Guía N° 1 – Introducción a la Estructura de Datos

1. OBJETIVOS ESPECÍFICOS DE LA PRÁCTICA DE LABORATORIO

En este problema, se pide implementar un tipo abstracto de datos que represente una matriz de enteros y sus principales operaciones (suma, traspuesta y multiplicación). También se pide implementar un TAD que represente un polinomio y sus diferentes operaciones.

2. PROBLEMA 1

Implementa una clase, Matrix, que representa una matriz matemática de enteros. La clase debe proporcionar métodos para crear matrices, manejarlas aritméticamente y algebraicamente, y determinar sus propiedades matemáticas (traza, rango, inversa, determinante).
¿Cómo se almacena una matriz MxN en la memoria? ¿Qué atributos se necesitan y de qué tipos son (estáticos o no estáticos)?

Se pide:

1) Añadir un método constructor para crear una matriz de ceros.

2) Añadir un método para inicializar los valores de una matriz con valores aleatorios entre 0 y 10.

3) Añadir un método no estático, show, para mostrar los valores de la matriz invocada.

4) Añadir un método no estático, transpose, para crear y devolver la transpuesta de la matriz invocada.

5) Añadir un método no estático, plus, que toma una matriz como parámetro de entrada, y devolver una nueva matriz que sea la suma de la matriz invocada y la matriz pasada como parámetro. Si las matrices no se pueden sumar, el método debería mostrar un mensaje de error y devolver null.

6) Añadir un método no estático, minus, que toma una matriz como parámetro de entrada, y devolver una nueva matriz que es el resultado de la diferencia entre la matriz invocada y la matriz pasada como parámetro. Si las matrices no se pueden restar, el método debería mostrar un mensaje de error y devolver null.

7) Añadir un método no estático, equal, que toma una matriz como parámetro de entrada, y verifica si la matriz invocada es igual a la matriz pasada como parámetro.

8) Añadir un método estático que cree y devuelva la matriz identidad NxN.

3. PROBLEMA 2

Sea un polinomio:

$$Q(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$$

La siguiente interfaz muestra la especificación formal del tipo de datos abstractos (TAD) de un polinomio

```
public interface iPolynomial {
    public int getDegree();
    public int getCoeficient(int n);
    public void setCoeficient(int n, int newValue);
    public int getValue(int x);
    public iPolynomial suma(iPolynomial p);
}
```

Crea una clase, Polynomial, que represente el TAD del polinomio. Se pueden seguirlos siguientes pasos:

a) Crea los atributos que consideres necesarios para representar el polinomio y escribe un método constructor que permita inicializar los coeficientes del polinomio (dichos valores se reciben como argumento del constructor)

b) Implementa el método getDegree(), que devuelve el grado del polinomio

Por ejemplo, $Q(x)=5$ tiene grado 0. $Q(x)=x^2+5$ tiene grado 2.

c) Implementa los método getCoefficient(int n) (devuelve el coeficiente del término de grado n) y el método setCoefficient(int n, int newValue) (modificael coeficiente del término de grado n).

Por ejemplo, dado el polinomio $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$, getCoefficient(3) returns a_3.

Por ejemplo, dado el polinomio $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+\ldots+a_nx^n$ *, this call setCoefficient(3,b) does that now Q(x) is* $a_0+a_1x+a_2x^2+bx^3+\ldots+a_nx^n$

d) Implementa el método CalculateValue(int x), que recibe un valor entero y calcular el valor del polinomio para dicho valor.

Por ejemplo, $Q(3)=a_0+a_13+a_29+a_327+\ldots+a_n3^n$

e) Implementa el método sum(IPolynomial p) que devuelve la suma del polinomio objeto y el polinomio p.

Por ejemplo, $Q(x)=3x^2+4x+5$, $p(x)=x^3+4x^2-2x-3$
Q.sum(p) $\rightarrow x^3+7x^2+2x+2$

Nota: Puedes utilizar el método Math.power(a,b) para calcular el valor de un número a elevado a una potencia b. También puedes usar la clase Math.max(x,y) para calcular el máximo de dos números x e y.