Guía 6 Ejercicios Programación II: Ordenamiento

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

1. **Ordenar Productos por Precio Ascendente**
   Crea una clase Producto con atributos nombre y precio. Implementa la interfaz Comparable<Producto> para que los productos se ordenen de manera natural por precio de menor a mayor. Luego, crea una clase ComparadorPorNombre que implemente Comparator<Producto> para ordenar los productos por nombre alfabéticamente.

2. **Ordenar Personas por Edad y Luego por Nombre**
   Crea una clase Persona con atributos nombre y edad. Implementa Comparable<Persona> para ordenar primero por edad de menor a mayor. Luego, utiliza Comparator<Persona> para ordenar alfabéticamente por nombre.

3. **Ordenar Libros por Título Inverso**
   Crea una clase Libro con atributo titulo. Implementa Comparator<Libro> para ordenar los libros por título en orden inverso.

4. **Ordenar Empleados por Salario y Después por Nombre**
   Crea una clase Empleado con atributos nombre y salario. Usa Comparable<Empleado> para ordenar por salario de menor a mayor, y Comparator<Empleado> para ordenar por nombre.

5. **Ordenar Productos por Precio y, si son iguales, por Nombre**
   Extiende el ejercicio 1 para que Producto se ordene primero por precio (de menor a mayor) y si el precio es igual, por nombre alfabéticamente.

6. **Ordenar Estudiantes por Nota y por Edad Descendente**
   Crea una clase Estudiante con atributos nombre, edad y notaPromedio. Implementa la interfaz Comparable<Estudiante> para que los estudiantes se ordenen de manera natural por su nota promedio de mayor a menor. Luego, crea un Comparator<Estudiante> para ordenar los estudiantes por edad de mayor a menor.

7. **Ordenar Juegos por Duración y luego por Género**
   Crea una clase Juego con atributos titulo, duracion (en horas) y genero. Implementa Comparable<Juego> para ordenar los juegos por duración de menor a mayor. Luego, crea un Comparator<Juego> para ordenar por género alfabéticamente.

8. **Ordenar Cuentas Bancarias por Saldo Ascendente y Luego por Número de Cuenta Descendente**
   Crea una clase `CuentaBancaria` con atributos `numeroCuenta` (de tipo `String`) y `saldo`. Implementa `Comparable<CuentaBancaria>` para ordenar las cuentas por saldo de menor a mayor. Luego, crea un `Comparator<CuentaBancaria>` para ordenar las cuentas por número de cuenta en orden descendente.