ACTIVIDAD
IMPORTANCIA DE LA ORIENTACIÓN A OBJETOS EN LA PROGRAMACIÓN, CLASES, OBJETOS, ATRIBUTOS Y MÉTODOS

Aprendizaje esperado

Utilizar elementos de la programación orientada a objetos para implementación de una pieza de software que da solución a un problema de baja complejidad.

Actividad

Contexto:
Una institución educativa desea implementar un sistema que permita gestionar de forma eficiente la relación entre estudiantes, cursos, docentes y evaluaciones. Esta aplicación debe reflejar fielmente los principios de la Programación Orientada a Objetos, considerando un diseño escalable y reutilizable que pueda expandirse en futuras versiones.

El proyecto será desarrollado gradualmente, conforme al avance en los contenidos del curso, incluyendo el uso de clases, atributos, métodos, composición, colaboración entre objetos, encapsulamiento, modificadores de acceso, entre otros.

Objetivo de la Actividad:
Diseñar e implementar una aplicación de consola en Java que modele un entorno académico en el que se representen estudiantes, cursos, docentes y evaluaciones, aplicando correctamente los fundamentos de la programación orientada a objetos. El sistema debe gestionar registros, relaciones, cálculos y reportes internos, fomentando el uso de buenas prácticas y convenciones de codificación.

Requerimientos del Sistema:
1. Gestión de Estudiantes

- Registrar estudiantes con RUT, nombre, carrera y año de ingreso.
- Asociar estudiantes a uno o varios cursos.
- Obtener el promedio general del estudiante basado en sus evaluaciones.

2. Gestión de Docentes

- Registrar docentes con RUT, nombre, área de especialización.
- Asociar docentes a uno o más cursos que dictan.

3. Gestión de Cursos

- Crear cursos con código, nombre, número de créditos, y docente asignado.
- Gestionar la inscripción de estudiantes a cursos.
- Asociar evaluaciones a los cursos (exámenes, tareas, controles).

4. Evaluaciones

- Crear evaluaciones asociadas a cursos con tipo (Examen, Trabajo, Control), fecha y puntaje máximo.
- Asignar calificaciones individuales por estudiante para cada evaluación.
- Calcular automáticamente el promedio del curso y de cada estudiante.

5. Reportes y Consultas

- Listar todos los estudiantes inscritos en un curso.
- Mostrar las calificaciones de un estudiante en todos sus cursos.
- Reportar los cursos con promedios más altos.
- Mostrar docentes con mayor carga académica (número de cursos).

Especificaciones Técnicas:

1. **Clases sugeridas**: Estudiante, Docente, Curso, Evaluacion, Inscripcion, SistemaAcademico.
2. **Encapsulamiento completo**: atributos privados con métodos get y set.
3. **Uso de constructores sobrecargados** y **métodos específicos** que representen acciones reales (ej. calcularPromedioEstudiante(), agregarEvaluacionCurso()).

4. **Relaciones de composición y colaboración**:
    - Un curso tiene una lista de evaluaciones.
    - Un estudiante tiene múltiples inscripciones.
    - Un docente puede dictar múltiples cursos.
5. **Uso de estructuras de datos** (ArrayList, HashMap) para colecciones dinámicas.
6. **Aplicación de control de acceso y validaciones**, como:
    - Evitar duplicidad de inscripciones.
    - Validar que las notas estén dentro del rango permitido.
7. **Uso de la clase String** para búsquedas, filtros y formatos.
8. **Uso de la clase Math** para cálculos de promedios, redondeos, etc.
9. **Gestión de fechas** (opcional) con java.time para evaluaciones.

Interacción de Consola:

Crear un **menú principal interactivo**, con opciones para:

- Registrar estudiantes, docentes y cursos.
- Gestionar inscripciones y evaluaciones.
- Consultar promedios, listar datos y generar reportes.
-

Debe utilizarse una estructura clara de sentencias, bloques y condiciones, evitando duplicación de código, y estructurando las operaciones en métodos bien definidos.

Entregables:

- Código fuente en Java, modularizado y separado por clases.
- Diagrama de clases UML, mostrando relaciones entre las entidades.
- Informe técnico, con:
    - Descripción de la lógica implementada.
    - Justificación del diseño orientado a objetos.
    - Ejemplos de uso del sistema con capturas de pantalla.
- Documentación Javadoc del sistema, al menos en las clases y métodos principales.