UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Segundo Parcial		
Apellido(1):		Fecha:	28 nov 2025
Nombre/s(1):		Docente a cargo(2):	Baus / Rocha
División(1):	122-1	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.

Sistema de Gestión de Personajes de Videojuegos(V1.1)

Se te pide implementar un sistema para gestionar **personajes de videojuegos**.
Cada personaje tendrá atributos como **ID**, **nombre**, **creador** y un **rol** asociado (tanque, mago, arquero, etc.).
El sistema debe permitir agregar, obtener, eliminar, filtrar, ordenar, guardar y cargar desde archivos binarios y CSV.

---

1. Clases involucradas

Personaje

Debe contener los siguientes atributos:

- **id** (entero)
- **nombre** (String)
- **creador** (String) — desarrollador o estudio (por ej., “Nintendo”, “Capcom”, “Valve”)
- **rol** (enum RolPersonaje)

Debe poder ordenarse naturalmente **por id en orden ascendente**.

RolPersonaje

Enum con roles típicos de videojuegos:

- TANQUE
- MAGO
- ARQUERO
- ASESINO
- SANADOR
- INGENIERO

ColeccionPersonajes

Implementa un inventario genérico que pueda almacenar **cualquier tipo de objeto que implemente CSVSerializable**.
Debe permitir:

- Agregar, eliminar y obtener personajes por índice
- Filtrar personajes según un criterio
- Ordenar personajes:
  - De forma natural (Comparable, por id)
  - Mediante un Comparator (por ejemplo, por nombre o creador)
- Guardar y cargar desde archivos binarios
- Guardar y cargar desde archivos CSV

CSVSerializable
Interfaz con:

- toCSV()

La clase **Personaje** debe implementar además:

- static fromCSV(String linea)

---

2. Operaciones del sistema
El sistema debe permitir:

- Agregar personajes
- Obtener personajes por índice
- Eliminar por índice
- Filtrar según distintos criterios
- Ordenar personajes (natural y Comparator)
- Guardar el inventario en un archivo binario
- Cargar desde archivo binario
- Guardar en archivo CSV
- Cargar desde archivo CSV

---

3. Filtrado
Debes implementar al menos dos filtros:

1. **Filtrar personajes por rol**
   Ejemplo: mostrar solo los personajes de tipo MAGO.
2. **Filtrar personajes cuyo nombre contenga una palabra clave**
   Ejemplo: “Mario”, “Link”, “Shadow”, “Mega”.

---

4. Métodos CSV
Ejemplo:
3,Shadow Assassin,Sega,ASESINO

---

5. Entrega

Debes entregar:

- Proyecto completo
- Métodos para ordenamiento, filtrado y persistencia
- Se debe utilizar el siguiente método main, el cual **NO puede modificarse**, excepto:
  - rutas hardcodeadas
  - agregado de expresiones Lambda

La estructura del proyecto debe respetar la organización en paquetes vista en la cátedra.

```
public class Main {

    public static void main(String[] args) {

        try {
            ColeccionPersonajes<Personaje> coleccion = new ColeccionPersonajes<>();

            coleccion.agregar(new Personaje(1, "Arthas", "Blizzard",
RolPersonaje.TANQUE));
            coleccion.agregar(new Personaje(2, "Zelda Mage", "Nintendo",
RolPersonaje.MAGO));
            coleccion.agregar(new Personaje(3, "Shadow Archer", "Sega",
RolPersonaje.ARQUERO));
            coleccion.agregar(new Personaje(4, "MedBot 2.0", "Valve",
RolPersonaje.SANADOR));
            coleccion.agregar(new Personaje(5, "Tech Engineer", "Capcom",
RolPersonaje.INGENIERO));

            // Mostrar todos los personajes
            System.out.println("Personajes:");
            coleccion.paraCadaElemento(/* Lambda */);

            // Filtrar por rol MAGO
            System.out.println("\nPersonajes MAGO:");
            coleccion.filtrar(/* Lambda */)
                     .forEach(/* Lambda */);

            // Filtrar por nombre que contenga "Shadow"
            System.out.println("\nPersonajes que contienen 'Shadow':");
            coleccion.filtrar(/* Lambda */)
                     .forEach(/* Lambda */);

            // Ordenar por ID (orden natural)
            System.out.println("\nPersonajes ordenados por ID:");
            coleccion.ordenar(/* Lambda */);
            coleccion.paraCadaElemento(/* Lambda */);

            // Ordenar personajes por nombre
            System.out.println("\nPersonajes ordenados por nombre:");
            coleccion.ordenar(/* Lambda */);

            // Guardar en archivo binario
            coleccion.guardarEnArchivo("src/data/personajes.dat");

            // Cargar desde archivo binario
            ColeccionPersonajes<Personaje> cargado = new ColeccionPersonajes<>();
```

```
            cargado.cargarDesdeArchivo("src/data/personajes.dat");

            System.out.println("\nPersonajes cargados desde archivo binario:");
            cargado.paraCadaElemento(/* Lambda */);

            // Guardar en archivo CSV
            coleccion.guardarEnCSV("src/data/personajes.csv");

            // Cargar desde archivo CSV
            cargado.cargarDesdeCSV("src/data/personajes.csv", /* Lambda */);

            System.out.println("\nPersonajes cargados desde archivo CSV:");
            cargado.paraCadaElemento(/* Lambda */);

        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```