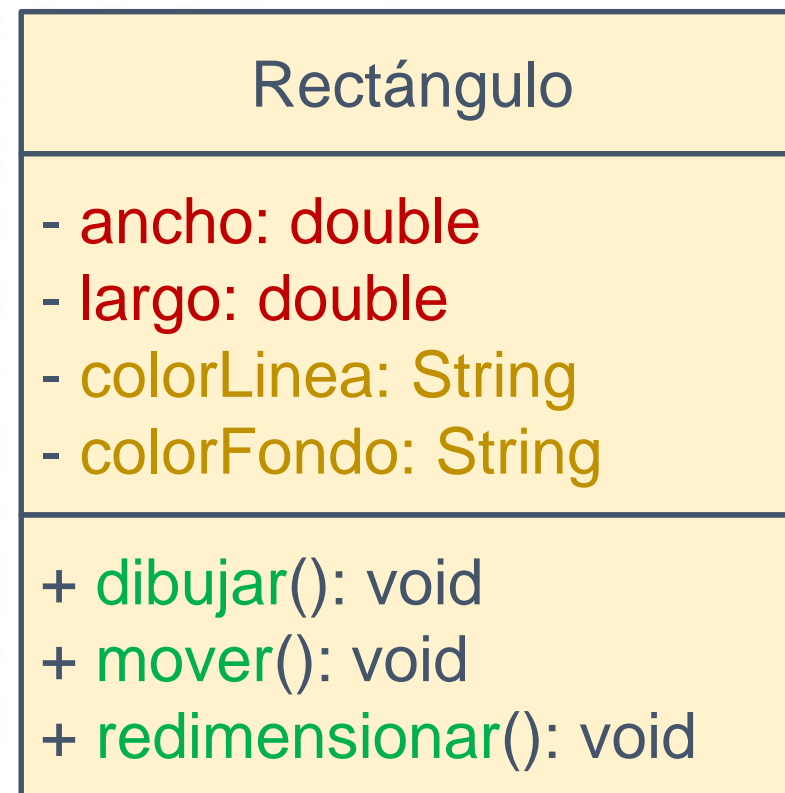


Clase y Objeto

Clase



Objeto

rec1

largo = 4.98
ancho = 2.18

```
public class Rectangulo {  
    private double ancho;  
    private double largo;  
    private String colorLinea;  
    private String colorFondo;  
  
    public void dibujar() {}  
    public void mover() {}  
    public void redimensionar() {}  
}
```

```
Rectangulo rec1 = new Rectangulo();  
rec1.setLargo(4.98);  
rec1.setAncho(2.18);  
rec1.setColorFondo("celeste");  
rec1.setColorLinea("azul");
```

Getters & Setters

```
public class Alumno {  
    ● private String nombre;  
    ● private int edad;  
    ● private int PC1;  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public int getEdad() {  
        return edad;  
    }  
  
    public int getPC1() {  
        return PC1;  
    }  
}
```

getters

```
public class Alumno {  
    ● private String nombre;  
    ● private int edad;  
    ● private int PC1;  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public void setEdad(int edad) {  
        this.edad = edad;  
    }  
  
    public void setPC1(int PC1) {  
        this.PC1 = PC1;  
    }  
}
```

setters

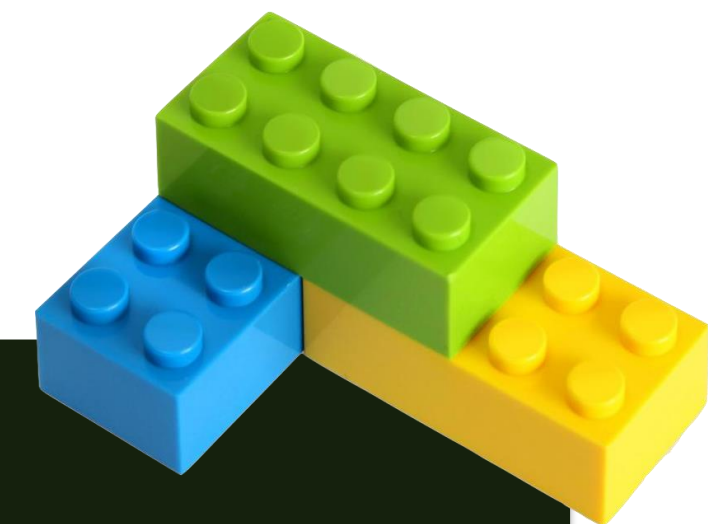
Constructores

```
public class Box {  
    private double width;  
    private double height;  
    private double depth;  
  
    public Box() {  
        width = 10;  
        height = 10;  
        depth = 10;  
    }  
  
    public Box(double width, double height,  
               double depth) {  
        this.width = width;  
        this.height = height;  
        this.depth = depth;  
    }  
}
```

Tipos de constructores:

- **Vacío** o por defecto.
- Parametrizado **completo**.
- Parametrizado **sobrecargado**.

```
public class Test {  
    public static void main(String[] args) {  
        Box myBox = new Box();  
        System.out.printf("Volume = %9.2f cm3\n",  
                           myBox.width * myBox.height * myBox.depth);  
  
        Box yourBox = new Box(12.5, 13.7, 10.4);  
        System.out.printf("Volume = %9.2f cm3",  
                           yourBox.width * yourBox.height * yourBox.depth);  
    }  
}
```



Procedimientos y funciones

```
public class OperadorArreglos {
    private int[] datos;

    public OperadorArreglos(int[] datos) {
        this.datos = new int[datos.length];
        System.arraycopy(datos, 0, this.datos, 0,
            datos.length);
    }

    public void calcularPromedio() {
        double suma = 0;
        for (int dato : datos) {
            suma += dato;
        }

        System.out.printf("Promedio: %.2f\n",
            suma / datos.length);
    }
}
```

procedimiento

```
public class OperadorArreglos {
    private int[] datos;

    public OperadorArreglos(int[] datos) {
        this.datos = new int[datos.length];
        System.arraycopy(datos, 0, this.datos, 0,
            datos.length);
    }

    public int obtenerMaximo() {
        int maximo = datos[0];
        for (int i = 1; i < datos.length ; i++) {
            if (maximo < datos[i]) {
                maximo = datos[i];
            }
        }
        return maximo;
    }
}
```

función

Sobreescritura y Sobrecarga

```
public class Alumno {  
    private String foto;  
    private String nombre;  
    private int edad;  
    private int pc1, pc2, pc3, ef;  
  
    // getters, setters, métodos  
  
    @Override  
    public String toString() {  
        return nombre + " - " + edad;  
    }  
}
```

sobrescritura

```
public class Sumador {  
    static int suma(int a, int b) {  
        return a + b;  
    }  
    static double suma(double a, double b) {  
        return a + b;  
    }  
    static double suma(double a, double b, double c) {  
        return a + b + c;  
    }  
    static double suma(double[] numeros) {  
        double suma = 0;  
        for(double numero : numeros) {  
            suma += numero;  
        }  
        return suma;  
    }  
}
```

sobrecarga

Práctica

Caso Resuelto. Evaluador Aritmético

- Crear la clase **EvaluadorAritmetico** que reciba como argumento de su método **main()** una expresión aritmética combinada simple (sumas y restas) y retorne el resultado de su procesamiento.

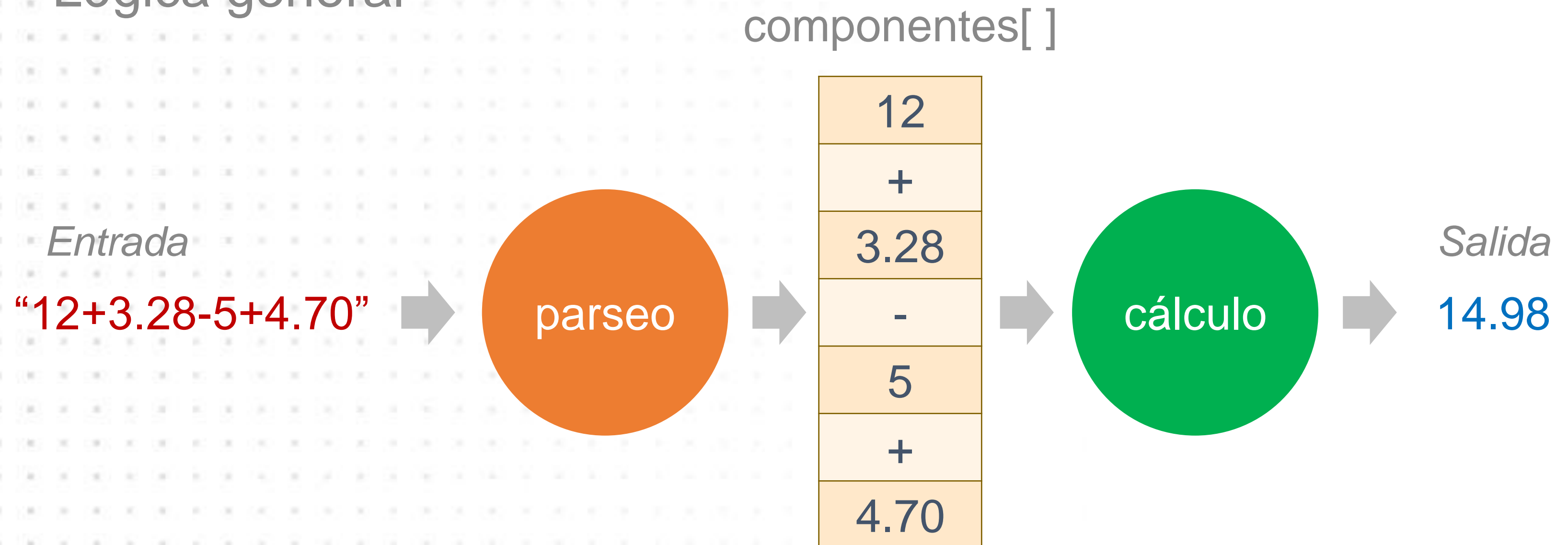
Ejemplo:

Si se ingresa "**12+3.28-5+4.70**" se deberá devolver como resultado **14.98**



Caso Resuelto. Evaluador Aritmético

- Lógica general



Caso Resuelto. Evaluador Aritmético

EvaluadorAritmetico	
-EXPRESION_REGULAR: String	
-componentes: String[]	
+evaluarExpresion(expresion: String): double	
-parsearExpresion(expresion: String): void	
-obtenerNumeroDeComponentes(expresion: String): int	
-procesarExpresion(): double	

Expresión regular para evaluar la expresión aritmética.

Arreglo que almacenará los operandos y operadores.

Función que retornará el resultado de la evaluación de la expresión.

Procedimiento que identifica los componentes y los distribuye en un arreglo.

Función que usa el arreglo **componentes** para calcular y devolver el resultado de la expresión.

Función que devuelve el número de componentes de una expresión aritmética.

Caso Resuelto. Evaluador Aritmético

La clase EvaluadorAritmetico

```
public class EvaluadorAritmetico {  
  
    // expresión regular para evaluar la expresión aritmética  
    private static final String EXPRESION_REGULAR = "(\\d+\\.\\d+|\\d+)|([+\\-])?";  
  
    // arreglo que contendrá los componentes de la expresión aritmética;  
    // esto es, operandos y operadores  
    private static String[] componentes = null;  
}
```

Probamos previamente la expresión regular utilizando <https://regex101.com/>

Caso Resuelto. Evaluador Aritmético

```
/**
 * Método que calcula el número de componentes de una expresión aritmética
 * El número de componentes es la suma de la cantidad de operandos +
 * la cantidad de operadores.
 * En general, si hay N operadores, habrá N + 1 operandos
 * @param expresion expresión aritmética a evaluar
 * @return número de componentes de la expresión
 */
private static int obtenerNumeroDeComponentes(String expresion) {
    int contador = 0;

    for(char c : expresion.toCharArray()) {
        if (c == '+' || c == '-') contador++;
    }

    return contador * 2 + 1;
}
```

"12+3.28-5+4.70"

Contamos el número de
operadores en la expresión

"12+3.28-5+4.70"

En general, si hay N
operadores, entonces habrá
N + 1 operandos.

Por tanto, el número de
componentes es igual a:
 $N + N + 1 \gg 2N + 1$

Caso Resuelto. Evaluador Aritmético

```
/**
 * Método que desagrega la expresión aritmética recibida como una
 * cadena en sus componentes (operandos y operadores)
 * @param expresion
 */
private static void parsearExpresion(String expresion) {
    componentes = new String[obtenerNumeroDeComponentes(expresion)];

    Pattern patronExpresion = Pattern.compile(EXPRESION_REGULAR);
    Matcher coincidencias = patronExpresion.matcher(expresion);

    int indice = 0;

    while (coincidencias.find()) {
        if (!coincidencias.group().isEmpty())
            componentes[indice++] = coincidencias.group();
    }
}
```

"12+3.28-5+4.70"



Pattern &
Matcher



componentes[]

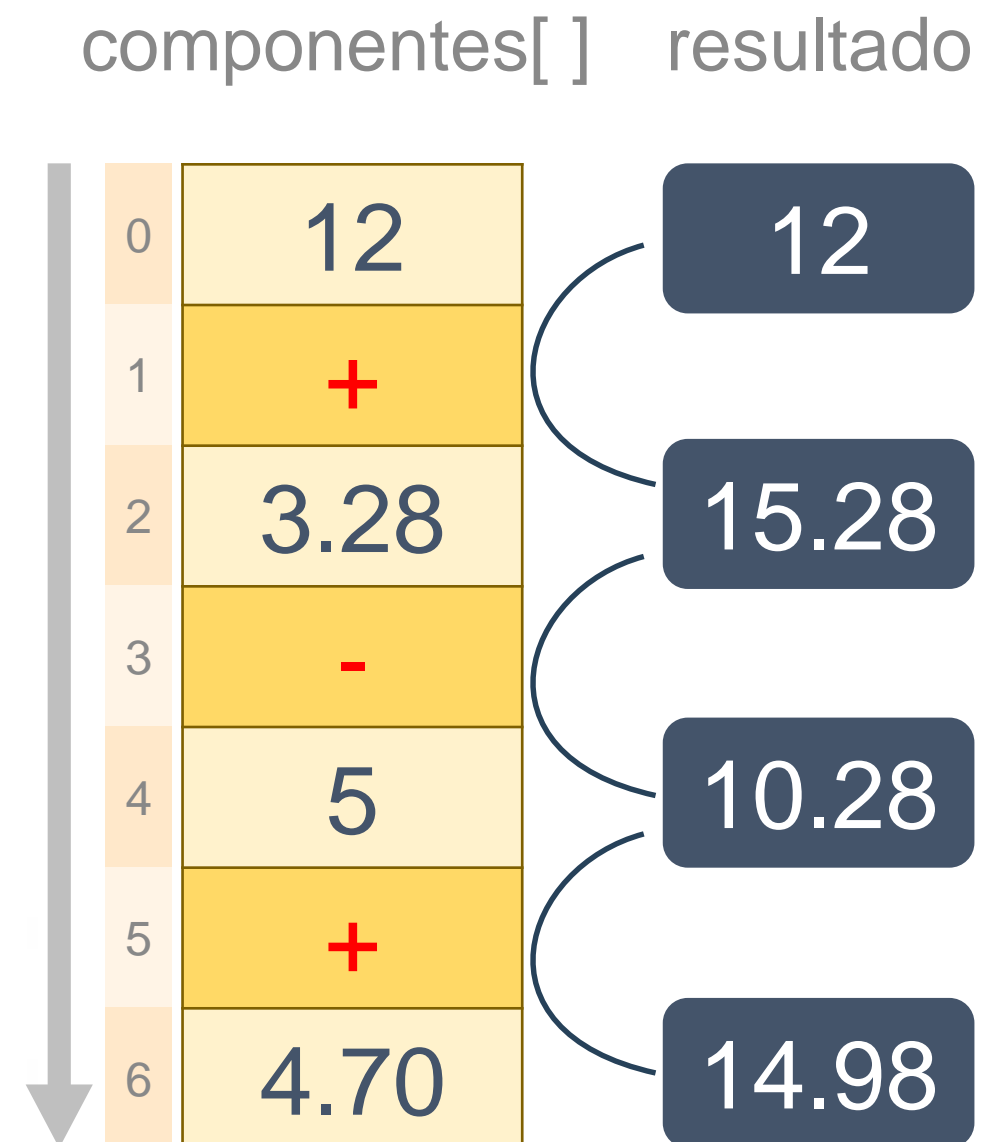
12
+
3.28
-
5
+
4.70

Caso Resuelto. Evaluador Aritmético

```
/**
 * Método que hace un recorrido por el arreglo de componentes de la
 * expresión y realiza el cálculo, teniendo en cuenta el operador en
 * cada caso.
 * @return resultado final del cálculo de la expresión.
 */
private static double procesarExpresion() {
    double resultado = Double.parseDouble(componentes[0]);

    for (int i = 1; i < componentes.length; i += 2) {
        resultado = componentes[i].charAt(0) == '+' ?
            resultado + Double.parseDouble(componentes[i + 1]) :
            resultado - Double.parseDouble(componentes[i + 1]);
    }

    return resultado;
}
```



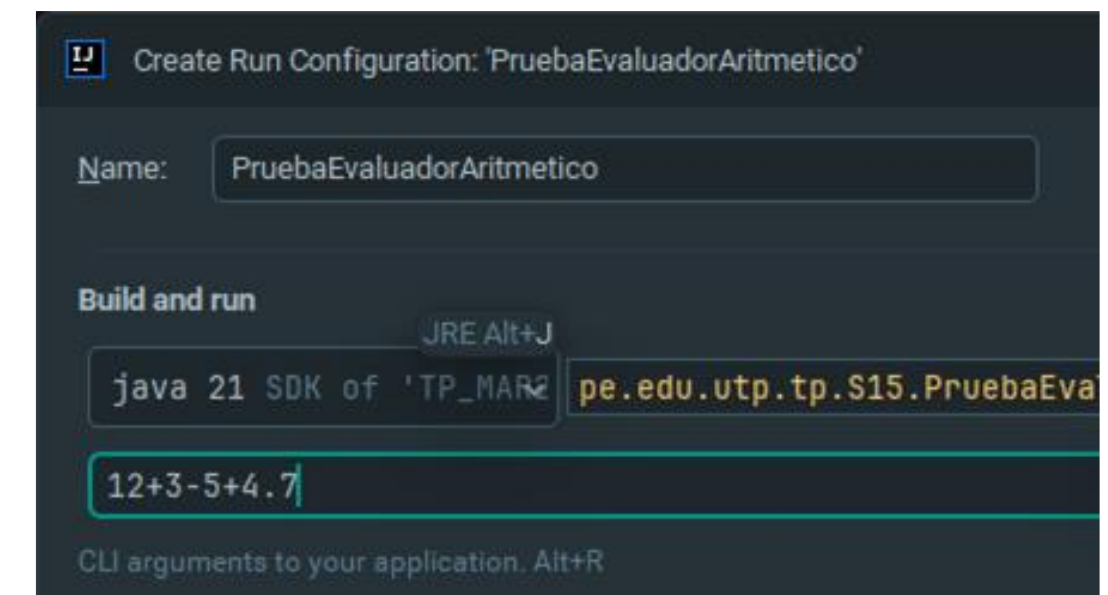
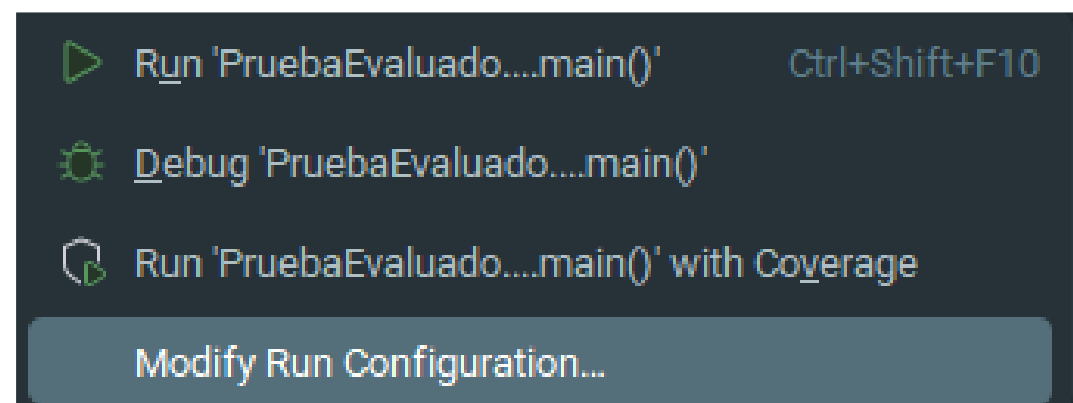
Caso Resuelto. Evaluador Aritmético

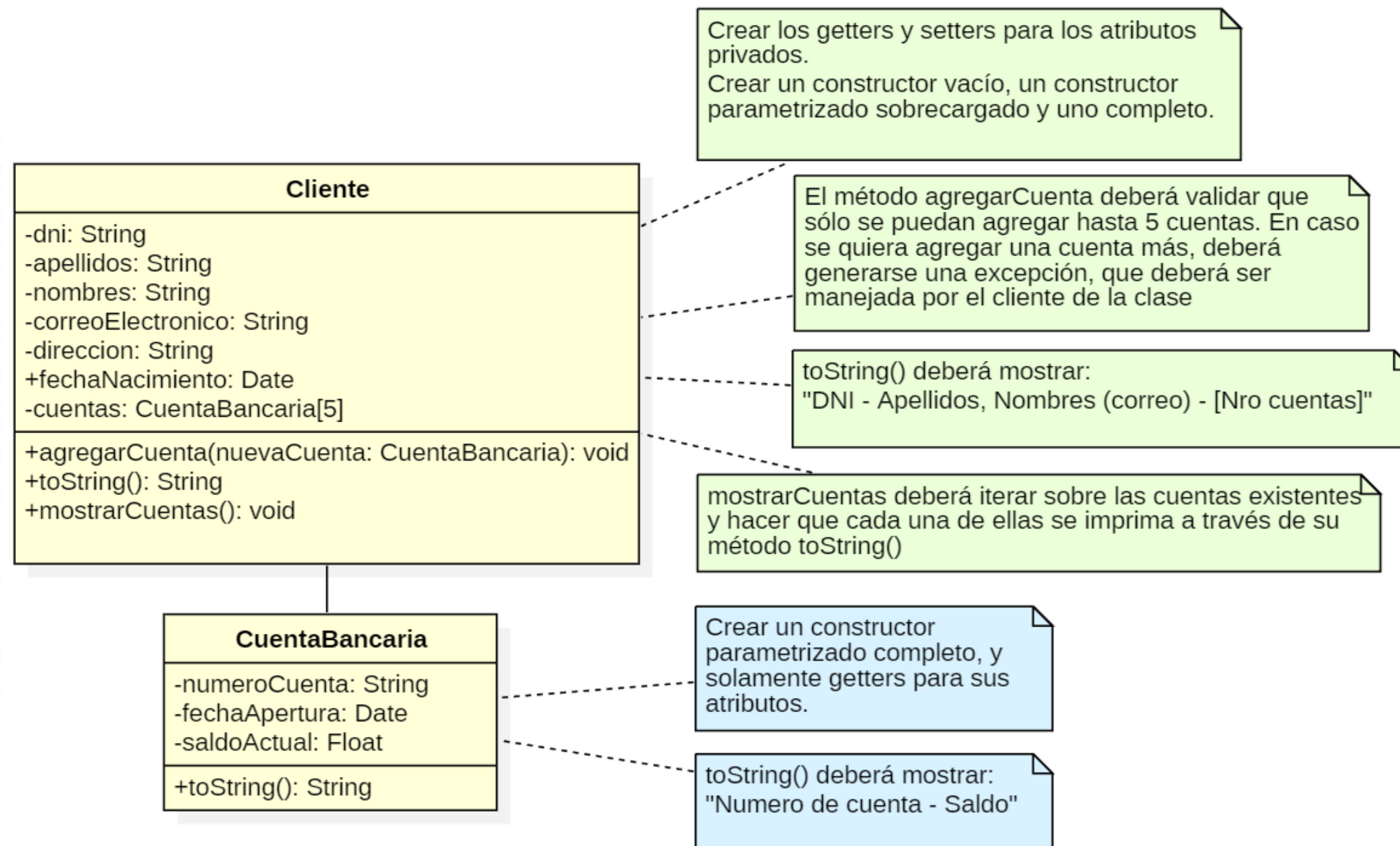
Finalmente, ponemos a prueba la clase implementada

```
public class PruebaEvaluadorAritmetico {  
    public static void main(String[] args) {  
  
        double resultado =  
            EvaluadorAritmetico.evaluarExpresion(args[0]);  
        System.out.printf("Resultado: %f", resultado);  
  
    }  
}
```

Desde el IDE podemos establecer el argumento del método **main()**.

Ejecuta luego la aplicación.





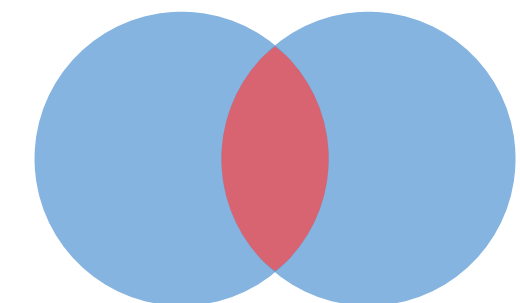
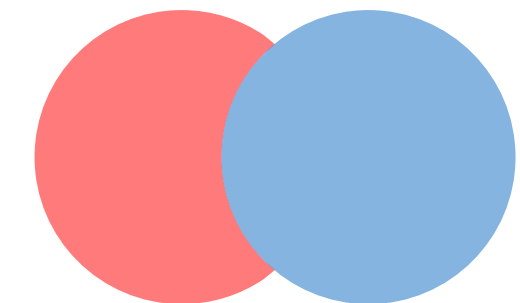
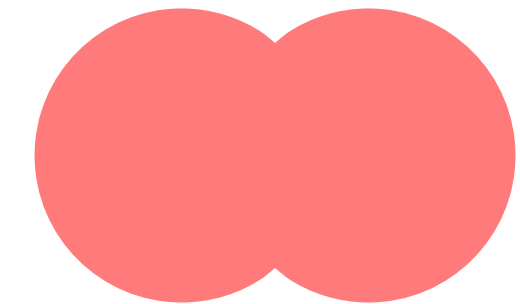
Ejercicio 1

Implementar en Java las clases mostradas en el diagrama UML.



Ejercicio 2

- La clase **OperadorArreglos** tiene 3 métodos que realizan operaciones sobre vectores enteros:
 - a) **sumarConjuntos(int[] arr1, int[] arr2)**: devuelve un arreglo con los números de arr1 y arr2, sin repetidos.
 - b) **restarConjuntos(int[] arr1, int[] arr2)**: devuelve un arreglo con los números que están en arr1 pero no en arr2, sin repetidos.
 - c) **intersectarConjuntos(int[] arr1, int[] arr2)**: devuelve un arreglo con los números que están en arr1 y arr2 a la vez, sin repetidos.
- La clase **AppPOO**, en su método **main()**, ofrecerá al usuario, a través de la clase **Menu**, un menú con las 3 opciones.
- Todas las peticiones de datos estarán a cargo de la clase **LectorDatos**, desarrolladas anteriormente.



Ejercicio 3

- Implementar el juego de 3 *en raya* utilizando arreglos bidimensionales y POO.

