Diseñando e implementando una matriz

Supongamos que queremos modelar una clase denominada Matriz. Queremos poder realizar las distintas operaciones típicas de una matriz:

- Sumar matrices
- Restar matrices
- Multiplicar matrices
- Trasponer una matriz.

En todos estos procesos está implicada al menos una matriz (estructura de filas y columnas).

TODO SON OBJETOS

Con la filosofía de la programación orientada a objetos (POO) tengo que analizar el estado de una matriz y su comportamiento.

ESTADO DE LA MATRIZ

Los elementos de esa matriz serán el estado de la matriz. Supongamos que queremos una matriz de enteros. Un atributo de la matriz sería entonces un array de enteros, por ejemplo…

```
public class Matriz {
    private int[][] matriz;
```

CONSTRUCTORES DE LA MATRIZ

En este caso sólo se van a permitir dos constructores.

Un primer constructor inicializará todos los valores a 0. Se invocará así:

```
Matriz matriz0 = new Matriz(2, 3);
matriz0.mostrar("Matriz 0");
```

Se quiere crear una matriz de 2 x 3. Sus valores se dejarán a 0.

```
Matriz 0

        0       0       0
        0       0       0
```

Y se implementa así:

```java
/**
 * Crea una matriz con todos sus valores a 0
 *
 * @param filas
 *            número de filas
 * @param columnas
 *            número de columnas
 */
public Matriz(int filas, int columnas) {
    matriz = new int[filas][columnas];
}
```

Otro constructor inicializará con valores aleatorios comprendidos entre un mínimo y un máximo.

```java
Matriz matriz1 = new Matriz(2, 3, 0, 5);
matriz1.mostrar("Matriz 1");
```

En este caso, se crea una matriz de 2x3 con valores comprendidos entre 0 y 5, ambos incluidos.

```
Matriz 1


        2       4       0
        3       2       5
```

```java
/**
 * Crea una matriz con valores aleatorios
 *
 * @param filas
 *            número de filas
 * @param columnas
 *            número de columnas
 * @param min
 *            valor mínimo aleatorio
 * @param max
 *            valor máximo aleatorio
 */
public Matriz(int filas, int columnas, int min, int max) {
    matriz = new int[filas][columnas];
    for (int i = 0; i < filas; i++)
        for (int j = 0; j < columnas; j++)
            matriz[i][j] = (int) (Math.random() * (max - min + 1)) + min;
}
```

COMPORTAMIENTO DE LA MATRIZ

Su comportamiento ya está descrito previamente, y se traduce en los siguientes métodos

```
    public void mostrar(String string) {
     * Muestra el estado de la matriz
    public void mostrar() {
     * Suma dos matrices
    public Matriz sumar(Matriz s2) {
     * Resta dos matrices
    public Matriz restar(Matriz sustraendo) {
     * Traspone la matriz
    public void trasponer() {
     * Multiplica dos matrices
    public Matriz multiplicar(Matriz p2) {
     * comprueba las dimensiones de dos matrices
    private boolean mismasDimensiones(Matriz m) {
     * Averigua la primera dimensión de la matriz
    private int filas() {
     * Averigua la segunda dimensión de la matriz
    private int columnas() {
```

TRASPONER EL OBJETO MATRIZ

Cuando al objeto matriz se le pase el mensaje trasponer, cambiará el número de filas por columnas, así como sus elementos.

La implementación del método puede ser la siguiente:

```
public void trasponer() {
    int filas = filas();
    int columnas = columnas();

    int[][] resultado = new int[columnas][filas];
    for (int i = 0; i < filas; i++)
        for (int j = 0; j < columnas; j++)
            resultado[j][i] = matriz[i][j];
    matriz = resultado; // se actualiza el estado del objeto
}
```

En ella modifico el estado de la matriz. Creo un nuevo objeto y machaco el anterior:

```
    matriz = resultado;
```

Como estoy modificando el estado del propio objeto, el método no precisa de parámetros ni devuelve nada.

La invocación del método (que es totalmente accesible porque es public) sería algo así:

```
    matriz1.trasponer();
```

Y el resultado:

```
Matriz 1


        3       0       1
        3       4       5
Matriz 1 traspuesta


        3       3
        0       4
        1       5
```

SUMAR DOS MATRICES

El método sumar necesita dos sumandos, que serán dos objetos matrices. El resultado se almacenará en una tercera matriz.

```
Sumando 1


        0       5       4
        1       2       2
Sumando 2


        9       7       8
        6       8       8
Resultado


        9       12      12
        7       10      10
```

Una suma implica tres objetos:

```
Matriz matriz3 = matriz1.sumar(matriz2);
```

- Sumando 1 (matriz1). Para realizar esta suma al objeto matriz 1 se le pasará el mensaje sumar. Será el sumando 1.
- Sumando 2 (matriz2). Como necesita un segundo sumando, se le pasará una nueva instancia de Matriz como argumento.
- Resultado (matriz3). El resultado será un nuevo objeto que se creará dentro del método sumar. Será devuelto.

Y su implementación:

```
public Matriz sumar(Matriz s2) {
    if (!mismasDimensiones(s2)) {
        System.out
            .println("\nNo puedo sumar las matrices porque no son de las mismas dimensiones");
        return null; //se sale del método porque no se pueden sumar
    }

    int filas = filas();
    int columnas = columnas();

    Matriz suma = new Matriz(filas, columnas); // se crea un nuevo objeto para el
                                               // resultado
    for (int i = 0; i < filas; i++)
        for (int j = 0; j < columnas; j++)
            suma.matriz[i][j] = matriz[i][j] + s2.matriz[i][j];
    return suma;// se devuelve el nuevo objeto
}
```

FLECOS QUE RECORTAR

En esta implementación hay detalles que perfilar:

- En los constructores, hay que comprobar que las dimensiones sean válidas. Un valor 0 ó negativo daría error.
- En el constructor con números aleatorios, hay que comprobar que el máximo sea mayor que el mínimo.
- Implementar el resto de métodos.

TAREAS

Completa las siguientes frases con estos términos:

- identificador
- estado
- atributos (campos)
- constructor
- POO
- comportamiento
- Inicializar
- constructor por defecto
- referencia a un objeto
- variables miembro
- automáticamente
- instanciación
- métodos o mensajes
- instancia

1. Los objetos son entidades que tienen un determinado_____, _____ e identidad:
2. El estado de un objeto está compuesto de datos o informaciones; serán los _____a los que se habrán asignado unos valores concretos. También se les denomina _____. La idea es que un atributo representa una propiedad determinada de un objeto.
3. El comportamiento está definido por los _____a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
4. La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su _____.

5. La _____difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.
6. La _____es la creación de un objeto a partir de una clase.
7. El objetivo del _____es el de _____un Objeto cuando éste es creado. Asignaremos los valores iniciales así como los procesos que ésta clase deba realizar.
8. Un _____es un constructor sin parámetros que no hace nada. Sin embargo será invocado cada vez que se construya un objeto sin especificar ningún argumento, en cuyo caso el objeto será inicializado con los valores predeterminados por el sistema (los atributos de datos primitivos a un valor predeterminado las referencias a objetos a null).
9. El constructor por defecto se crea _____siempre que el programador no diseñe ningún otro constructor.
10. Un objeto es una _____de una clase predefinida en Java o declarada por el usuario y referenciada por una variable que almacena su dirección de memoria. A esa variable se le denomina _____.