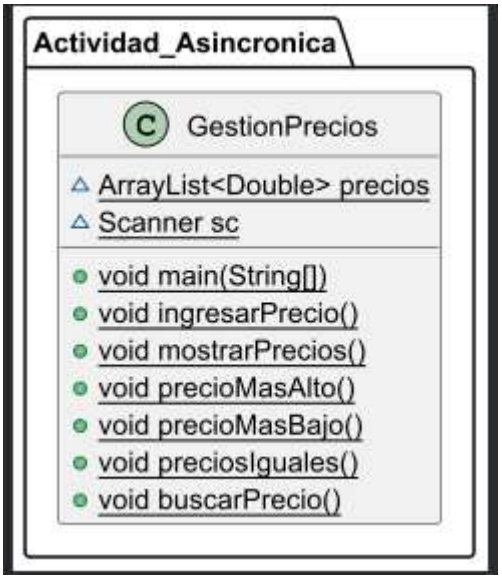
ACTIVIDAD SEMANA 8

EJERCICIO #1



EJERCICIO #2

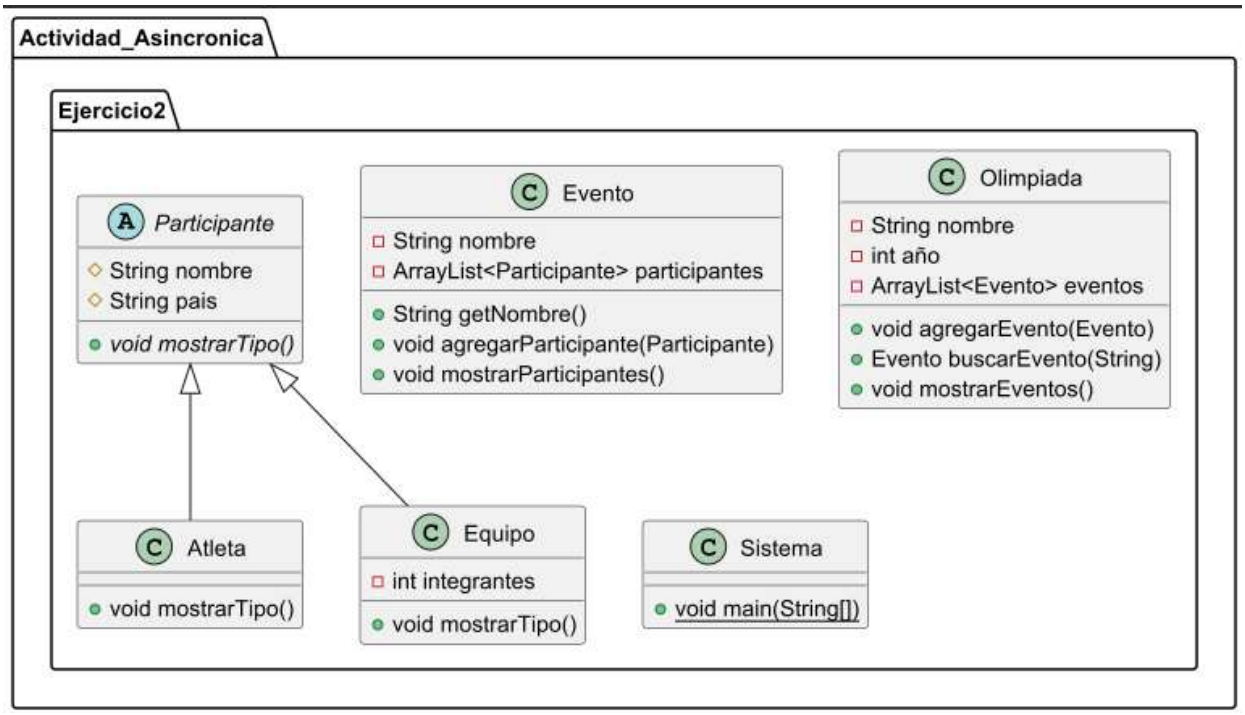
Sistema de Registro de Olimpiadas

Desarrolle un programa en Java que permita registrar información básica sobre unas Olimpiadas, sus eventos y los participantes. El objetivo es aplicar conceptos fundamentales de la Programación Orientada a Objetos, tales como clases, herencia, polimorfismo, manejo de listas (ArrayList) y excepciones personalizadas.

Diagrama UML (Descripción)

El sistema debe incluir un conjunto de clases que se relacionan mediante herencia y composición. El diagrama UML debe mostrar:

- Una clase abstracta Participante.
- Clases hijas Atleta y Equipo.
- La clase Evento, que contiene una lista de participantes.
- La clase Olimpiada, que contiene una lista de eventos.



Subir el Git Hub Readme detalle del trabajo realizado, Enunciado,captura de código y resultados .

PREGUNTAS Y RESPUESTAS GUÍA – EJERCICIO #1 (ArrayList – Gestión de Precios)

1. ¿Qué estructura de datos utiliza el ejercicio y por qué es adecuada?

El ejercicio utiliza un ArrayList<Double>, porque permite almacenar una cantidad variable de precios, agregar y consultar elementos de forma dinámica sin necesidad de definir un tamaño fijo.

2. ¿Por qué es necesario validar que el precio ingresado sea mayor que cero?

Porque un precio no puede ser negativo ni cero en un contexto real. La validación garantiza que los datos registrados sean válidos y tengan sentido en el sistema.

3. ¿Cómo se obtiene el precio más alto recorriendo el ArrayList?

Se recorre el ArrayList comparando cada elemento con una variable temporal.
Cada vez que se encuentra un valor mayor, se actualiza esa variable.
Al terminar el recorrido, esa variable contiene el precio más alto.

4. ¿Qué condición debe cumplirse para mostrar "No existen precios registrados"?

Debe cumplirse que el ArrayList esté vacío, es decir, que precios.isEmpty() sea verdadero.

5. ¿Cómo funciona la opción de buscar un precio específico?

El usuario ingresa un precio y el sistema usa el método contains() del ArrayList para verificar si ese valor existe.
Si contiene el valor, se muestra "Precio encontrado", caso contrario "No existe".

6. ¿Por qué es recomendable separar las funcionalidades en métodos como ingresarPrecio(), mostrarPrecios(), precioMasAlto(), etc.?

Porque mejora la organización, legibilidad, reutilización y mantenimiento del código. Cada método cumple una única responsabilidad, lo cual facilita pruebas y modificaciones.

7. ¿Qué estructura de repetición permite mostrar el menú hasta que el usuario decida salir y por qué?

Se usa un while o do-while, porque permiten repetir el menú continuamente hasta que el usuario ingrese la opción de salir, controlando así el ciclo del programa.

PREGUNTAS Y RESPUESTAS GUÍA – EJERCICIO #2

1. **¿Qué parte del diseño aplica encapsulamiento?**

Los atributos privados dentro de las clases (Participante, Atleta, Equipo, Evento, Olimpiada), junto con sus getters y setters, protegen los datos del acceso directo y permiten un manejo seguro**.**

2. **¿Cómo se aplica la herencia en este sistema?**

Las clases Atleta y Equipo heredan de la clase abstracta Participante, adquiriendo sus atributos y comportamientos básicos.

3. **¿Por qué la clase Participante debe ser abstracta?**

Porque representa un concepto general que no debe instanciarse directamente.
Un participante siempre será un Atleta o un Equipo, por lo que el comportamiento común se define en la clase abstracta.

4. **¿Dónde se observa el polimorfismo dentro del sistema?**

Cuando se sobrescribe el método abstracto mostrarTipo() o un método similar en Atleta y Equipo, y ambos pueden ser tratados como Participante, ejecutando comportamientos distintos.

5. **¿Qué propósito tiene sobrescribir métodos en las clases hijas?**

   Permite que cada tipo de participante (Atleta o Equipo) tenga su propia implementación del método heredado, adaptándose a sus características específicas.

6. **¿Cuándo se utilizan excepciones personalizadas en este ejercicio?**

Cuando los datos ingresados no cumplen los requisitos, como nombres vacíos, eventos sin participantes o información insuficiente.
Las excepciones permiten manejar estos errores de forma controlada.

7. **¿Qué papel cumple la clase Evento dentro del sistema?**

   Representa cada competencia individual de la Olimpiada y contiene una lista de participantes, permitiendo agregarlos, listarlos y gestionarlos.

8. **¿Qué representa la clase Olimpiada dentro del modelo UML?**

   Es la clase principal del sistema.
   Almacena los datos generales de la olimpiada y mantiene una lista de eventos, mostrando una relación de composición.

9. **¿Por qué el UML debe incluir herencia y composición?**

   Herencia: para mostrar la relación entre Participante, Atleta y Equipo.

   Composición: porque Evento “tiene” participantes y Olimpiada “tiene” eventos.
   Ayuda a entender mejor cómo se estructura el sistema.

10. **¿Por qué se usa un ArrayList en lugar de un arreglo normal?**

    Porque un ArrayList permite agregar o eliminar elementos dinámicamente sin definir un tamaño fijo, algo esencial cuando el número de eventos o participantes puede cambiar.