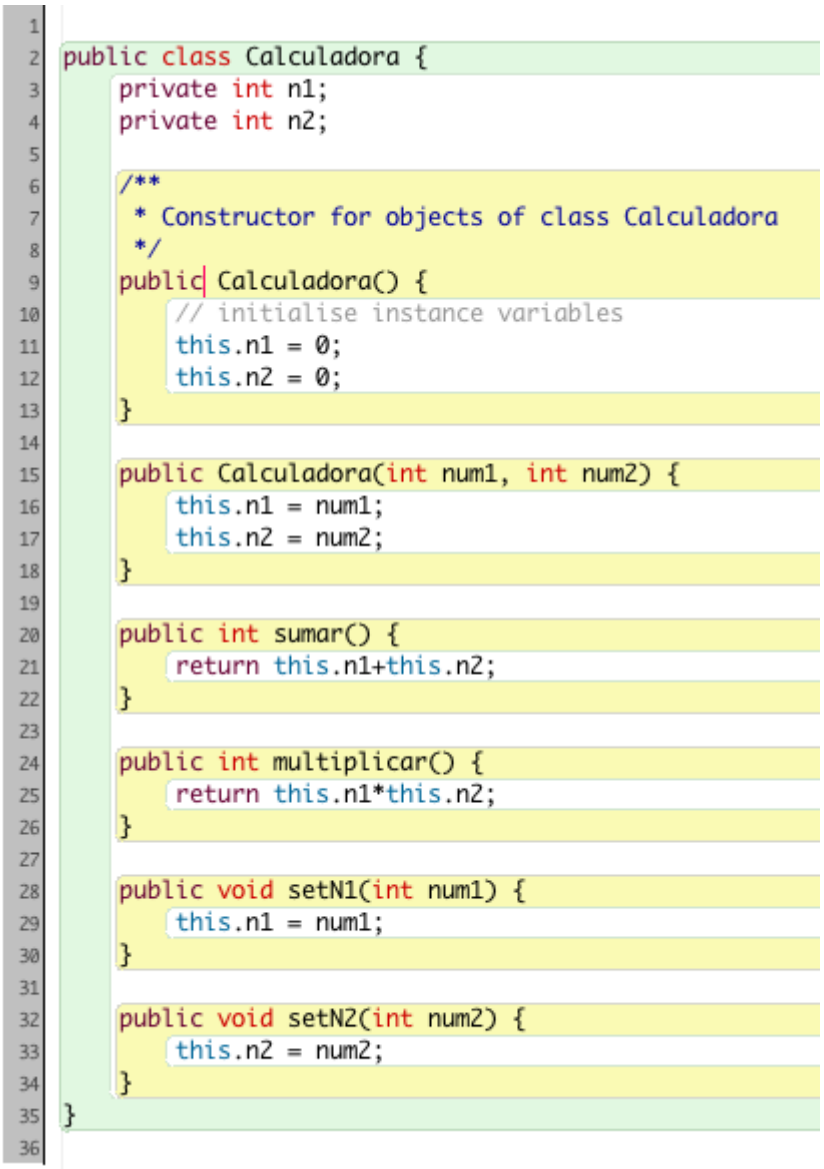
Evidencia 07: Análisis de código y Dependencia

Integrantes: Sebastián Martínez, Joaquín Silva, Gerson Urrea.

Imagenes para analizar



```java

public class Calculadora {
    private int n1;
    private int n2;

    /**
     * Constructor for objects of class Calculadora
     */
    public Calculadora() {
        // initialise instance variables
        this.n1 = 0;
        this.n2 = 0;
    }

    public Calculadora(int num1, int num2) {
        this.n1 = num1;
        this.n2 = num2;
    }

    public int sumar() {
        return this.n1+this.n2;
    }

    public int multiplicar() {
        return this.n1*this.n2;
    }

    public void setN1(int num1) {
        this.n1 = num1;
    }

    public void setN2(int num2) {
        this.n2 = num2;
    }
}

```

```java
public class CarroCompra {
    private int [ ][ ] productos =  new int[2][5];

    /**
     * Constructor for objects of class CarroCompra
     */
    public CarroCompra() {
        // initialise instance variables
        for (int i=0; i<5;i++) {
            productos[0][i] = 1;
            productos[1][i] = 1000;
        }
    }

    private int calcularTotal( ) {
        int total = 0, subtotal=0;

        for (int i=0; i<5;i++) {
            total +=subTotal(productos[0][i],productos[1][i]);
        }
        return total;
    }

    private int subTotal(int cant, int precio) {
        Calculadora calc = new Calculadora(cant, precio);
        return calc.multiplicar();
    }

    public void mostrarTotal() {
        System.out.println("El total de la compra es: " +this.calcularTotal());
    }
}
```

Actividades

1. Identifique las clases y lo que éstas representan. Luego, establezca una descripción textual breve del contexto problema.

   Hay 2 clases publicas, la primera llamada Calculadora que incluye métodos con las operaciones necesarias para que el programa funcione correctamente, la segunda clase llamada CarroCompra que hace uso de un objeto de la clase Calculadora en uno de sus métodos.

   El desafío implica el desarrollo de una calculadora destinada a la compra de productos. Esta calculadora opera mediante la creación de una matriz, en la cual la fila superior almacena la cantidad de productos y la fila inferior guarda los precios unitarios de dichos productos. De esta manera, se permite calcular el costo total multiplicando las cantidades por los precios unitarios y luego sumando estos valores para obtener el resultado, el cual se muestra por consola.

2. Analice los atributos y métodos de cada clase, luego, identifique las relaciones existentes entre las clases identificadas y establezca una descripción textual breve del contexto problema.

   La clase Calculadora posee dos atributos de tipo int, llamados n1 y n2. Esta clase posee dos constructores, el primer constructor inicializa un objeto de la clase Calculadora sin la necesidad de ingresar parámetros (ambos atributos se inician con el valor 0), en cambio el otro constructor de Calculadora requiere de parámetros de ingreso para inicializar los atributos de la clase. Dentro de los métodos de la clase Calculadora podemos mencionar: el método sumar que suma los atributos del objeto instanciado y retorna el resultado, el método multiplicar que multiplica los atributos del objeto y retorna el resultado, los métodos setN1 y setN2 que corresponden a métodos setter que posibilitan la modificación de los valores de los atributos del objeto en cuestión.

   La clase CarroCompra posee un atributo de tipo arreglo de enteros de dimensiones 2 por 5. Esta clase posee un constructor que le da valores a los elementos de la matriz, las entradas de la fila 1 serán 1 y las entradas de la fila 2 será el número 1000. Dentro de los métodos de la clase Carrocompra podemos mencionar: el método privado calcularTotal crea dos variables enteras, **total** y **subtotal** asignándoles el valor 0 a ambas variables, para luego utilizar un bucle for donde se hace uso del método subTotal el cual tiene como argumentos los valores de la primer y segunda fila de la matriz, el valor retornado por el método subTotal en cada iteración es sumado a la variable total; en el método privado subtotal se instancia un objeto de la clase Calculadora utilizando como parámetros del constructor los parámetros int cant e int precio, para luego retornar a través del método multiplicar de la clase Calculadora la multiplicación de estos atributos; por último tenemos el metodo mostrarTotal cuya función es mostrar por pantalla el valor retornado por el método CalcularTotal del objeto en cuestión.

3. De lo anterior, establezca una representación detallada del código fuente, usando un diagrama de clases UML y la herramienta de modelado Visual Paradigm.

4. Genere un código fuente Java a partir de su modelo de clases.

```java
public class Calculadora {


   private int n1;

   private int n2;


   public Calculadora() {

      // TODO - implement Calculadora.Calculadora

      throw new UnsupportedOperationException();

   }


   /**

    *

    * @param num1

    * @param num2

    */

   public Calculadora(int num1, int num2) {

      // TODO - implement Calculadora.Calculadora

      throw new UnsupportedOperationException();

   }


   public void sumar() {
```

```
        // TODO - implement Calculadora.sumar
        throw new UnsupportedOperationException();
    }

    public void multiplicar() {
        // TODO - implement Calculadora.multiplicar
        throw new UnsupportedOperationException();
    }

    /**
     *
     * @param num1
     */
    public void setN1(int num1) {
        this.n1 = num1;
    }

    /**
     *
     * @param num2
     */
    public void setN2(int num2) {
        this.n2 = num2;
    }

}
```

```java
public class CarroCompra {

    private int[][] productos = new int[2][5];

    public CarroCompra() {
        // TODO - implement CarroCompra.CarroCompra
        throw new UnsupportedOperationException();
    }

    private int calcularTotal() {
        // TODO - implement CarroCompra.calcularTotal
        throw new UnsupportedOperationException();
    }

    /**
     *
     * @param cant
     * @param precio
     */
    private int subtotal(int cant, int precio) {
        // TODO - implement CarroCompra.subtotal
        throw new UnsupportedOperationException();
    }

    public void mostrarTotal() {
        // TODO - implement CarroCompra.mostrarTotal
```

```
        throw new UnsupportedOperationException();

    }



}
```