Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Primer Parcial		
Apellido(1):		Fecha:	
Nombre/s(1):		Docente a cargo(2):	
División(1):		Nota(2):	
DNI(1):		Firma(2):	

(1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
(2) Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

Objetivo General

De acuerdo con las descripciones de las siguientes clases, se pide: Modelar en **UML** (adjuntando la imagen en la entrega) y posteriormente crear el código correspondiente en **Java** (en todos los casos, reutilizar código).

Desarrollar un sistema para la gestión de eventos culturales, implementando un CRUD completo para gestionar los eventos y garantizando la persistencia de los datos en archivos mediante serialización. Este proyecto integrará los conceptos de Programación Orientada a Objetos en Java, incluyendo herencia, polimorfismo y la organización en paquetes.

Contexto

Un centro cultural requiere un sistema que gestione eventos, organizadores, y asistentes, manteniendo un historial guardado en archivos para análisis futuro.

Funcionalidades CRUD

- Crear: Registrar nuevos eventos culturales.
- Leer: Consultar eventos individuales o mostrar la lista completa.
- Actualizar: Modificar los datos de un evento existente.
- Eliminar: Eliminar eventos registrados.

Estructura del Proyecto

1. Herencia y Polimorfismo

- Jerarquía de Productos

Crear una jerarquía de clases para representar diferentes tipos de eventos:

- Clase abstracta Evento:
  - Atributos comunes:
    - String codigo
    - String titulo
    - LocalDate fecha
    - String organizador
    - int capacidadMaxima
  - Métodos abstractos:
    - boolean estaLleno()
    - void mostrarDetalles()
- Clases hijas:
  - Concierto:
    - Atributos adicionales:
      - String artistaPrincipal
      - String generoMusical
    - Método específico: boolean esMusicaClasica()
  - Conferencia:
    - Atributos adicionales:
      - String tema
      - List<String> panelistas
    - Método específico: int cantidadPanelistas ()

2. Serialización y Manejo de Archivos
   - Guardar la lista de eventos en un archivo usando **serialización**.
   - Cargar eventos desde un archivo para su consulta.
3. Excepciones y Paquetes
   - Excepciones personalizadas:
     - EventoNoEncontradoException
     - CapacidadExcedidaException
   - Paquetes sugeridos:
     - com. cultura. eventos
     - com. cultura. gestores

- com. cultura. excepciones

4. Interfaz Funcional y Análisis
    - Interfaz funcional FiltroEvento para filtrar eventos por fecha, tipo, o capacidad restante.
    - Generar estadísticas de asistencia en formato de texto o JSON.
5. Extras
    - Implementar una funcionalidad para exportar los eventos como un archivo CSV.
    - (Bonus) Crear una interfaz gráfica para agregar y consultar eventos.

MAIN

```
import com.cultura.eventos.*;

import com.cultura.gestores.*;

import com.cultura.excepciones.*;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        GestorEventos gestor = new GestorEventos();

        Scanner scanner = new Scanner(System.in);

        int opcion;

        do {

            System.out.println("\n--- Gestión de Eventos Culturales ---");

            System.out.println("1. Registrar Evento");

            System.out.println("2. Mostrar Todos los Eventos");

            System.out.println("3. Buscar Evento por Código");

            System.out.println("4. Modificar Evento");
```

```
            System.out.println("5. Eliminar Evento");
            System.out.println("6. Guardar Eventos en Archivo");
            System.out.println("7. Cargar Eventos desde Archivo");
            System.out.println("8. Salir");
            System.out.print("Seleccione una opción: ");
            opcion = scanner.nextInt();
            switch (opcion) {
                case 1 -> gestor.registrarEventoDesdeConsola(scanner);
                case 2 -> gestor.mostrarEventos();
                case 3 -> gestor.buscarEventoPorCodigoDesdeConsola(scanner);
                case 4 -> gestor.modificarEventoDesdeConsola(scanner);
                case 5 -> gestor.eliminarEventoDesdeConsola(scanner);
                case 6 -> gestor.guardarEventos("eventos.dat");
                case 7 -> gestor.cargarEventos("eventos.dat");
                case 8 -> System.out.println("Saliendo del sistema...");
                default -> System.out.println("Opción inválida. Intente nuevamente.");
            }
        } while (opcion != 8);
        scanner.close();
    }
}
```

IMPORTANTE:

- Dos (2) errores en el mismo tema anulan su puntaje.
- NO se corregirán exámenes que NO compilen.

- NO se corregirán exámenes que NO contengan la imagen del modelado en UML.
- No se corregirán proyectos que no sea identificable su autor.
- Reutilizar tanto código como sea posible.
- Colocar nombre de la clase (en estáticos), this o super en todos los casos que corresponda.
- Subir los proyectos y la imagen UML en un único archivo .7z, .zip, .rar o similar. Nombrarlo con su Apellido.Nombre.

Consideraciones Finales:

- Fecha de entrega: 29 de noviembre de 2024 (18.30 hs)
- Trabajo individual
- Presentación oral y defensa del proyecto (29 de noviembre de 2024)