Actividad práctica Crud

Guía Práctica

Problema:

Desarrolla una aplicación que gestione los alumnos inscritos en el gimnasio de **DUOC UC Puerto Montt**. La aplicación debe permitir realizar las siguientes acciones:

- **Agregar alumno**: Los datos a ingresar son:
  - **ID del Alumno**: Un número único que identifica a cada alumno.
  - **Nombre completo del alumno**: El nombre completo del alumno.
  - **Edad**: La edad del alumno.
  - **Tipo de Membresía**: Tipo de membresía adquirida (Mensual, Trimestral, Anual).
- **Mostrar alumnos**: Visualiza la lista de todos los alumnos inscritos en el gimnasio.
- **Actualizar alumno**: Permite actualizar los datos de un alumno existente, identificándolo mediante su ID.
- **Eliminar alumno**: Elimina un alumno del sistema, identificándolo mediante su ID.

Instrucciones paso a paso para implementar el CRUD:

Paso 1: Crear la clase Alumno

1. Define una clase que represente a un alumno.
2. Los atributos que debe tener la clase son: `id`, `nombre`, `edad`, y `membresia`.
3. Define un constructor para la clase que reciba estos valores como parámetros y los asigne a los atributos.

4. Crea métodos ***getter* y *setter*** para acceder y modificar cada uno de los atributos.
5. Añade un método `toString()` para mostrar los datos del alumno de forma clara cuando se imprima.

Paso 2: Crear la clase Gimnasio

1. Crea una nueva clase que será responsable de gestionar la lista de alumnos.
2. Utiliza un `ArrayList` para almacenar objetos de tipo `Alumno`.
3. Añade un constructor que inicialice la lista y cualquier otro recurso que sea necesario (por ejemplo, un objeto `Scanner` para leer los datos).
4. Implementa los siguientes métodos en la clase **Gimnasio**:
    - **Agregar alumno**:
        1. Pide al usuario que ingrese el ID, nombre, edad y tipo de membresía del alumno.
        2. Crea un objeto de tipo `Alumno` con los datos ingresados.
        3. Añade el objeto a la lista de alumnos.
    - **Mostrar alumnos**:
        1. Recorre la lista de alumnos.
        2. Muestra cada alumno usando el método `toString()` de la clase `Alumno`.
        3. Si no hay alumnos en la lista, muestra un mensaje indicando que la lista está vacía.
    - **Actualizar alumno**:
        1. Solicita al usuario el ID del alumno que desea actualizar.
        2. Busca al alumno en la lista por su ID.
        3. Si el alumno es encontrado, pide los nuevos valores (nombre, edad, tipo de membresía) y actualiza los atributos correspondientes.
        4. Si no se encuentra el alumno, muestra un mensaje de error.
    - **Eliminar alumno**:

1. Solicita al usuario el ID del alumno que desea eliminar.
2. Busca al alumno en la lista por su ID.
3. Si lo encuentra, elimina el alumno de la lista.
4. Si no se encuentra, muestra un mensaje de error.

Paso 3: Crear la clase Main para interactuar con el usuario

1. En la clase principal (`Main`), crea una instancia de la clase **Gimnasio**.
2. Diseña un menú que ofrezca las opciones para agregar, mostrar, actualizar y eliminar alumnos.
3. Utiliza un bucle `do-while` para mantener el menú activo hasta que el usuario elija la opción de salir.
4. Por cada opción seleccionada, llama al método correspondiente de la clase **Gimnasio** para realizar la operación deseada.
5. Asegúrate de que el menú maneje correctamente las entradas no válidas y siga funcionando sin errores.

Paso 4: Carga el código de la actividad en tu repositorio de GitHub

1. Comparte el link con el Docente.

Recomendaciones:

- Organiza el código en diferentes archivos en Visual Studio Code: uno para cada clase.
- Asegúrate de utilizar la consola para interactuar con el usuario en cada paso del programa.
- Recuerda validar las entradas del usuario para evitar errores en los datos ingresados.

Objetivo:

Al final de esta práctica, deberás tener una aplicación funcional que permita gestionar los alumnos del gimnasio de DUOC UC Puerto Montt, utilizando el menú

para realizar todas las operaciones CRUD.