PROGRAMACIÓN III

Trabajo Práctico - Funcional

OBJETIVO GENERAL

Practicar operaciones intermedias y terminales de los Streams en Java para procesar colecciones de forma declarativa.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Stream	flujo de datos que permite aplicar transformaciones (map, filter, sorted).
Collectors	permiten agrupar, contar, promediar, unir en cadenas, etc.
Expresiones Lambda + Streams	permiten procesar colecciones con mínima complejidad.
Operaciones terminales	útiles para agrupar productos, calcular totales y generar reportes.
Inmutabilidad	clave para evitar errores en sistemas de datos.

1. Caso Práctico

Dada la clase `Alumno(nombre, nota, curso)`:

1. Obtener los nombres de los alumnos aprobados (nota ≥ 7) en mayúsculas y ordenados.
2. Calcular el promedio general de notas.
3. Agrupar alumnos por curso usando `Collectors.groupingBy()`.
4. Obtener los 3 mejores promedios.

CONCLUSIONES ESPERADAS

- Entender el pipeline de Streams.
- Usar operaciones de mapeo, filtrado, reducción y agrupación.
- Reemplazar código imperativo por un enfoque declarativo más claro.

2. Caso Práctico

Dada la clase `Producto(nombre, categoria, precio, stock)`:

1. Listar los productos con precio mayor a 100, ordenados por precio descendente.
2. Agrupar productos por categoría y calcular el stock total.
3. Generar un String separando con ";" cada producto que contenga nombre y precio, usando `map` + `collect(joining)`.
4. Calcular el precio promedio general y por categoría.

CONCLUSIONES ESPERADAS

- Aplicar programación funcional en un contexto real.
- Generar reportes y estadísticas con Streams.
- Consolidar lambdas, collectors y manipulación funcional de listas.

3. Caso Práctico

Dada la clase Libro(titulo, autor, paginas, precio):

1. Listar los títulos de los libros con más de 300 páginas, ordenados alfabéticamente.
2. Calcular el promedio de páginas de todos los libros.
3. Agrupar los libros por autor y contar cuántos libros tiene cada uno.
4. Obtener el libro más caro de la lista.

CONCLUSIONES ESPERADAS

- Practicar operaciones de filter, map, sorted y collect.
- Manejar promedios y máximos con Streams.
- Entender la agrupación con Collectors.groupingBy() y counting().

4. Caso Práctico

Dada la clase Empleado(nombre, departamento, salario, edad):

1. Obtener la lista de empleados cuyo salario sea mayor a 2000, ordenados por salario descendente.
2. Calcular el salario promedio general.
3. Agrupar los empleados por departamento y calcular la suma de salarios de cada uno.
4. Obtener los nombres de los 2 empleados más jóvenes.

CONCLUSIONES ESPERADAS

- Aplicar operaciones de filtro, ordenamiento y límite.
- Calcular promedios y sumatorias con collectors.
- Practicar groupingBy con downstream collectors (summingDouble).