Alexis Tarifa Pérez u232817
Arnau Carbonell Carrasco u214562

LAB 1

El objetivo principal de este laboratorio fue implementar en Java el diseño presentado en el Seminar 1, que consistía en desarrollar las clases Vector y Record, además de una clase de prueba denominada TestRecord destinada a comprobar el correcto funcionamiento de las anteriores. El propósito del programa es representar y manipular vectores matemáticos utilizando los principios fundamentales de la programación orientada a objetos.

La clase Vector fue diseñada para modelar un vector cuyos elementos se almacenan en un array de tipo double. Esta clase permite realizar diferentes operaciones matemáticas entre vectores o con escalares, tales como suma, resta, multiplicación y división elemento a elemento, así como multiplicación y división por un escalar. También se implementaron métodos para calcular la raíz cuadrada de cada componente, el producto escalar y la norma del vector. Además, se incluyeron dos constructores: uno que recibe un array ya existente y otro que crea un vector de una dimensión dada, inicializando todos sus elementos con un valor específico. Finalmente, el método toString se redefinió para permitir una representación textual clara del vector.

Por otro lado, la clase Record tiene como función almacenar un par compuesto por un vector de entrada (input) y un valor de salida (output). Esta clase incluye un constructor, dos métodos getter para acceder a sus atributos y un método toString que devuelve una representación legible del registro.

Otra posibilidad discutida fue la implementación de una versión más compacta del código utilizando funciones lambda, tal como se sugiere en la parte opcional del enunciado. Esta opción se descartó por considerarse innecesariamente compleja para los objetivos del laboratorio, que estaban centrados en la correcta aplicación de los conceptos básicos de la programación orientada a objetos.

En cuanto al manejo de la división, tanto en las operaciones vectoriales como en las operaciones escalares, en un principio se decidió devolver el valor 0 en los casos en que el divisor fuera igual a cero. Esta solución permitiría evitar errores de ejecución y asegurar que el programa continuara funcionando correctamente sin interrupciones. Sin embargo, tras consultar con el profesor, se nos recomendó utilizar los siguientes mensajes de error y finalizar la ejecución del programa con System.exit(0) cuando las dimensiones de los vectores no coinciden o cuando ocurre una división no válida:

```
System.out.println("Los vectores son de diferentes dimensiones");

System.exit(0);
```

Alexis Tarifa Pérez u232817
Arnau Carbonell Carrasco u214562

Durante la implementación del método toString, inicialmente se escribió una versión manual basada en un bucle for que recorría los elementos del array y los concatenaba en una cadena con formato. Sin embargo, tras la explicación del profesor en clase, se reemplazó por la llamada a Arrays.toString(elems), una forma más sencilla, legible y eficiente de obtener la representación textual del vector.

Código versión anterior:

```java
public String toString() {
    String s = "[";
    for (int i = 0; i < elems.length; i++) {
        s += elems[i];
        if (i < elems.length - 1) {
            s += ", ";
        }
    }
    s += "]";
    return s;
}
```

En cuanto a los conceptos teóricos aplicados, el desarrollo del laboratorio permitió reforzar principios esenciales de la programación orientada a objetos. Se aplicó encapsulación, manteniendo los atributos como privados y accediendo a ellos únicamente mediante métodos públicos. Se utilizaron constructores para ofrecer distintas formas de inicializar los objetos, y se aplicó composición, dado que la clase Record contiene un objeto de tipo Vector.

Tras realizar diferentes pruebas en la clase TestRecord, se comprobó que las operaciones aritméticas entre vectores y las operaciones con escalares funcionaban correctamente. Las salidas producidas coincidían con los resultados esperados, lo que demostró que los métodos fueron implementados de manera adecuada.

En conclusión, el trabajo permitió aplicar de manera práctica los fundamentos de la programación orientada a objetos, como la encapsulación, la composición y la reutilización de código. Las clases Vector y Record funcionan de acuerdo con las especificaciones del laboratorio, y el conjunto del programa ofrece una base sólida para extender el proyecto en futuras prácticas. El proceso de implementación y prueba ayudó a afianzar los conocimientos teóricos y a comprender mejor la importancia de un diseño estructurado y modular en la programación orientada a objetos.