KEVIN ALEXANDER AQUINO VASQUEZ

PROGRAMA CALCULADORA

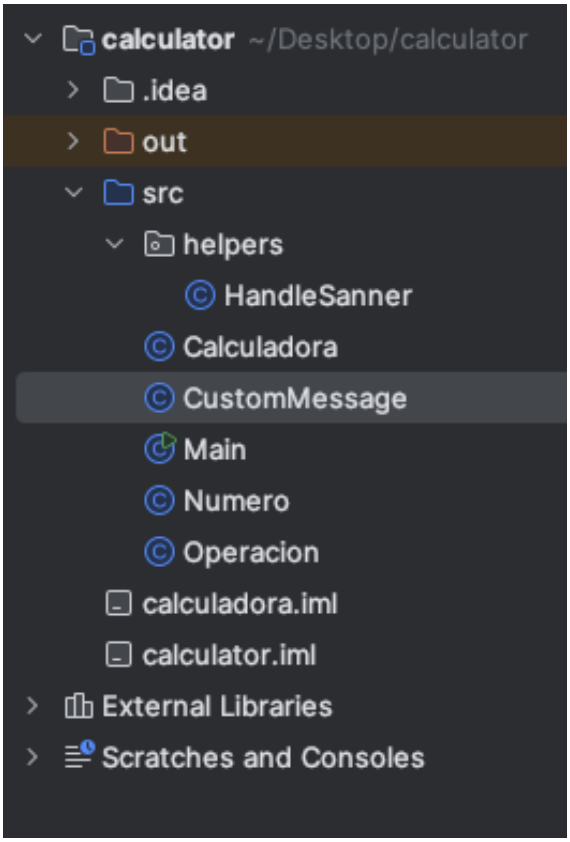
KODIGO

Tabla de contenido

Descripcion de directorio

Como se puede visualizar en la ilustracion 1, tenemos nuestro directorio de archivos, los cuales contiene lo siguiente:

- Clase operación
- Clase Numero
- Clase CustomMessage
- Clase Calculadora
- Clase handleScanner
- Clase Main



Clase HandleScanner

Clase que implementa el uso del Scanner con su respectivas validaciones

Dicha clase cuenta con dos metodos estaticos, los cuales son:

- getInt
- getDouble

los dos metodos reciben como parametro “customMessage” de tipo String.

Metodo getInt

Metodo estatico, que implementa validaciones a nivel de consola para los valores que ingresa los usuario, si dicho valor no es del tipo solicitado, se solicita nuevamente

```java
/**
 * @param customMessage String
 * @return int
 * @name getInt
 * @description Metodo estatico, que implementa validaciones a nivel de consola para los valores que
 * ingresa los usuario, si dicho valor no es del tipo solicitado, se solicita nuevamente
 */
public static int getInt(String customMessage) {  2 usages  ± kenvas-dev
    Scanner sc = new Scanner(System.in);
    do {
        if (sc.hasNext()) {
            if (sc.hasNextInt()) {
                return sc.nextInt();
            } else {
                sc.next();
                System.out.println("No ha introducido un número.");
                System.out.println("Favor introducir un numero del menu de arriba: ");
                System.out.println(customMessage);
            }
        } else {
            return 0;
        }
    } while (true);
}
```

Metodo getDouble

Metodo estatico, que implementa validaciones a nivel de consola para los valores que ingresa los usuario, si dicho valor no es del tipo solicitado, se solicita nuevamente

```java
    /**
     * @param customMessage String
     * @return double
     * @name getDouble
     * @description Metodo estatico, que implementa validaciones a nivel de consola para los valores que
     * ingresa los usuario, si dicho valor no es del tipo solicitado, se solicita nuevamente
     */
    public static double getDouble(String customMessage) {  2 usages   ± kenvas-dev
        Scanner sc = new Scanner(System.in);
        do {
            if (sc.hasNext()) {
                if (sc.hasNextDouble()) {
                    return sc.nextDouble();
                } else {
                    sc.next();
                    System.out.println("No ha introducido un número.");
                }
            } else {
                return 0;
            }
        } while (true);
    }
}
```

Clase CustomMessage

Clase que implementa el uso del System.out.print.

Clase presonalizada para mostrar mensajes.

Dicha clase cuenta con dos metodos estaticos, los cuales son:

- title
- out
- outln

los dos metodos (out, outln) reciben como parametro “texts” de tipo String[].

El metodo title recibe unicamente el parametro “title” de tipo String

Metodo title

Banner creado para mostrar enunciados

```
/**
 * @name title
 * @description
 * Banner creado para mostrar enunciados
 * @param title
 */
public static void title(String title) {  1 usage   kenvas-dev
    System.out.println("*****************************************");
    System.out.println("*** "+title+" ***");
    System.out.println("*****************************************");
}
```

Ilustración 1 - Metodo title

Metodo out

Es una personalizacion del syop para un listado de opciones, sin saltos de linea.

```java
/**
 * @name out
 * @description
 * Es una personalizacion del syop
 * para un listado de opciones, sin saltos de linea.
 * @param texts
 */
public static void out(String[] texts) {  kenvas-dev
    for (String text : texts) {
        System.out.print(text+"   ");
    }
}
```

Ilustración 2 - Metodo out

Metodo outln

Es una personalizacion del syop para un listado de opciones, con saltos de linea.

```java
/**
 * @name outln
 * @description
 * Es una personalizacion del syop
 * para un listado de opciones, con saltos de linea.
 * @param texts
 */
public static void outln(String[] texts) {  18 usages  kenvas-dev
    for (String text : texts) {
        System.out.println(text);
    }
}
```

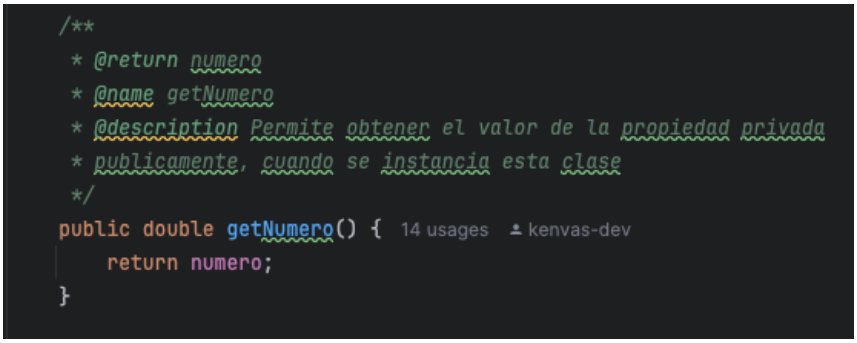
Ilustración 3 - Metodo outln

Clase Numero

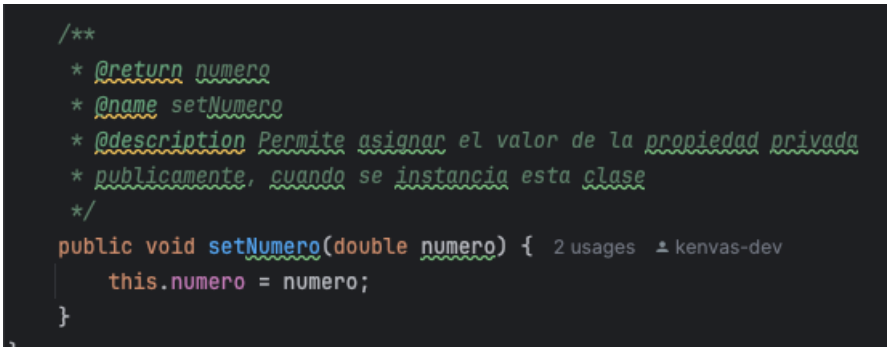
Clase Numero que permite crear un objeto con sus propiedad 'numero'

Consta de una propiedad privada llamada numero de tipo double y sus metodos de acceso, set y get.

Metodo get



Metodo set



Clase Operación

En esta clase se realiza la ejecucion de los calculos aritmeticos solicitados, los cuales son:

- SUMA
- RESTA
- MULTIPLICACION
- DIVISION
- POTENCIA
- RAIZ

Metodo suma

Metodo responsable de realizar la operacion de la suma, y retorna el resultado de dicha suma. Si falla retorna ‘0’.

```java
/**
 * @return number1 + number2;
 * @name suma
 * @description metodo responsable de realizar la operacion de la suma
 */
public double suma(double number1, double number2) {  1 usage   kenvas-dev
    try {
        return number1 + number2;
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo resta

Metodo responsable de realizar la operacion de la resta, y retorna el resultado de dicha resta. Si falla retorna ‘0’.

```java
/**
 * @return number1 - number2
 * @name resta
 * @description metodo responsable de realizar la operacion de la resta
 */
public double resta(double number1, double number2) {  1 usage  kenvas-dev
    try {
        return number1 - number2;
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo multiplicacion

Metodo responsable de realizar la operacion de la multiplicacion, y retorna el resultado de dicha multiplicacion. Si falla retorna ‘0’.

```java
/**
 * @return number1 * number2
 * @name multiplicacion
 * @description metodo responsable de realizar la operacion de la multiplicacion
 */
public double multiplicacion(double number1, double number2) {  1 usage  kenvas-dev
    try {
        return number1 * number2;
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo division

Metodo responsable de realizar la operacion de la division, y retorna el resultado de dicha division. Si falla retorna ‘0’.

```java
/**
 * @return number1 / number2
 * @name division
 * @description metodo responsable de realizar la operacion de la division
 */
public double division(double number1, double number2) {  1 usage  kenvas-dev
    try {
        return number1 / number2;
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo potencia

Metodo responsable de realizar la operacion de la potencia, y retorna el resultado de dicha potencia. Si falla retorna ‘0’.

```java
/**
 * @return Math.pow(number1, number2)
 * @name potencia
 * @description metodo responsable de realizar la operacion de la potencia.
 * Donde 'number1' es la base y 'number2' es el exponente
 */
public double potencia(double number1, double number2) {  1 usage  kenvas-dev
    try {
        return Math.pow(number1, number2);
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo raiz

Metodo responsable de realizar la operacion de la raiz, y retorna el resultado de dicha raiz. Si falla retorna ‘0’.

```java
/**
 * @return Math.sqrt(number1)
 * @name raiz
 * @description metodo responsable de realizar la operacion de la raiz de un numero
 */
public double raiz(double number1, double number2) {  1 usage   kenvas-dev
    try {
        return Math.sqrt(number1);
    } catch (Exception e) {
        return 0;
    }
}
```

Clase Calculadora

En esta clase se realiza la implementacion de los calculos aritmeticos solicitados, los cuales son:

- SUMA
- RESTA
- MULTIPLICACION
- DIVISION
- POTENCIA
- RAIZ

Cuenta con las siguientes propiedades, las cuales son:

- lblNumber1
- lblNumber2
- lblOption
- number1
- number2
- sc
- operacionImplement

Metodo suma

Metodo responsable de realizar la implementacion de la suma con sus respectivas validaciones.

```
/**
 * @return number1 + number2;
 * @name suma
 * @description metodo responsable de realizar la implementacion de la suma
 */
public double suma() {  1 usage   kenvas-dev
    getParameters( operation: "Suma",  hasMultipleOperation: true);
    return operacionImplement.suma(number1.getNumero(), number2.getNumero());
}
```

Metodo resta

Metodo responsable de realizar la implementacion de la resta con sus respectivas validaciones.

```java
/**
 * @return number1 - number2
 * @name resta
 * @description metodo responsable de realizar la implementacion de la resta
 */
public double resta() {  1 usage   kenvas-dev
    getParameters( operation: "Resta",  hasMultipleOperation: true);
    return operacionImplement.resta(number1.getNumero(), number2.getNumero());
}
```

Metodo multiplicacion

Metodo responsable de realizar la implementacion de la multiplicacion con sus respectivas validaciones.

```java
/**
 * @return number1 * number2
 * @name multiplicacion
 * @description metodo responsable de realizar la implementacion de la multiplicacion
 */
public double multiplicacion() {  1 usage   kenvas-dev
    getParameters( operation: "Multiplicacion",  hasMultipleOperation: true);
    return operacionImplement.multiplicacion(number1.getNumero(), number2.getNumero());
}
```

Metodo division

Metodo responsable de realizar la implementacion de la division con sus respectivas validaciones.

```java
/**
 * @return number1 / number2
 * @name division
 * @description metodo responsable de realizar la implementacion de la division
 */
public double division() {  1 usage   kenvas-dev
    getParameters( operation: "Division",  hasMultipleOperation: true);
    if (number2.getNumero() == 0) {
        CustomMessage.outln(new String[]{"No se puede realizar una divisio entres '0'"});
        return 0;
    }
    return operacionImplement.division(number1.getNumero(), number2.getNumero());
}
```

Metodo potencia

Metodo responsable de realizar la implementacion de la potencia con sus respectivas validaciones.

```java
/**
 * @return Math.pow(number1, number2)
 * @name potencia
 * @description metodo responsable de realizar la implementacion de la potencia.
 */
public double potencia() {  1 usage   kenvas-dev
    getParameters( operation: "Potencia",  hasMultipleOperation: true);
    return operacionImplement.potencia(number1.getNumero(), number2.getNumero());
}
```

Metodo raiz

Metodo responsable de realizar la implementacion de la raiz con sus respectivas validaciones.

```java
/**
 * @return Math.sqrt(number1)
 * @name raiz
 * @description metodo responsable de realizar la implementacion de la raiz de un numero
 */
public double raiz() {  1 usage   kenvas-dev
    getParameters( operation: "Raiz cuadrada",  hasMultipleOperation: false);
    if (number1.getNumero() < 0) {
        CustomMessage.outln(new String[]{"Tienes que ingresar un numero igual o mayor a '0'"});
        return 0;
    }
    return operacionImplement.raiz(number1.getNumero(), number2.getNumero());
}
```

Metodo getParameters

Metodo responsable de solicitar los datos al usuario

```java
/**
 * @name getParameters
 * @description metodo responsable de solicitar los datos al usuario
 */
public void getParameters(String operation, boolean hasMultipleOperation) {  6 usages   kenvas-dev
    CustomMessage.outln(new String[]{lblOption + operation});
    CustomMessage.outln(new String[]{lblNumber1});
    double number1 = HandleScanner.getDouble( customMessage: "");
    this.number1.setNumero(number1);
    if (hasMultipleOperation) {
        CustomMessage.outln(new String[]{lblNumber2});
        double number2 = HandleScanner.getDouble( customMessage: "");
        this.number2.setNumero(number2);
    }
}
```

Clase Main

Clase principal la cual llama a la clase calculadora para realizar las operaciones.

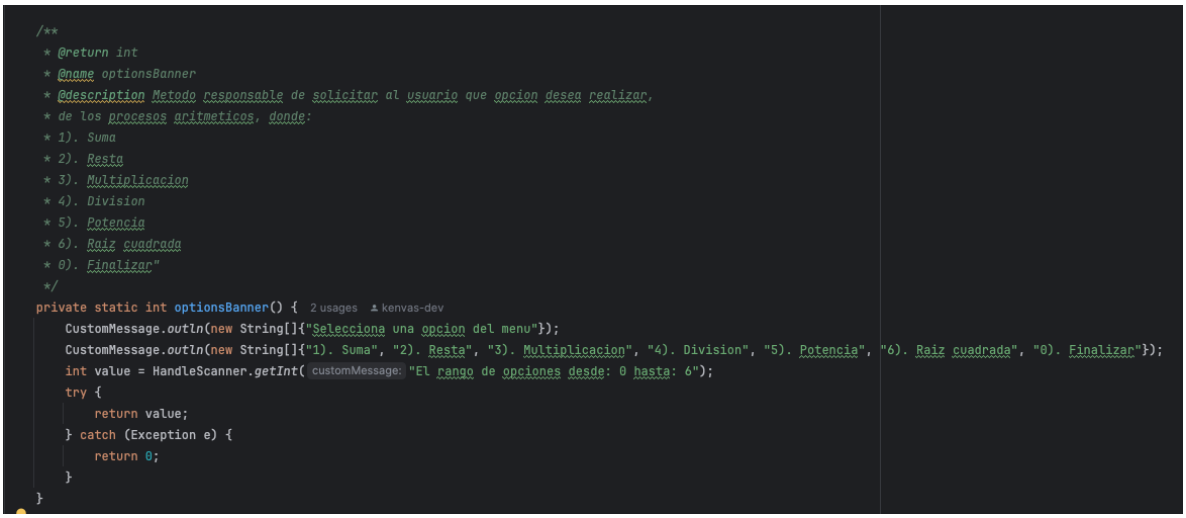
Esta clase cuenta con las siguientes propiedades y metodos:

- minOptionSelected
- optionSelected
- optionSelectedContinue
- hasContinue
- sc
- Metodo main
- Metodo optionsBanner
- Metodo optionsContinue
- Metodo runCalculator

Metodo optionsBanner

Metodo responsable de solicitar al usuario que opcion desea realizar, de los procesos aritmeticos, donde:

- 1). Suma
- 2). Resta
- 3). Multiplicacion
- 4). Division
- 5). Potencia
- 6). Raiz cuadrada
- 0). Finalizar



```java
/**
 * @return int
 * @name optionsBanner
 * @description Metodo responsable de solicitar al usuario que opcion desea realizar,
 * de los procesos aritmeticos, donde:
 * 1). Suma
 * 2). Resta
 * 3). Multiplicacion
 * 4). Division
 * 5). Potencia
 * 6). Raiz cuadrada
 * 0). Finalizar"
 */
private static int optionsBanner() {  2 usages  kenvas-dev
    CustomMessage.outln(new String[]{"Selecciona una opcion del menu"});
    CustomMessage.outln(new String[]{"1). Suma", "2). Resta", "3). Multiplicacion", "4). Division", "5). Potencia", "6). Raiz cuadrada", "0). Finalizar"});
    int value = HandleScanner.getInt( customMessage: "El rango de opciones desde: 0 hasta: 6");
    try {
        return value;
    } catch (Exception e) {
        return 0;
    }
}
```

Metodo optionsContinue

Metodo responsable de solicitar al usuario si desea continuar o no,realizando procesos aritmeticos

```java
/**
 * @name optionsContinue
 * @description Metodo responsable de solicitar al usuario si desea continuar o no,
 * realizando procesos aritmeticos
 */
private static void optionsContinue() {  2 usages  kenvas-dev
    CustomMessage.outln(new String[]{"Desea continuar?"});
    CustomMessage.outln(new String[]{"1). Si", "0). No",});
    int option = HandleScanner.getInt( customMessage: "El rango de opciones desde: 0 hasta: 1");
    optionSelectedContinue = option;
    if (option == 0) {
        hasContinue = false;
    }
}
```

Metodo runCalculator

Metodo responsable de centralizar los metodos para poder hacer un calculo aritmetico.

```
private static void runCalculator() {  1 usage   kenvas-dev

    while (true) {
        try {
            // Se muestra las opciones para realizar un calculo artimetico
            optionSelected = optionsBanner();
            if (optionSelected == minOptionSelected) {
                CustomMessage.outln(new String[]{"Gracias por usar mi calculadora :D "});
                break;
            }

            //Implementacion de la clase calculadora
            Calculadora calculadora = new Calculadora();

            // Se valida la opcion seleccionada
            // En base a la opcion seleccionada, se realiza el calculo
            while ((optionSelected > minOptionSelected && optionSelected <= maxOptionSelected) && hasContinue && (optionSelectedContinue >= 0 && optionSelectedContinue <= 1)) {
                switch (optionSelected) {
                    case 1:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.suma()});
                        break;
                    case 2:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.resta()});
                        break;
                    case 3:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.multiplicacion()});
                        break;
                    case 4:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.division()});
                        break;
                    case 5:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.potencia()});
                        break;
                    case 6:
                        CustomMessage.outln(new String[]{"El resultado es: " + calculadora.raiz()});
                        break;

                }

                // Se consulta al ususario si desea continuar
                optionsContinue();
                while (optionSelectedContinue < 0 || optionSelectedContinue > 1) {
                    optionsContinue();

                }
                if (hasContinue) {
                    optionSelected = optionsBanner();

                    if (optionSelected == minOptionSelected) {
                        hasContinue = false;
                        break;
                    }
                }
                if (!hasContinue) break;
            }
            if (!hasContinue) {
                CustomMessage.outln(new String[]{"Gracias por usar mi calculadora :D "});
                break;
            }
        } catch (Exception e) {

            // Se emite un error controlado, si el usuario ingresa un valor no comprendido
            CustomMessage.outln(new String[]{"Al parecer insertaste un valor que no esta en el rango de opciones esperado"});
        }
    }
}
```

Metodo main

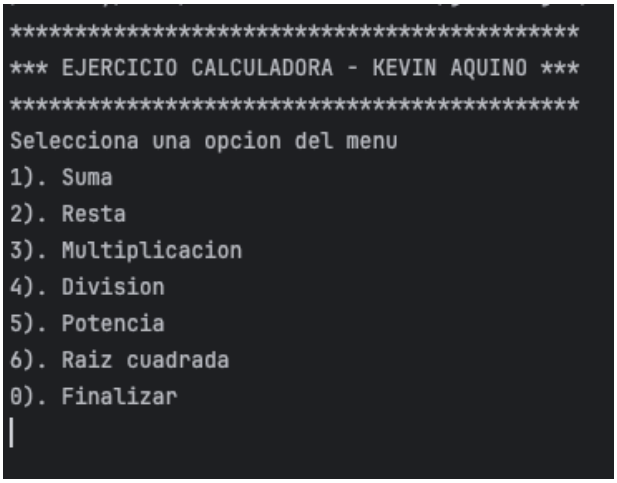
Implementacion del programa

```java
/**
 * @param args
 * @name main
 * @description Implementacion del programa
 */
public static void main(String[] args) {  kenvas-dev
    CustomMessage.title("EJERCICIO CALCULADORA - KEVIN AQUINO");
    runCalculator();
}
```

Ejecucion del programa

Menu de opciones

Se debe de digitar una opcion de las 7 que se muestra, si se selecciona una diferente, vuelve a mostrar el mismo mensaje, hasta que ekl usuario digite una correcta, o finalice el programa.



En la siguiente imagen se puede observar si el usuario digita una opcion que no es la correcta, similar es el mensaje cuando se digita algo incorrecto.

```
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
****************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
sxx
No ha introducido un número.
Favor introducir un número del menu de arriba:
El rango de opciones desde: 0 hasta: 6
```

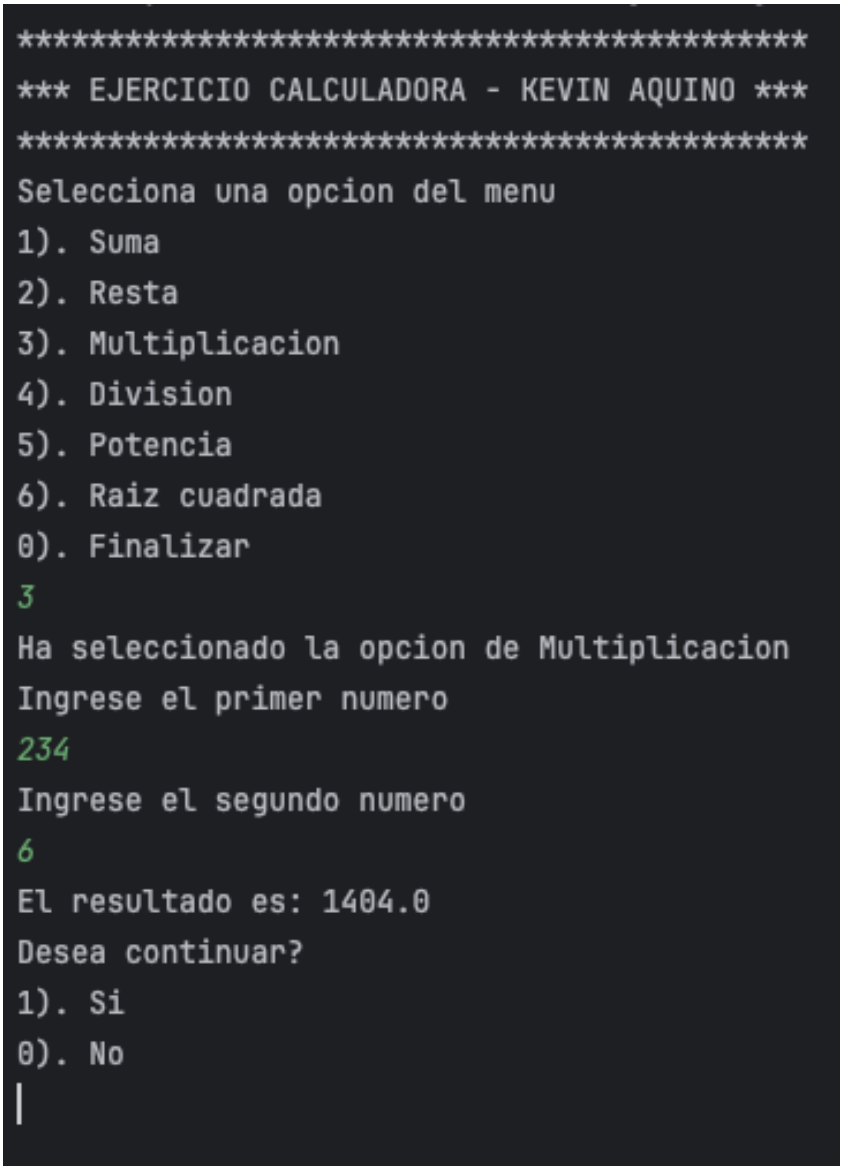
Suma

```
**********************************************
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
**********************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
1
Ha seleccionado la opcion de Suma
Ingrese el primer numero
23
Ingrese el segundo numero
5644
El resultado es: 5667.0
Desea continuar?
1). Si
0). No
```

Resta

```
*****************************************
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
*****************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
2
Ha seleccionado la opcion de Resta
Ingrese el primer numero
345
Ingrese el segundo numero
65
El resultado es: 280.0
Desea continuar?
1). Si
0). No
```

Multiplicacion



Division

```
*****************************************
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
*****************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
4
Ha seleccionado la opcion de Division
Ingrese el primer numero
100
Ingrese el segundo numero
2
El resultado es: 50.0
Desea continuar?
1). Si
0). No
```

Potencia

```
*****************************************
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
*****************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
5
Ha seleccionado la opcion de Potencia
Ingrese el primer numero
23
Ingrese el segundo numero
2
El resultado es: 529.0
Desea continuar?
1). Si
0). No
```

Raiz

```
*****************************************
*** EJERCICIO CALCULADORA - KEVIN AQUINO ***
*****************************************
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
6
Ha seleccionado la opcion de Raiz cuadrada
Ingrese el primer numero
23
El resultado es: 4.795831523312719
Desea continuar?
1). Si
0). No
```

Continuar

```
Desea continuar?
1). Si
0). No
1
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
```

Finalizar

```
Selecciona una opcion del menu
1). Suma
2). Resta
3). Multiplicacion
4). Division
5). Potencia
6). Raiz cuadrada
0). Finalizar
0
Gracias por usar mi calculadora :D
```