Descripcion de los Tipos de Dato Abstracto (T.D.A):

Nombre de la clase: Menu.

Atributos de la clase:

- peliculas_vistas. Sera una lista de peliculas.
- peliculas_no_vistas. Sera una lista de peliculas.
- peliculas_recomendadas. Sera una lista de peliculas.

Metodos de la clase:

//Descripcion: Metodo constructor de objetos de clase Menu.
//Precondiciones: -.
//Postcondiciones: Le destina espacio del Heap a los atributos "peliculas_vistas", "peliculas_no_vistas"
//"peliculas_recomendadas" de un objeto de clase Menu.
+Menu();

//Descripcion: Se encarga de crear las listas de películas vistas y no vistas a partir de los archivos
//"peliculas_vistas.txt" y "peliculas_no_vistas.txt".
//Precondiciones: Le llegan como parametros, los nombres de los archivos de las peliculas vistas y no vistas.
//Postcondiciones: Las listas de peliculas vistas y no vistas son cargadas correctamente.
+vacio crear_lista_de_peliculas(cadena de caracteres peliculas_vistas_txt, cadena de caracteres peliculas_no_vistas_txt);

//Descripcion: Muestra por pantalla el menu principal, requiriendole al usuario que ingrese la opcion deseada.
//Precondiciones: -.
//Postcondiciones: El menu correspondiente es visualizado por el usuario.
+vacio interfaz();

//Descripcion: Metodo destructor de objetos de clase Menu.
//Precondiciones: -.
//Postcondiciones: Se libera la memoria destinada a alocar los atributos de un objeto de clase Menu
//"peliculas_vistas", "peliculas_no_vistas" y "peliculas_recomendadas".
+~Menu();

//Descripcion: Se encarga de leer los datos de los archivos ingresados por el usuario y guardarlos
//correctamente en las correspondientes listas de un objeto de clase Menu.
//Precondiciones: Le llegan como parametros, el nombre del archivo a analizar, junto con la lista en la cual se
//desean guardar los datos del correspondiente archivo.
//Postcondiciones: Los datos de los archivos son guardados eficientemente en las correspondientes listas.
-vacio leer_archivo(cadena de caracteres archivo_txt, puntero a lista lista_de_peliculas);

//Descripcion: Se imprimen por pantalla todos los datos correspondientes a una determinada pelicula.
//Precondiciones: Le llega como parametro la pelicula que se desea mostrar por pantalla.
//Postcondiciones: Los datos de la pelicula pasada como parametro son visualizados por el usuario.
-vacio mostrar_pelicula(puntero a pelicula pelicula);

//Descripcion: Se muestran por pantalla la lista de actores de una determinada pelicula.
//Precondiciones: Le llega como parametro, la lista de actores a mostrar por pantalla.

//Postcondiciones: La lista de actores pasada como parametro es visualizada por el usuario.
-vacio mostrar_actores(puntero a lista lista);

//Descripcion: De acuerdo con la opcion ingresada por el usuario, se efectua una determinada tarea correspondiente
//al menu principal.
//Precondiciones: Le llega como parametro la opcion ingresada por el usuario.
//Postcondiciones: Se ejecuta la tarea requerida por el usuario.
-vacio ejecutar_opcion(entero opcion);

//Descripcion: A partir de una pelicula vista y una no vista, se fija si hay coicidencia en, al menos, uno de los
//actores de estas dos peliculas.
//Precondiciones: Recibe una pelicula vista y una pelicula no vista.
//Postcondiciones: Devuelve true en caso de que, al menos, coincida un actor de ambas peliculas, o false
//en caso contrario.
-booleano coinciden_actores(puntero a pelicula peli_v, puntero a pelicula peli_nv);

//Descripcion: Se fija si el puntaje de una pelicula no vista que le llega como parametro es alto. Además, revisa
//si el director de la pelicula no vista coincide con el de la pelicula vista, que tambien le llega como parametro.
//Y tambien chequea si alguno de los actores coincide entre estas dos peliculas.
//Precondiciones: Recibe una pelicula vista y una pelicula no vista.
//Postcondiciones: En caso de cumpla alguna de estas condiciones de pelicula recomendable, devuelve true, o
//false en caso contrario.
-booleano cumple_condiciones_peli_recomendable(puntero a pelicula peli_v, puntero a pelicula peli_nv);

//Descripcion: Determina si una pelicula no vista puede considerarse como pelicula recomendable.
//Precondiciones: Le llega como paramatro la susodicha pelicula no vista.
//Postcondiciones: Devuelve true en caso de que la pelicula pueda considerarse recomendable, o false en caso contrario.
-booleano es_pelicula_recomendada(puntero a pelicula peli_nv);

//Descripcion: Muestra por pantalla todas las peliculas de una lista de peliculas determinada.
//Precondiciones: Le llega como parametro la lista de pelicula que se desea mostrar.
//Postcondiciones: Muestra por pantalla todas las peliculas pertenecientes a la lista que se pasa como parametro.
-vacio mostrar_lista_de_peliculas(puntero a lista lista);

//Descripcion: Agrega todas las peliculas que puedan considerarse recomendables, al atributo de un objeto de clase
//Menu "peliculas_recomendadas".
//Precondiciones: Las listas de peliculas vistas y no vistas tienen, al menos, una pelicula cada una.
//Postcondiciones: La lista de peliculas recomendadas es generada correctamente.
-vacio crear_lista_de_peliculas_recomendadas();

//Descripcion: Libera el contenido de las Lista(s) que contienen Pelicula(s).
//Precondiciones: -.
//Postcondiciones: Libera la memoria usada para cada Pelicula guardada.
-vacio limpiar_listas(Puntero a lista aux_lista);

-

Nombre de la clase: Lista.

Atributos de la clase:

- primero. Sera un puntero a un nodo.
- tam. Sera un entero. Su dominio es {0...100}.

Metodos de la clase:

```
//Descripcion: Metodo constructor de objetos de clase Lista.
//Precondiciones: -.
//Postcondiciones: Los atributos "primero" y "tam" de un objeto de clase Lista son igualados a
//NULL y cero respectivamente.
+Lista();

//Descripcion: Metodo destructor de objetos de clase Lista.
//Precondiciones: -.
//Postcondiciones: Se eliminan todos los nodos correspondientes a una lista. La liberacion de memoria
//se realiza en otro metodo que es llamado por este mismo.
+~Lista();

//Descripcion: Se agrega un nuevo nodo con el dato que le llega como parametro a un objeto de clase
//Lista en la posicion que tambien el metodo recibe como parametro.
//Precondiciones: Recibe el dato a agregar y la posicion en la lista en la cual se desea agregar este dato.
//"pos" es mayor o igual que uno, y menor o igual a "tam" mas uno.
//Postcondiciones: El dato es agregado correctamente al objeto de clase Lista.
+vacio agregar(Dato d, entero pos);

//Descripcion: Devuevle el tammanio del objeto de clase Lista.
//Precondiciones: -.
//Postcondiciones: El tamanio de la lista es devuelto correctamente.
+entero tamanio();

//Descripcion: Se devuelve el dato ubicado en la posicion "pos" de la lista. Esta posicion "pos" es
//recibida por el metodo como parametro.
//Precondiciones: Le llega como parametro la posicion de la cual se desea obtener el dato.
//"pos" es mayor o igual que uno, y menor o igual que "tam".
//Postcondiciones: El dato de la posicion "pos" de la lista es devuelto correctamente.
+Dato consultar(entero pos);

//Descripcion: Se elimina un nodo, correspondiente a la posicion "pos" de un objeto de clase Lista.
//"pos" es recibido por el metodo como parametro.
//Precondiciones: Le llega como parametro la posicion de la cual se desea eliminar un nodo.
//"pos" es mayor o igual que uno, y menor o igual que "tam".
//Postcondiciones: El nodo de la posicion "pos" es eliminado correctamente.
+vacio eliminar(entero pos);

//Descripcion: Se encarga de obtener el nodo correspondiente a la posicion "pos" de un objeto de
//clase Lista. "pos" es recibido por el metodo como parametro.
//Precondiciones: Le llega como parametro la posicion de la cual se desea obtener el nodo.
//"pos" es mayor o igual que uno, y menor o igual que "tam".
```

//Postcondiciones: El nodo correspondiente a la posicion "pos" de un objeto de clase Lista es
//devuelto correctamente.
-puntero a nodo obtener_nodo(entero pos);

//Descripcion: Verifica si el objeto de clase Lista tiene al menos un nodo.
//Precondiciones: -.
//Postcondiciones: En caso de que el tamanio de la lista sea cero devuelve true, o false en caso
//contrario.
-booleano lista_vacia();

Nombre de la clase: Nodo.

Atributos de la clase:

- dato. Sera un tipo de dato ingresado por el usuario a traves de una plantilla.
- siguiente. Sera un puntero a un nodo.

Metodos de la clase:

//Descripcion: Metodo constructor de objetos de clase Nodo.
//Precondiciones: Le llega como parametro el dato que se desea que almacene el nodo.
//Postcondiciones: El atributo de un objeto de clase Nodo "dato" es igualado a "d", mientras
//que el atributo "siguiente" es igualado a NULL.
+Nodo(Dato d);

//Descripcion: El atributo de un objeto de clase Nodo "dato" es asignado a "d", valor que el
//metodo recibe como parametro.
//Precondiciones: -.
//Postcondiciones: Se asigna correctamente el atributo de un objeto de clase Nodo "dato" al valor "d".
+vacio cambiar_dato(Dato d);

//Descripcion: El atributo de un objeto de clase Nodo "siguiente" es asignado a "ps", valor que el
//metodo recibe como parametro.
//Precondiciones: Recibe el puntero al cual se desea igualar el atributo "siguiente" de un objeto
//de clase Nodo.
//Postcondiciones: El atributo "siguiente" de un objeto de clase Nodo es asignado correctamente.
+vacio cambiar_siguiente(puntero a nodo ps);

//Descripcion: Devuelve el atributo "dato" de un objeto de clase Nodo.
//Precondiciones: -.
//Postcondiciones: El atributo "dato" de un objeto de clase Nodo es devuelto correctamente.
+Dato obtener_dato();

//Descripcion: Devuelve el atributo "siguiente" de un objeto de clase Nodo.
//Precondiciones: -.
//Postcondiciones: El atributo "siguiente" de un objeto de clase Nodo es devuelto correctamente.
+puntero a nodo obtener_siguiente();

Nombre de la clase: Pelicula.

Atributos de la clase:

- nombre. Sera una cadena de caracteres. La cadena podra tener como maximo 100 caracteres.
- genero. Sera una cadena de caracteres. La cadena podra tener como maximo 100 caracteres.
- puntaje. Sera un entero. Su dominio ira del 1 al 10.
- director. Sera una cadena de caracteres. La cadena podra tener como maximo 100 caracteres.
- actores. Sera un puntero a una lista de strings.

Metodos de la clase:

//Descripcion: Metodo constructor de objetos de clase Pelicula.
//Precondiciones: Recibe el nombre, genero, director y puntaje de la pelicula que se desea
//crear. Todos los valores que le llegan como parametro pertencen al dominio de los correspondientes
//atributos de un objeto de clase Pelicula.
//Postcondiciones: Los atributos de un objeto de clase Pelicula son asignados a las respectivas
//variables que se pasan como parametro.
+Pelicula(cadena de caracteres nombre, cadena de caracteres genero, cadena de caracteres director, entero puntaje);

//Descripcion: Destructor de la clase Pelicula. Libera la memoria usada para la Lista de "actores".
//Precondiciones: -.
//Postcondiciones: La memoria usada para la Lista de "actores" es liberada correctamente.
+~Pelicula();

//Descripcion: Devuelve el atributo "nombre" de un objeto de clase Pelicula.
//Precondiciones: -.
//Postcondiciones: El atributo "nombre" de un objeto de clase Pelicula es devuelto correctamente.
+cadena de caracteres obtener_nombre();

//Descripcion: Devuelve el atributo "genero" de un objeto de clase Pelicula.
//Precondiciones: -.
//Postcondiciones: El atributo "genero" de un objeto de clase Pelicula es devuelto correctamente.
+cadena de caracteres obtener_genero();

//Descripcion: Devuelve el atributo "director" de un objeto de clase Pelicula.
//Precondiciones: -.
//Postcondiciones: El atributo "director" de un objeto de clase Pelicula es devuelto correctamente.
+cadena de caracteres obtener_director();

//Descripcion: Devuelve el atributo "puntaje" de un objeto de clase Pelicula.
//Precondiciones: -.
//Postcondiciones: El atributo "puntaje" de un objeto de clase Pelicula es devuelto correctamente.
+entero obtener_puntaje();

//Descripcion: Agrega un actor a la lista de cadenas de caracteres "actores" de un objeto de clase Pelicula.
//Precondiciones: Le llega como parametro un actor, cuyo nombre pertenece al dominio del atributo

//"nombre" de un objeto de clase Pelicula.
//Postcondiciones: El actor es agregado correctamente a la lista de actores de un objeto de clase
//Pelicula.
+vacio agregar_actor(cadena de caracteres nuevo_actor);

//Descripcion: Devuelve el atributo "actores" de un objeto de clase Pelicula.
//Precondiciones: -.
//Postcondiciones: El atributo "actores" de un objeto de clase Pelicula es devuelto correctamente.
+puntero a lista obtener_lista_de_actores();
