

Tarea # 1

Unidad 2. Relación entre

Clases

*Actividades*

**Actividad:** Ampliación de Contenidos Audiovisuales

# Descripción de la actividad:

Objetivo:

Ampliar y mejorar un proyecto Java utilizando conceptos avanzados de Programación Orientada a Objetos (POO), tales como herencia, asociación, agregación, y composición. Al final de esta actividad, los estudiantes habrán integrado nuevas clases en un proyecto existente, creado un diagrama de clases, y desarrollado un sistema funcional. También aprenderán a gestionar proyectos en GitHub

Instrucciones:

A partir del código de ejemplo de la Unidad 2 (Código Ejemplo 30), que contiene las clases ContenidoAudiovisual, Película, SerieDeTV y Documental (Este código se puede importar en Eclipse desde GitHub utilizando el siguiente enlace: [https://github.com/CS-Programacion-](https://github.com/CS-Programacion-Orientada-Objetos/poo_unidad1) [Orientada-Objetos/poo\_unidad1](https://github.com/CS-Programacion-Orientada-Objetos/poo_unidad1)), realizar las siguientes etapas de trabajo:

Etapa 1: Importación del Proyecto en Eclipse

* Paso 1: Abre Eclipse y selecciona la opción "Importar un proyecto desde GitHub".
* Paso 2: Utiliza el enlace proporcionado para clonar el repositorio en tu entorno de desarrollo.
* Paso 3: Asegúrate de que el proyecto se ha importado correctamente y está listo para ser modificado.

Etapa 2: Incorporación de Nuevas Clases

* Paso 1: Crea las siguientes tres clases adicionales dentro del proyecto: Actor, Temporada e Investigador.
  + Actor: Relacionada con Película,
  + Temporada: Relacionada con SerieDeTV.
  + Investigador: Relacionada con Documental.
* Paso 2: Implementa las relaciones apropiadas entre estas clases y las clases existentes (asociación, agregación, o composición).

Etapa 3: Diagrama de Clases

* Paso 1: Usa una herramienta de diagramación (como Lucidchart, draw.io, o la herramienta integrada de Eclipse) para crear un diagrama de clases del proyecto.
* Paso 2: Asegúrate de incluir las clases originales y las clases (Actor, Temporada, Investigador), reflejando claramente sus relaciones.
* Paso 3: Incorpora en el diagrama las relaciones entre las clases, especificando si son de tipo asociación, agregación, o composición.

Etapa 4: Expansión del Proyecto con Nuevas Subclases

* Paso 1: Crea dos nuevas subclases que hereden de ContenidoAudiovisual. Ejemplo: Video YouTube o Cortometraje. Estos son solo dos ejemplos de clases adicionales, se recomienda que sean clases iniciativa del estudiante, deben incorporar además atributos, constructores y métodos.
* Paso 2: Implementa y codifica las relaciones entre estas nuevas clases y las existentes, utilizando POO en Java.
* Paso 3: Actualiza el diagrama de clases para incluir estas nuevas subclases y sus relaciones.

Etapa 5: Desarrollo de un Sistema Funcional

* Paso 1: Asegúrate de que el sistema es totalmente funcional, implementando métodos que permitan la creación y manipulación de objetos de todas las clases.
* Paso 2: Prueba todas las funcionalidades del sistema, asegurándote de que no existan errores y que las relaciones entre clases se implementen correctamente.

Etapa 6: Gestión del Proyecto en GitHub

* Paso 1: Crea un repositorio personal en GitHub.
* Paso 2: Sube todo el proyecto al repositorio, asegurándote de que la estructura del proyecto sea clara y bien organizada.
* Paso 3: El proyecto debe contener al menos 3 commits adicionales.
* Paso 4: Incluye un archivo README.md, en el repositorio, donde describas:
  + Los objetivos y propósito del proyecto.
  + Las clases y funcionalidades nuevas que has añadido.
  + Instrucciones para clonar y ejecutar el proyecto.
  + Cualquier mejora adicional que hayas implementado (por ejemplo, pruebas unitarias, optimización de código, etc.).

Contenido Mínimo en GitHub

1. README.md
   * Descripción del Proyecto: Explicación clara del propósito de la aplicación, sus características principales, y a qué problema resuelve.
   * Instrucciones de Instalación y uso: Guía paso a paso para clonar el repositorio, instalar dependencias, y ejecutar la aplicación localmente. Ejemplos de cómo usar la aplicación, con comandos y capturas de pantalla si es necesario.
2. Estructura de Carpetas
   * Se da una estructura básica de las carpetas en GitHub.
   * src/: Carpeta que contiene el código fuente de la aplicación.
   * main/java: Código principal de la aplicación.

Entrega

Crear un informe PDF indicando los aspectos más importantes implementados en cada etapa y sobre todo el enlace a GitHub del proyecto.

**Extensión máxima:** 5 páginas (Times New Roman o Calibri Light, tamaño 12 e interlineado 1,5).

**Nombre del fichero:** “primerApellido\_primerNombre\_siglasAsignatura\_U#”, ejemplo: Lopez\_Juan\_ComSisElecRedElecInt\_U1

# Formato de archivo a subir: PDF

**Rúbrica:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterios** | **Nivel Bajo** | **Nivel Medio** | **Nivel Alto** | **Sub- Puntajes** |
| **1. Importación del Proyecto en Eclipse** | No logra importar correctamente el proyecto en Eclipse o hay errores críticos de configuración.  **0** | Importa el proyecto con algunos errores menores que no afectan la funcionalidad general.  **0.5** | Importa correctamente el proyecto sin errores de configuración.  **1** | **1** |
|  | Las nuevas clases no | Las nuevas clases están definidas pero requieren mejoras en la estructura o en la definición de relaciones.  **1** | Las nuevas clases están |  |
|  | están bien definidas o | correctamente definidas |  |
| **2. Incorporación**  **de Nuevas Clases** | faltan elementos  importantes como | con atributos, métodos  y relaciones claros y | **2** |
|  | atributos y métodos. | bien implementados. |  |
|  | **0** | **2** |  |
| **3. Diagrama de Clases** | El diagrama de clases es incompleto, carece de detalles importantes o no sigue las convenciones UML.  **0** | El diagrama de clases incluye la mayoría de las clases y relaciones, pero hay algunos errores o faltan detalles.  **1** | El diagrama de clases es claro, completo y sigue todas las convenciones UML adecuadamente.  **2** | **2** |
|  | Las subclases añadidas | Las subclases añadidas | Las subclases están bien |  |
| **4. Expansión del Proyecto con Nuevas Subclases** | no muestran un uso  adecuado de la herencia  o las relaciones entre clases. | muestran un uso adecuado de  la herencia pero requieren  mejoras en la lógica o implementación. | implementadas,  haciendo uso correcto  de la herencia y de las relaciones entre clases. | **2** |
|  | **0** | **1** | **2** |  |
|  | El sistema desarrollado | El sistema desarrollado es | El sistema es |  |
|  | es mínimo y tiene fallos | funcional pero presenta | completamente |  |
| **5. Desarrollo de un**  **Sistema Funcional** | graves que impiden su  correcto | algunos errores o áreas de  mejora en su lógica y | funcional, con una  lógica clara, eficiente y | **3** |
|  | funcionamiento. | eficiencia. | sin errores aparentes. |  |
|  | **0.75** | **1.5** | **3** |  |
| **6. Gestión del Proyecto en GitHub** | El uso de GitHub es mínimo, con pocos commits o mala organización de los  repositorios. | El uso de GitHub es adecuado, pero puede mejorar en cuanto a la frecuencia de commits y la claridad en la documentación. | El uso de GitHub es excelente, con commits regulares, descripciones claras y una gestión  eficiente del repositorio. | **2** |
|  | **0** | **1** | **2** |  |
| **Totales** | | | | **12** |