**Elaborado por: ING. Lucián Rodríguez Plasencia**

**Ejercicio: Sistema de gestión académica por consola (Java)**

**Contexto / Descripción del problema**

Vas a implementar un sistema académico simplificado para una facultad que gestiona **estudiantes**, **profesores**, **cursos** y **matriculaciones**. El sistema debe permitir crear entidades, inscribir y retirar estudiantes de cursos, registrar calificaciones por evaluación (por curso cada estudiante tiene un arreglo de notas), y generar varios reportes estadísticos.  
**Todas** las estructuras que contienen colecciones deben implementarse con **arreglos básicos** (por ejemplo Estudiante[], Curso[], double[]), y si se necesita más espacio, implementar manualmente la lógica de redimensionamiento (p. ej. duplicar capacidad).

**Clases sugeridas**

* Main — clase con main(String[] args) y menú por consola. Gestiona el flujo y llama a Sistema.
* Sistema — clase «controladora» que contiene los arreglos principales: Estudiante[] estudiantes, Profesor[] profesores, Curso[] cursos, Matricula[] matriculas y métodos para CRUD y operaciones solicitadas.
* Estudiante — atributos: int id, String nombre, int edad, String carrera, int[] cursosInscritos (o int[] con ids de cursos), etc.
* Profesor — atributos: int id, String nombre, String departamento, int[] cursosDictados (ids).
* Curso — atributos: int id, String nombre, int profesorId, int capacidad, int[] estudiantesIds, double[][] notas — **notas por estudiante por evaluación** (ver inciso).
* Matricula — registro de inscripción: int id, int estudianteId, int cursoId, boolean activo.
* Utilidades (o ArrayUtils) — métodos estáticos reutilizables para operar arreglos: buscarPorId, eliminarYCompacar, insertarConResize, ordenarPorBurbuja, etc.
* Reporte (opcional) — métodos para generar y mostrar reportes por consola (promedios, top N, distribución de notas).
* Validaciones (opcional) — leer y validar entradas del usuario (enteros, rangos).

Puedes implementar Matricula o integrar la matrícula dentro de Curso y Estudiante. Lo importante es que haya **al menos 7 clases** y que los arreglos sean predominantes en la solución.

**Reglas / restricciones**

* **No** usar GUI. Todo por consola (Scanner).
* **No** usar java.util.ArrayList, LinkedList, HashMap u otras colecciones. **Solo** arreglos (Tipo[]) y operaciones manuales para insertar/eliminar/redimensionar.
* Manejar entradas inválidas de forma robusta.
* Documentar cada clase y los métodos públicos con comentarios breves.
* Proveer un archivo README.md con instrucciones para compilar y ejecutar.
* Entregar archivos .java separados por clase.

**Incisos (mínimo 10 — la mayoría operan sobre arreglos)**

**a) Diseño e implementación de clases (70% del proyecto)**

Diseña e implementa las clases listadas arriba. Cada entidad (Estudiante, Profesor, Curso, Matricula) debe almacenarse en un arreglo dentro de Sistema. Implementa métodos básicos: crear, leerPorId, actualizar, eliminar.

**Objetivo de arreglo:** por ejemplo Estudiante[] estudiantes con capacidad inicial (ej. 10). Al excederla, duplicar tamaño copiando elementos.

**b) Alta masiva desde consola (uso intensivo de arrays)**

Implementa en Main una opción para **cargar N estudiantes** y **M cursos** desde la consola en la misma ejecución (pedir N, luego N filas con datos).

* Almacenar los datos en arreglos (Estudiante[], Curso[]).
* Validar duplicados por id usando búsqueda lineal en el arreglo.

**c) Inscripción y baja de estudiantes en cursos (manipulación de arrays)**

* Implementa inscribirEstudiante(int estudianteId, int cursoId) que:
* Añada estudianteId al arreglo Curso.estudiantesIds (redimensionando si es necesario).
* Añada cursoId al arreglo Estudiante.cursosInscritos.
* Cree una Matricula en el arreglo Sistema.matriculas.
* Implementa retirarEstudiante(int estudianteId, int cursoId) que elimine el id del arreglo correspondiente y marque la matrícula como inactiva. **Requiere desplazar elementos** al eliminar.

**d) Registro de calificaciones por evaluación (arrays multidimensionales)**

Cada curso tiene **E** evaluaciones (p. ej. 3: parcial, proyecto, final).

* Implementa en Curso un double[][] notas, donde cada fila corresponde a un estudiante inscrito (en el mismo orden que estudiantesIds), y cada columna a una evaluación.
* Al inscribir a un estudiante, debes **insertar una fila** en notas (copiando/reestructurando el arreglo).
* Implementa registrarNota(int cursoId, int estudianteId, int evaluacionIndex, double nota) que ubique el índice del estudiante dentro de estudiantesIds y guarde la nota en notas[idx][evaluacionIndex].

**e) Cálculo de promedios (operaciones sobre arrays)**

* Implementa double promedioEstudiante(int estudianteId) que calcule el promedio ponderado de todas las notas del estudiante a través de sus cursos (recorrer arreglos Curso.notas y Curso.estudiantesIds).
* Implementa double promedioCurso(int cursoId, int evaluacionIndex) que calcule el promedio de una evaluación específica entre todos los estudiantes inscritos.

**f) Estadísticas por curso (uso intensivo de arrays)**

Para cada curso generar:

* Número de inscritos.
* Promedio general (promedio de promedios por estudiante dentro del curso).
* Desviación estándar de las notas finales (calcula nota final por estudiante como promedio de sus evaluaciones).  
  Implementar recorridos y cálculos manuales sobre arreglos (no usar Streams ni librerías estadísticas).

**g) Ordenamiento y top-N (manipulación y ordenación de arrays)**

* Implementa Estudiante[] topNEstudiantesPorCurso(int cursoId, int N) que devuelva los N estudiantes con mayor nota final en ese curso.
* Debes ordenar **solo un arreglo auxiliar** (p. ej. usando un algoritmo simple: burbuja, selección o inserción) que trabaje con arreglos paralelos (ids y promedios) y luego retornar un arreglo de Estudiante con los seleccionados.

**h) Búsqueda y reportes por rango de notas (búsqueda lineal)**

* Implementa una función que, dado cursoId y un rango min,max, devuelva los ids y nombres de estudiantes cuya nota final esté dentro del rango.
* Demostrar su ejecución con ejemplos por consola.

**i) Eliminación lógica y física (arrays + matriculas)**

* Implementa eliminación **lógica** de estudiantes (activo = false), y una rutina de **compactación** que recorra el arreglo Estudiante[] y elimine físicamente las entradas inactivas (copiando a un nuevo arreglo).
* La compactación debe mantener la consistencia: actualizar referencias en Curso.estudiantesIds (si un estudiante fue eliminado, quitar su id y la fila correspondiente en notas).

**j) Merge/concatenación de arreglos y exportación por consola**

* Implementa una función que dado dos cursos genere un **nuevo arreglo** de estudiantes (sin duplicados) que estén en cualquiera de los dos cursos (unión). Trabajar solo con arreglos y búsqueda lineal para eliminar duplicados.
* Exporta ese listado por consola con id, nombre, carrera.

**k) Historial de notas y tendencias (arrays temporales)**

* Para un estudiante dado, construye un arreglo double[] con su **promedio por periodo** (considerar cada curso como un periodo o usar las evaluaciones ordenadas). Implementa una función que calcule si la tendencia es ascendente, descendente o estable (comparando medias consecutivas).

**l) Pruebas automáticas simples en Main (uso y recorrido de arrays)**

* Implementa en Main una opción "Correr pruebas de muestra" que:
* Cree 5 estudiantes, 2 profesores, 3 cursos.
* Inscriba estudiantes en cursos, registre algunas notas y muestre por consola: listados, promedios, top 3 por curso, resultado de merge, y la compactación de estudiantes.
* Esto servirá como demostración del correcto manejo de arreglos.

**Entradas y salidas (interacción por consola — ejemplo breve)**

Menú por consola (ejemplos de opciones):

* Agregar estudiante
* Agregar profesor
* Agregar curso
* Inscribir estudiante en curso
* Retirar estudiante de curso
* Registrar nota
* Mostrar promedio estudiante
* Mostrar estadísticas de curso
* Top N estudiantes por curso
* Compactar estudiantes (eliminar inactivos)
* Pruebas de muestra
* Salir

Ejemplo de interacción para registrar una nota:

> 6 (Registrar nota)

Ingresar cursoId: 101

Ingresar estudianteId: 2001

Ingresar índice de evaluación (0..2): 1

Ingresar nota (0.0 - 10.0): 8.5

Nota registrada correctamente.

**Casos de prueba mínimos a presentar en el README**

* Crear 5 estudiantes (uno con id duplicado — debe rechazarse).
* Crear 2 cursos (capacidad 3). Inscribir 4º estudiante — el sistema debe redimensionar correctamente el arreglo de inscritos.
* Registrar notas (3 evaluaciones). Calcular promedios por curso y por estudiante.
* Mostrar top 2 por curso.
* Retirar un estudiante y ejecutar compactación; verificar que Curso.notas se compactó correctamente.
* Merge entre dos cursos con alumnos comunes (sin duplicados).

**Extensiones opcionales (si quieres más desafío)**

* Persistencia simple a archivo (guardar/leer arreglos desde .csv) usando flujos — **manteniendo** la carga/guardado a arreglos (sin Collections).
* Implementar ordenamiento por merge sort (con arreglos) para grandes volúmenes.
* Soportar ponderaciones distintas por evaluación (por curso) y calcular notas finales ponderadas.