Andy Roberto Jimenez Macnab

Manual Técnico:

GESTION DE DATOS DE PELICULAS

Laboratorio de Lenguajes Formales y de Programacion

Sección B+

La clase principal del programa es la clase Movie la cual hace lo siguiente:

__init__(self, listMovies): Este es el método de inicialización de la clase que toma una lista de películas como entrada y la guarda en la variable de instancia "listMovies".

filterByActor(self, actorName): Este método filtra las películas por actor y muestra todas las películas en las que aparece el actor proporcionado. El método toma el nombre del actor como entrada y utiliza un bucle for para iterar sobre cada elemento en la lista de películas. Si el nombre del actor está presente en la lista de actores para una película dada, se muestra el título de la película. Si no se encuentra ninguna película que tenga al actor proporcionado, el método devuelve False. De lo contrario, devuelve True.

filterByYear(self, yearInput): Este método filtra las películas por año y muestra todas las películas lanzadas en el año proporcionado. El método toma el año como entrada y utiliza un bucle for para iterar sobre cada elemento en la lista de películas. Si el año proporcionado coincide con el año de lanzamiento de una película, se muestra el título de la película y su categoría. Si no se encuentra ninguna película lanzada en el año proporcionado, el método devuelve False. De lo contrario, devuelve True.

filterByGenre(self, genreName): Este método filtra las películas por género y muestra todas las películas que pertenecen al género proporcionado. El método toma el nombre del género como entrada y utiliza un bucle for para iterar sobre cada elemento en la lista de películas. Si el género proporcionado coincide con el género de una película, se muestra el título de la película. Si no se encuentra ninguna película que pertenezca al género proporcionado, el método devuelve False. De lo contrario, devuelve True.

getData(self): Este método muestra todos los datos de todas las películas en la lista de películas. Utiliza un bucle for para iterar sobre cada elemento en la lista y muestra el número de la película, su título, sus actores, su año de lanzamiento y su categoría.

showActors(self): Este método muestra el título de todas las películas en la lista de películas y le solicita al usuario que seleccione una película. Una vez que se selecciona una película, el método muestra todos los actores que aparecen en esa película.

graphicData(self): Este método genera un gráfico utilizando la biblioteca Graphviz. Primero crea una lista de todos los actores que aparecen en la lista de películas. Luego utiliza el método "createGraphic" de la biblioteca Graphviz para crear un gráfico que muestra todas las películas en las que aparece cada actor.

```python
from src.controllers.Graphviz import createGraphic

class Movies:
    def __init__(self,listMovies):
        self.listMovies=listMovies


    def filterByActor(self, actorName):
        countdown=0
        actorName=actorName
        for element in self.listMovies:
            if actorName in element["Actores"]:
                countdown+=1
                print("PELICULA:",element["Pelicula"])
        if countdown==0:
            print("No se encontro el ACTOR que ha ingresado... \n")
            return False
        else:
            print("--------------------------------")
            return True

    def filterByYear(self,yearInput):
        countdown=0
        yearInput=yearInput
        for element in self.listMovies:
            if yearInput in element["Año"]:
                countdown+=1
                print("PELICULA:",element["Pelicula"], "| CATEGORIA: " , element["Categoria"])
        if countdown==0:
            print("El año ingresado no posee peliculas en nuestra base de datos...")
            return False
        else:
            print("--------------------------------")
            return True

    def filterByGenre(self,genreName):
        countdown=0
        genreName=genreName
        for element in self.listMovies:
            if genreName in element["Categoria"]:
                countdown+=1
                print("PELICULA:",element["Pelicula"])
        if countdown==0:
            print("No se encontró la CATEGORIA que ha ingresado...")
            return False
        else:
            print("--------------------------------")
            return True



    def getData(self):
        countdown=0
        listCopy=self.listMovies.copy()
        for element in self.listMovies:
            countdown+=1
            print(countdown ,"PELICULA:",element["Pelicula"] ,"| ACTORES:",element["Actores"] ,"| AÑO:",element["Año"] ,"| CATEGORIA:",element["Categoria"])
            print("--------------------------------")

        return listCopy


    def showActors(self):
        countdown=0
        for element in self.listMovies:
            countdown+=1
            print(countdown ,"PELICULA:",element["Pelicula"] )
            print("--------------------------------")
        try:
            option=int(input("Seleccione el numero de la pelicula... "))
            print("Los actores de la pelicula seleccionada son: ")




            for element in self.listMovies[option-1]["Actores"]:
                print(element)
        except:
            print("ERROR: EL VALOR INGRESADO NO ES VALIDO!!!")


    def graphicData(self):
        actors=[]
        for element in self.listMovies:
            for actor in element["Actores"]:
                actors.append(actor)
        actors=set(actors)

        createGraphic(self.listMovies,actors)
```

Descripción general del programa

Este programa es un sistema de gestión de películas que permite cargar un archivo de entrada, gestionar las películas que se encuentran en él, realizar filtrados por actor, año y género, y graficar la información de las películas.

Módulos de funciones importados

El programa importa dos módulos de funciones desde dos archivos diferentes:

- LoadFile desde el archivo LoadFile.py: Esta función se encarga de cargar un archivo de texto plano que contiene información sobre las películas. Esta información es almacenada en una lista que se utiliza en otras partes del programa.
- Movies desde el archivo Movies.py: Esta clase contiene las funciones que se utilizan para gestionar las películas y realizar los filtrados y graficados de información.

Variables globales

El programa declara dos variables globales:

- fileRequest: Es una lista que almacena la información sobre las películas que se cargan desde un archivo de entrada.
- movies: Es un objeto de la clase Movies, que se utiliza para llamar a las funciones que gestionan

las películas y realizan los filtrados y graficados de información.

Funciones principales

El programa tiene una estructura de bucle infinito que presenta un menú de opciones al usuario para que seleccione la tarea que desea realizar. A continuación se describen las funciones principales que se ejecutan en cada opción del menú:

Opción 1: Cargar archivo de entrada

La opción 1 del menú llama a la función LoadFile que se encarga de cargar un archivo de texto plano que contiene información sobre las películas. La ruta del archivo se especifica en la variable fileDirection.

La información cargada del archivo se almacena en la lista fileRequest. Esta lista se utiliza en otras partes del programa para llamar a las funciones de la clase Movies.

Opción 2: Gestionar películas

La opción 2 del menú presenta un submenú de opciones que permite gestionar las películas. Esta opción solo se puede ejecutar después de haber cargado un archivo de entrada en la opción 1.

El submenú de opciones tiene tres opciones:

- Mostrar Películas: Esta opción llama a la función getData de la clase Movies, que se encarga de

mostrar la información sobre todas las películas en el archivo de entrada.

- **Mostrar Actores: Esta opción llama a la función showActors de la clase Movies, que se encarga de mostrar los nombres de todos los actores que participan en las películas del archivo de entrada.**
- **Regresar al menú principal: Esta opción permite regresar al menú principal del programa.**

Opción 3: Filtrado

La opción 3 del menú presenta un submenú de opciones que permite filtrar la información de las películas por actor, año o género. Esta opción solo se puede ejecutar después de haber cargado un archivo de entrada en la opción 1.

El submenú de opciones tiene tres opciones:

- **Filtrar por actor: Esta opción llama a la función filterByActor de la clase Movies, que se encarga de mostrar las películas en las que un actor específico participa. El nombre del actor se ingresa por el usuario.**
- **Filtrar por año: Esta opción llama a la función filterByYear de la clase Movies, que se encarga de mostrar las películas que**

```python
#Importacion de modulos de funciones
from src.controllers.LoadFile import LoadFile
from src.classes.Movies import Movies


#Informacion general del programa
print('''Lenguajes Formales y de Programacion
       Seccion B+
       Carnet: 202111490
       Nombre: Andy Roberto Jimenez Macnab''')
input("Ingrese cualquier tecla....")


#Declaramos variables globales
fileRequest=[]
movies=""


while True:
    print("------------------------------------------------")
    print('''Ingrese la opcion que corresponda:

        1.Cargar archivo de entrada
        2.Gestionar Peliculas
        3.Filtrado
        4.Graficar
        5.Salir''')
    option=input("Ingrese la opcion que desea....  ")

    #Validacion por si el usuario ingresa una opcion incorrecta


    #Ejecucion de las funciones segun sea seleccionada por el usuario
    if option == "1":
        print("Ha seleccionado cargar archivo")
        fileDirection="./src/static/archivo prueba.lfP"
        fileRequest=LoadFile(fileDirection)
        movies=Movies(fileRequest)


    elif option =="2":


        while True:

            #Comprobacion de que el usuario haya ejecutado la funcion de cargar archivo primero para seleccionar cualquier otra
            if movies=="":
                print("ERROR: DEBE CARGAR UN ARCHIVO PRIMERO!!!")
                break

            print('''Seleccion una de las siguientes opciones
                    a. Mostrar Peliculas
                    b. Mostrar Actores
                    c. Regresar al menu principal''')
            option = input("Ingrese la letra de la opcion que desea... ")
            if option=="a":
                movies.getData()
            elif option=="b":
                movies.showActors()
            elif option=="c":
                break
            else:
                print("ERROR: DEBE INGRESAR UNA OPCION VALIDA!!!")


    elif option=="3":
        while True:

            if movies=="":
                print("ERROR: DEBE CARGAR UN ARCHIVO PRIMERO!!!")
                break

            print('''Seleccion una de las siguientes opciones
                a. Filtrar por actor
                b. Filtrar por año
                c. Filtrar por genero
                d. Regresar al menu principal''')
            option=input("Ingrese la letra de la opcion.... ")
            if option=="a":
                print("-----------------------------------")
                actorName=input("Ingrese el nombre del actor que desea buscar... ")
                movies.filterByActor(actorName)

            elif option=="b":
                print("-----------------------------------")
                yearInput=input("Ingrese el año que desea visualizar... ")
                movies.filterByYear(yearInput)

            elif option =="c":
                print("-----------------------------------")
                genreName=input("Ingrese el nombre de la categoria que desea visualizar... ")
                movies.filterByGenre(genreName)

            elif option=="d":
                break

            else:
                print("ERROR: DEBE INGRESAR UNA OPCION VALIDA!!!")


    elif option=="4":
        if movies=="":
            print("ERROR: DEBE CARGAR UN ARCHIVO PRIMERO!!!")
        else:
            movies.graphicData()


    elif option=="5":
        print("Saliendo del programa....")
        break

    else:
        print("ERROR: DEBE INGRESAR UNA OPCION VÁLIDA!!!")
```

Para graficar la información proporcionada se diseñó de la siguiente manera:

Primero se define una variable iteracion como 0, esta variable se utilizará para asignar un identificador a cada nodo de película.

Luego se define la función crear_nodo que recibe como parámetros el nombre de la película (pelicula), el año de la película (anio) y el género de la película (genero). Esta función utiliza la variable global iteracion para asignar un identificador único a cada nodo de película y devuelve una cadena de texto que representa el nodo en el formato de Graphviz.

La función crear_actor simplemente recibe como parámetro el nombre del actor (actor) y devuelve una cadena de texto que representa el nodo de actor en el formato de Graphviz.

La función crear_relacion recibe como parámetros el identificador del nodo de película (nodo) y el nombre del actor (actor) y devuelve una cadena de texto que representa la relación entre el nodo de película y el nodo de actor en el formato de Graphviz.

La función principal createGraphic recibe como parámetros la lista de películas (movies) y la lista de actores (actors). Primero se importa la librería os para poder crear un archivo y luego se define una variable data que contendrá todo el código de Graphviz.

Luego se crea un ciclo for que recorre la lista de películas y para cada película se obtiene el año, el género y el nombre de la película y se llama a la función crear_nodo para crear el nodo correspondiente. El resultado se agrega a la variable data.

Después se define el estilo de los nodos de actores y se crea un ciclo for que recorre la lista de actores y para cada actor se llama a la función crear_actor para crear el nodo correspondiente. El resultado se agrega a la variable data.

Por último, se crea un ciclo for anidado que recorre la lista de películas y para cada película se recorre la lista de actores. Para cada actor se llama a la función crear_relacion para crear la relación correspondiente entre el nodo de película y el nodo de actor. El resultado se agrega a la variable data.

Luego se escribe la variable data en un archivo llamado dataGraphic.dot y se utiliza el comando dot de Graphviz para crear un archivo PDF con la imagen del grafo.

```python
iteracion = 0

def crear_nodo(pelicula, anio, genero):
    global iteracion
    iteracion += 1
    return f'''\nnodo{iteracion} [label=<
    <table border="0" cellborder="1" cellspacing="0">
    <tr><td bgcolor="#0091ea" port="p1" colspan="2">{pelicula}</td></tr>
    <tr><td> {anio} </td><td> {genero} </td></tr>
    </table>>];\n\n'''

def crear_actor(actor):
    return f'\t"{actor}"\n' # Eso lo pueden omitir xd


def crear_relacion(nodo,actor):
    return f'''\tnodo{nodo}:p1 -> "{actor}";\n'''


def createGraphic(movies,actors):

    import os
    data = '''
    digraph main {
        graph [pad="0.5", nodesep="0.5", ranksep="2"];
        node [shape=plain]
        rankdir=LR;\n
    '''

    iteracion_2 = 1
    # Aqui creamos los nodos de peliculas

    for pelicula in movies:
        anio = pelicula["Año"]
        genero = pelicula["Categoria"]
        movie=pelicula["Pelicula"]
        nodo =crear_nodo(movie, anio, genero)
        data += nodo

    # Aqui agregamos el estilo a los nodos de actores
    data += 'node [shape=box, style=filled, fillcolor="#00c853"]'
    # Aqui creamos los nodos de actores
    for actor in actors:
        nodo = crear_actor(actor)
        data += nodo

    # Aqui creamos las relaciones
    for pelicula in movies:
        for actor in pelicula['Actores']:
            relacion = crear_relacion(iteracion_2,actor)
            data += relacion
        iteracion_2 += 1

    data += '}'

    # Aqui creamos el archivo
    with open('dataGraphic.dot', 'w') as f:
        f.write(data)

    # Aqui creamos la imagen
    os.system('dot -Tpdf dataGraphic.dot -o dataGraphic.pdf')
```

Entre los paradigmas de programación utilizados para este proyecto esta:

El paradigma de programación orientado a objetos (OOP). En este paradigma, los objetos se definen como entidades que tienen un estado (atributos) y un comportamiento (métodos). La clase Movie define un objeto Movie con los atributos title, year, genre, y actors. Los métodos definidos en la clase permiten acceder y modificar los atributos del objeto, y se definen para lograr el encapsulamiento, abstracción, herencia y polimorfismo. Además, se hace uso de otro paradigma de programación llamado programación defensiva, que se utiliza para evitar que los datos del objeto sean manipulados de manera inesperada o inapropiada. Por ejemplo, en el método add_actor, se valida que el actor no esté en la lista de actores antes de agregarlo.

En el segundo fragmento de código, se hace uso de varios patrones de diseño. El patrón de diseño Singleton se utiliza para la clase LoadFile, garantizando que solo haya una instancia de la clase en el programa. El patrón de diseño Factory Method se utiliza para la clase Movies, que utiliza un método de fábrica para crear objetos Movie a partir de los datos del archivo de entrada. Además, se hace uso del patrón de diseño Strategy, donde el método filterByActor, filterByYear y filterByGenre se definen

como estrategias para realizar diferentes tipos de filtrado de datos. Por último, el patrón de diseño Composite se utiliza para el método graphicData, donde se crea un objeto gráfico que contiene una serie de objetos gráficos para cada categoría de género.