vamos a enlazar los nodos de las películas con los artistas utilizando otro for pero este obtendrá el artísta y las películas asociadas en el diccionario, dentro de este for vamos a crear un nodo ártísta, con el valor del nombre como identificador y etiqueta y con atributos similares a los que utilizamos al crear los nodos de películas pero cambiamos el color a un celeste. Por último vamos a crear aristas entre la película y el artísta con un for que recorra todas las películas obtenidas del for anterior.

Luego de eso ya estaría listo y solamente especificamos que la dirección del grafo es de izquierda a derecha y de carácter ortogonal (solo que se formen los nodos como una cuadrícula), agregamos un título a la imagen obtenida del método render, también definimos el formato y si se abre al crearse automáticamente.

Clase Main

```python
if __name__ == '__main__':
    app = App()
    app.bienvenida()
    input("Presione Enter para continuar...")
    print()
    print("-"*30,"carga de archivos de prueba","-"*30)
    app.carga("pruebas.lfp")
    app.carga("pruebas2.lfp")
    print("-"*30, "Listado de películas", "-"*30)
    app.configurarListados()
    print("-"*80)
    print()
    app.menu()
```

Más que una clase, es una configuración inicial; que verifica que la ejecución del archivo actual no sea un módulo (es decir que no se esté importando) y sea una ejecución directa (que efectivamente el archivo practicaUnica.py se ejecute).