Segundo Parcial
16/11/2020

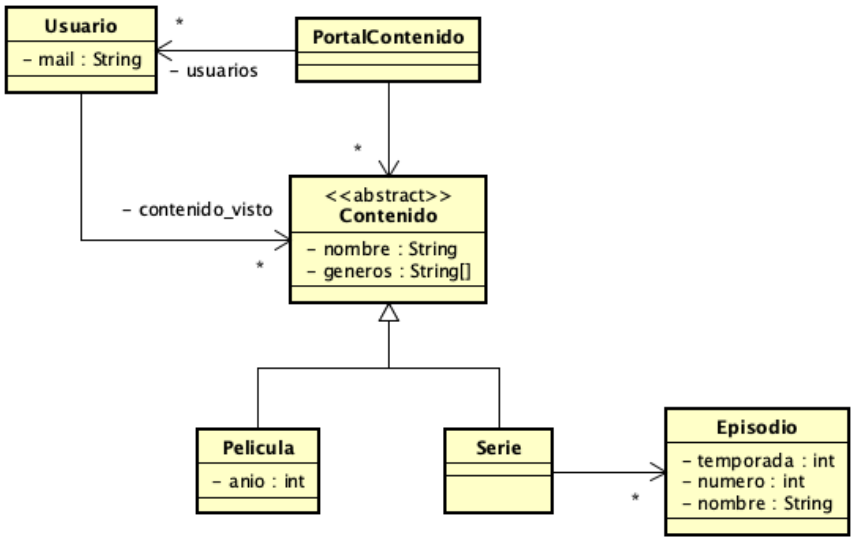
Tener en cuenta que:

- La duración del parcial es de 3 horas.
- Se debe realizar en computadora, para este fin se deberá seguir los siguientes pasos:
    1. Crear una carpeta dentro de su computadora de nombre segundoparcial2020.
        - En caso de realizar el parcial en una computadora del laboratorio usar la carpeta Z.
    2. Dentro de esta carpeta se deben crear todos los módulos necesarios para la resolución del problema planteado.
- Culminado las horas del parcial se deberá entregar a través del Moodle en la tarea "Entrega Segundo Parcial". Se espera que se suba un Zip de la carpeta "segundoparcial2020" creada en el paso 1. Puede usar de nombre del zip segundoparcial2020.zip.
- Tenga en cuenta que no se corrige código con errores del interprete de Python.
- No se puede usar material, ni uso de internet, ni celulares durante la realización del parcial.
- Se contestan dudas solo en la primer hora y media del parcial.

Ejercicio 1 - Practico (Tiempo recomendado 3:00h)

Una startup desea proveer un portal de contenidos de streaming al estilo Netflix o Amazon Prime. Se esta comenzando a evaluar el contenido a brindar y como este va a ser presentado al usuario. Se prevee que dentro del portal los usuarios van a podar visualizar Peliculas y Series y se quiere proveer un recomendor de contenidos que en base al historial de visualización sugiera contenidos a los usuarios.

Para ello se diseña el siguiente diagramas de clases parcial que modela la realidad a implementar:



Teniendo en esto en mente se debe implementar las siguientes operaciones en la clase PortalContenido:

- **crear_pelicula(self, nombre, generos, anio)** # Notar que se coloca el campo año como anio para evitar problemas de codificación del código fuente.
    - Esta operación registra una película que se identifica por su nombre, recibe una lista de géneros (Strings) y el año de estreno.
    - Lanza la excepción ContenidoYaExiste si un contenido con el nombre indicado ya se encuentra registrado (serie o película)

  - Lanza la excepción ContenidoInvalido si algunos de los argumentos son vacío (None).
- **crear_serie(self, nombre, generos)**
  - Esta operación registra una serie que se identifica por su nombre y recibe una lista de géneros (Strings).
  - Lanza la excepción ContenidoYaExiste si un contenido con el nombre indicado ya se encuentra registrado (serie o película)
  - Lanza la excepción ContenidoInvalido si algunos de los argumentos son vacíos (None).
- **agregar_episodio_serie(self, nombre_serie, numero_temporada, numero_episodio, nombre_episodio)**
  - Esta operación agrega un Episodio a una serie ya registrada.
  - Lanza la excepción ContenidoNoExiste si la serie no fue registrada aún.
  - Lanza la excepción ContenidoYaExiste si el episodio con el nombre indicado ya se encuentra creado dentro de la serie.
  - Lanza la excepción ContenidoInvalido si algún de los argumentos son vacíos (None).
- **visualizar_contenido(self, email_usuario, nombre_contenido)**
  - Esta operación registra una visualización de contenido para un usuario identificado por su email. En caso de que el usuario no este registrado, se debe crear con el email pasado.
  - Lanza la excepción ContenidoNoExiste si el contenido que el usuario desea visualizar no esta registrado.
- **recomendar_contenido(self, email_usuario)**
  - Esta operación debe retornar una lista de contenidos que el usuario se le recomienda visualizar en base a su historial de visualización. El recomendador debe seguir las siguientes reglas:
    - Si el usuario no visualizo contenido se debe retornar todo el contenido sin filtrar.
    - Solo se debe retornar contenido de los géneros que el usuario visualizo.
    - No se debe retornar contenido ya visto por el usuario.
  - Para la implementación de esta operación se debe implementar la operación abstracta en Contenido es_recomendado(self, generos_vistos), teniendo en cuenta que generos_vistos es una lista de géneros. Debe retornar True si el contenido cumple las reglas de recomendación o False en caso contrario.

Consideraciones

- La clase Contenido es abstracta y dispone de una operación abstracta es_recomendado(self, generos).
- Todas las clases se debe colocar en un módulo propio.
- Todas las clases sin ser las excepciones y la clase PortalContenido se deben colocar en el paquete entities. La clase PortalContenido se debe colocar en la raíz del proyecto. Las excepciones se deben colocar en el paquete exceptions.
- Se debe preservar el encapsulamiento de todas las entidades, para las cuales se deben usar getters (todas las propiedades) y setters (si son realmente necesarios).
- Se debe crear en club una operación “main” que pruebe el correcto funcionamiento de las operaciones implementadas dentro del módulo portal_contenido.py. Se espera al menos una prueba de cada operación.
- En caso de ser necesario saber si una instancia es del tipo Serie o Pelicula se debe utilizar la operación isinstance(instance, Type) provista por Python, que recibe como argumento una instancia y el tipo que se quiere validar, retorna True en caso que coincida el tipo falso en caso contrario. Un ejemplo para una instancia temp_contenido que se quiere validar si es Serie sería:
  - if isinstance(temp_contenido, Serie):
    - print(‘Es serie!’)