

Resolución paso a paso

Paso A: Las tareas se distribuyeron equitativamente, asignando dos tareas a cada miembro del grupo. Joaquín Silva se encargó de crear un menú que conectó todos los métodos y realizó la Tarea 3, que implicaba calcular perímetros, áreas y volúmenes según correspondía para las siguientes figuras geométricas: cuadrado, rectángulo, círculo, esfera, cubo y cono. Gerson Urrea asumió la responsabilidad de las Tareas 1 y 5, que consisten en calcular las cuatro operaciones aritméticas principales y calcular la ecuación de una recta. Por último, Sebastián Martínez se encargó de las Tareas 2 y 4, que involucraron calcular las soluciones de una ecuación cuadrática y obtener las soluciones de un sistema de ecuaciones. Finalmente se realizaron pruebas unitarias

Paso B: Después de realizar las pruebas unitarias, es posible identificar diversas excepciones que pueden ser aprovechadas en nuestro beneficio para minimizar errores y aplicar try-catch, contribuyendo así a mejorar nuestro código.

Le ponemos la respuesta a esto:

B.) Ahora a partir de lo anterior, considere lo siguiente:

- ¿Qué pasa si se intenta ingresar como divisor un CERO?

Inicialmente, al intentar utilizar 0 como divisor, se obtenía un resultado de "infinity" o, en otras palabras, una indeterminación. Este problema se resolvió mediante la implementación de un método que previene este escenario.

- ¿Qué pasa si la base y exponente de la potencia son CERO?

En las etapas iniciales del método de cálculo de potencias, cuando se utilizaba 0 como base y exponente, se obtenía un resultado de 1 debido a un diseño preestablecido de la función Math.pow. Para abordar este problema, se implementó un método que evita que ambos valores sean simultáneamente iguales a 0.

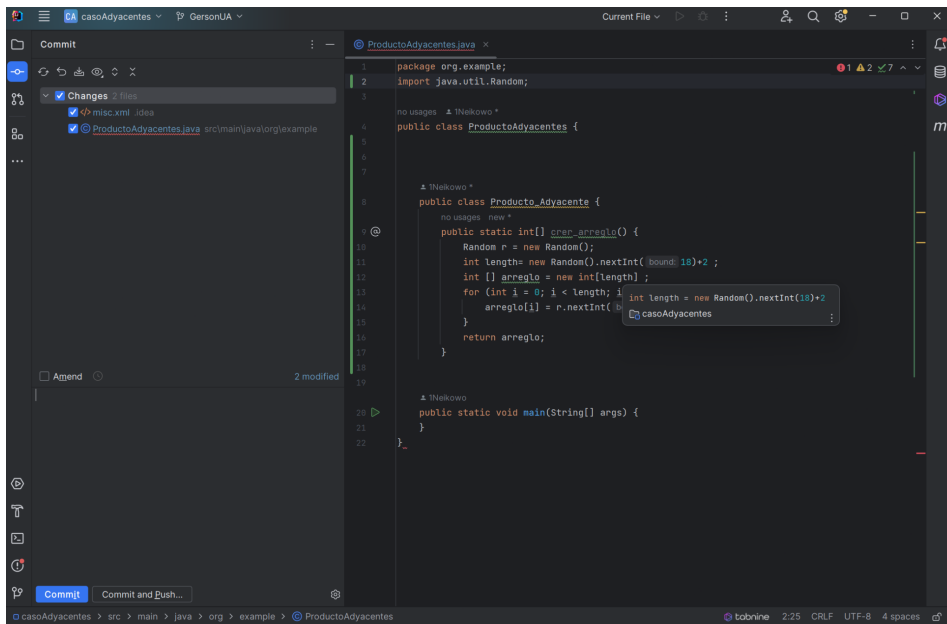
- ¿Qué pasa si se intenta ingresar una variable de entrada no numérica?

Al intentar ingresar una variable no numérica, se producía un error de InputMismatchException. Para resolver este problema, se implementó un bloque try-catch en el cual la sección catch maneja esta excepción al mostrar un mensaje de error correspondiente.

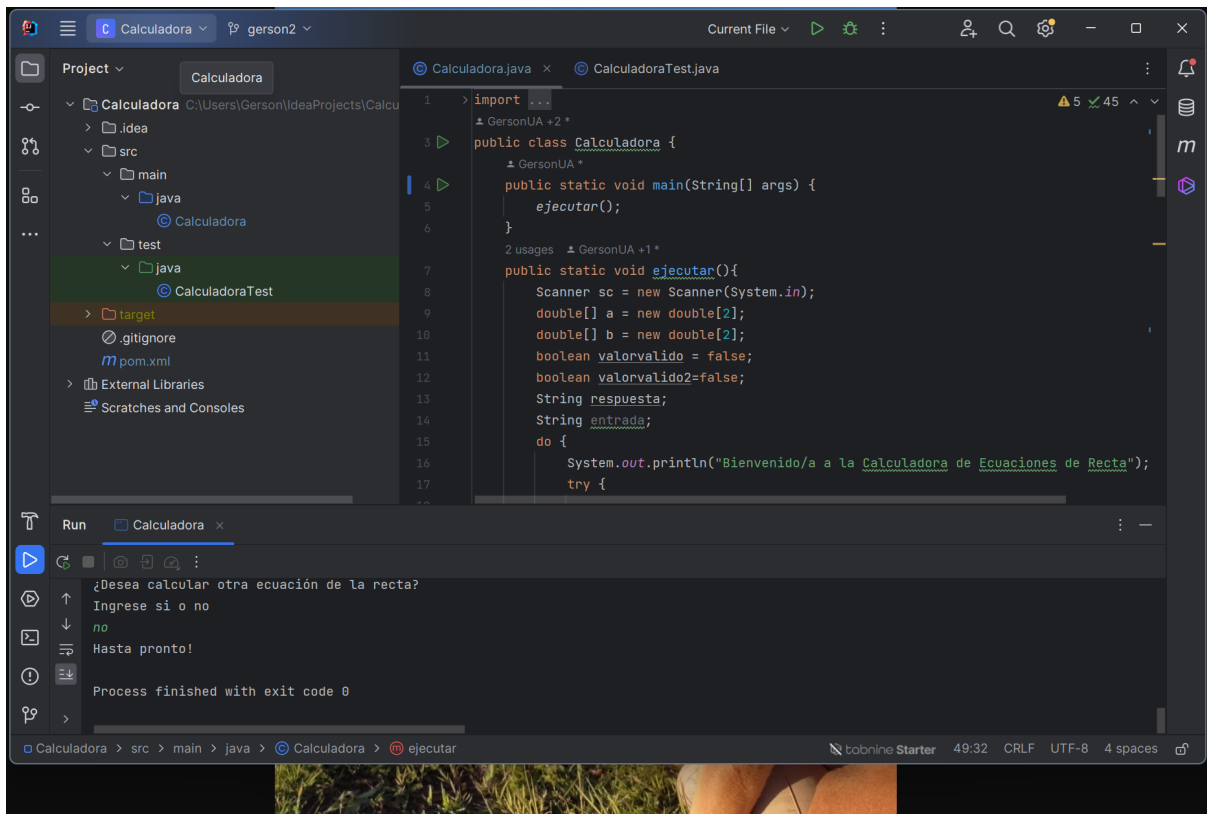
Paso C: A partir de todas las soluciones desarrolladas, se subieron todos los códigos a GitHub con el propósito de llevar a cabo un merge que integrara todas estas soluciones en un único conjunto de código.

Evidencia Trabajo de cada Integrante

Gerson Urrea

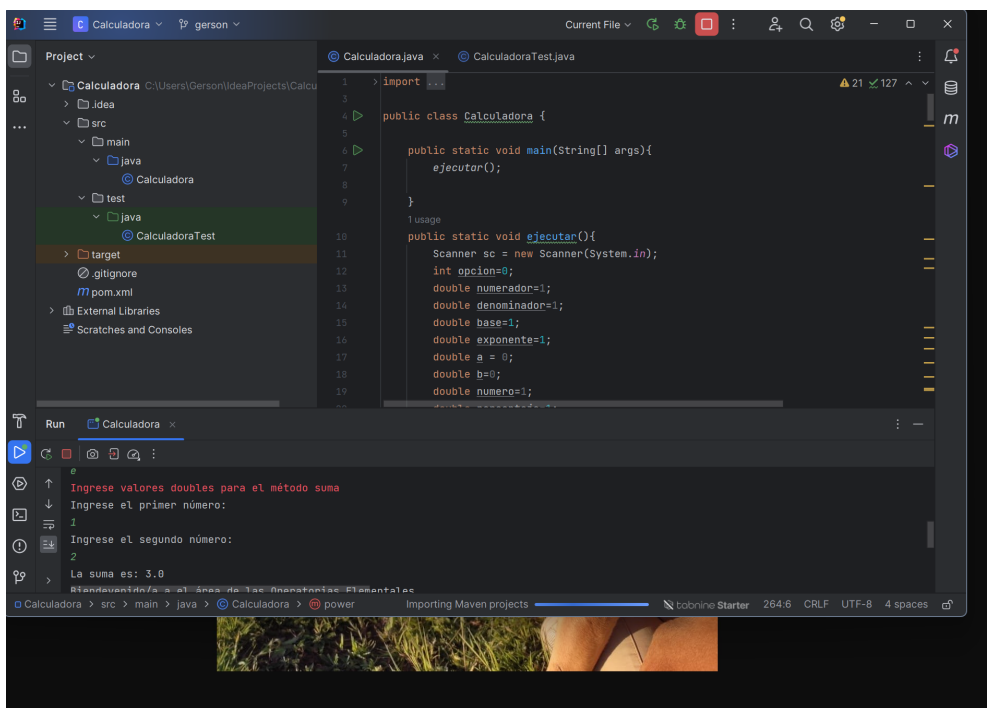
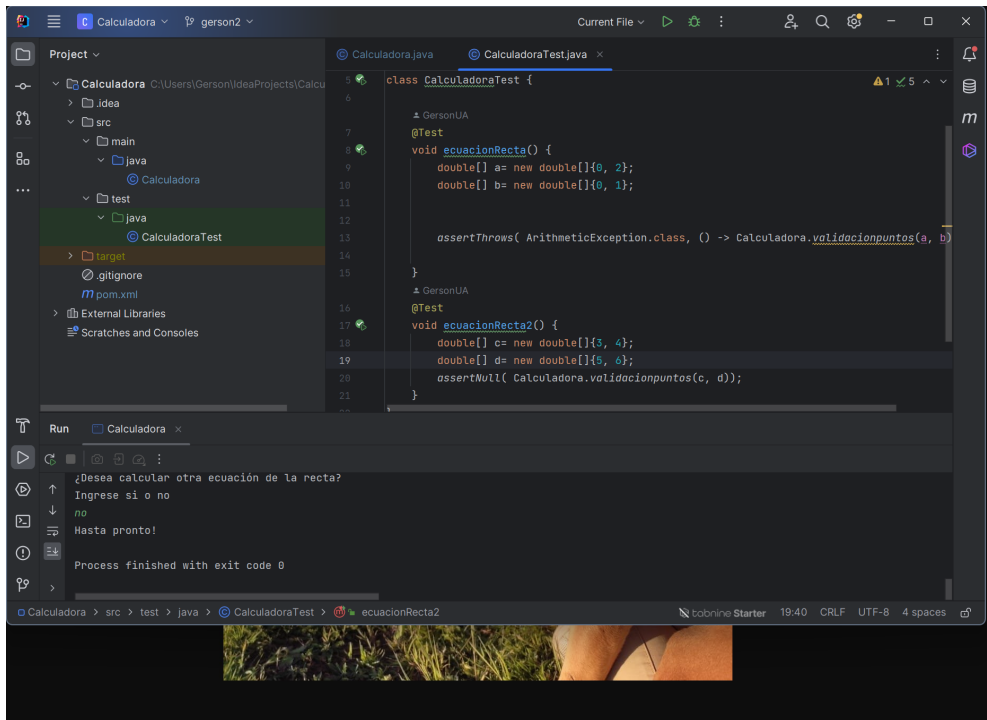


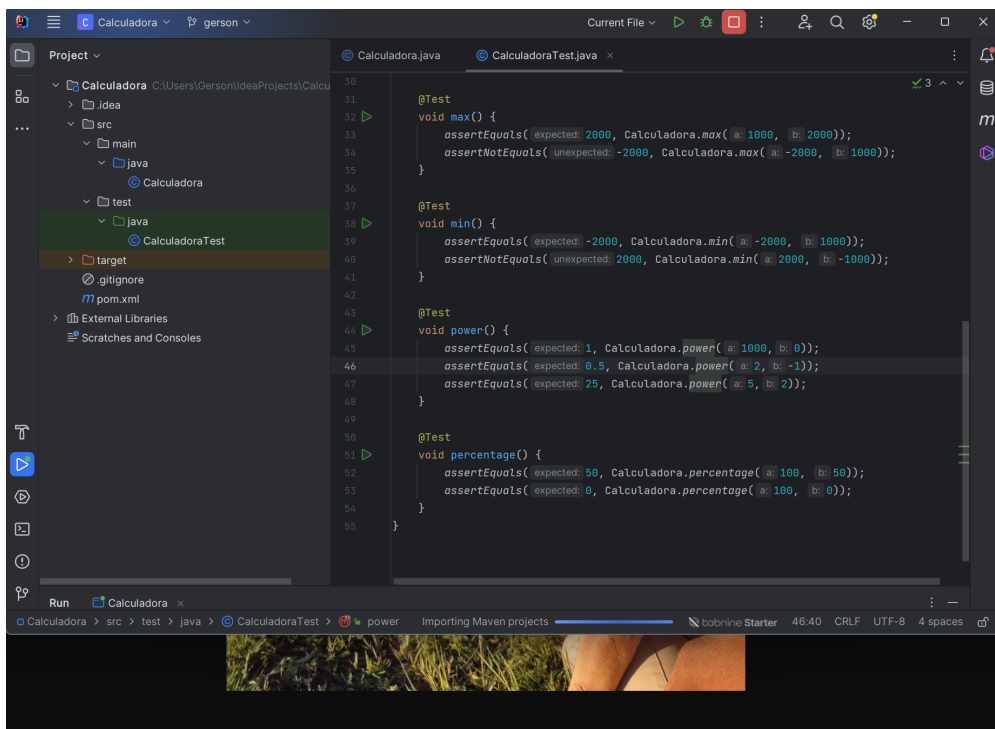
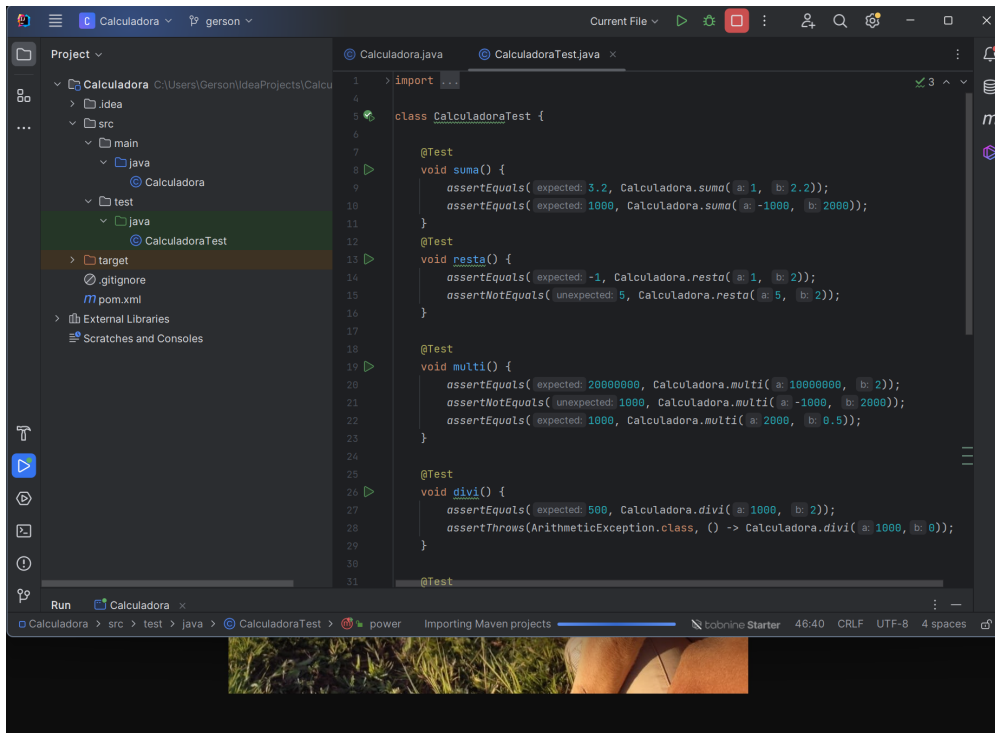
```
1 package org.example;
2 import java.util.Random;
3
4 no usages *
5 public class ProductoAdyacentes {
6
7     + Nuekovo *
8     public class ProductoAdyacente {
9         no usages new *
10         public static int[] generarArreglo() {
11             Random r = new Random();
12             int length = new Random().nextInt(10)+2;
13             int[] arreglo = new int[length];
14             for (int i = 0; i < length; i++) {
15                 int length = new Random().nextInt(10)+2;
16                 arreglo[i] = r.nextInt(length);
17             }
18             return arreglo;
19         }
20
21         + Nuekovo *
22         public static void main(String[] args) {
23             // TODO: write your code here
24         }
25     }
26 }
```



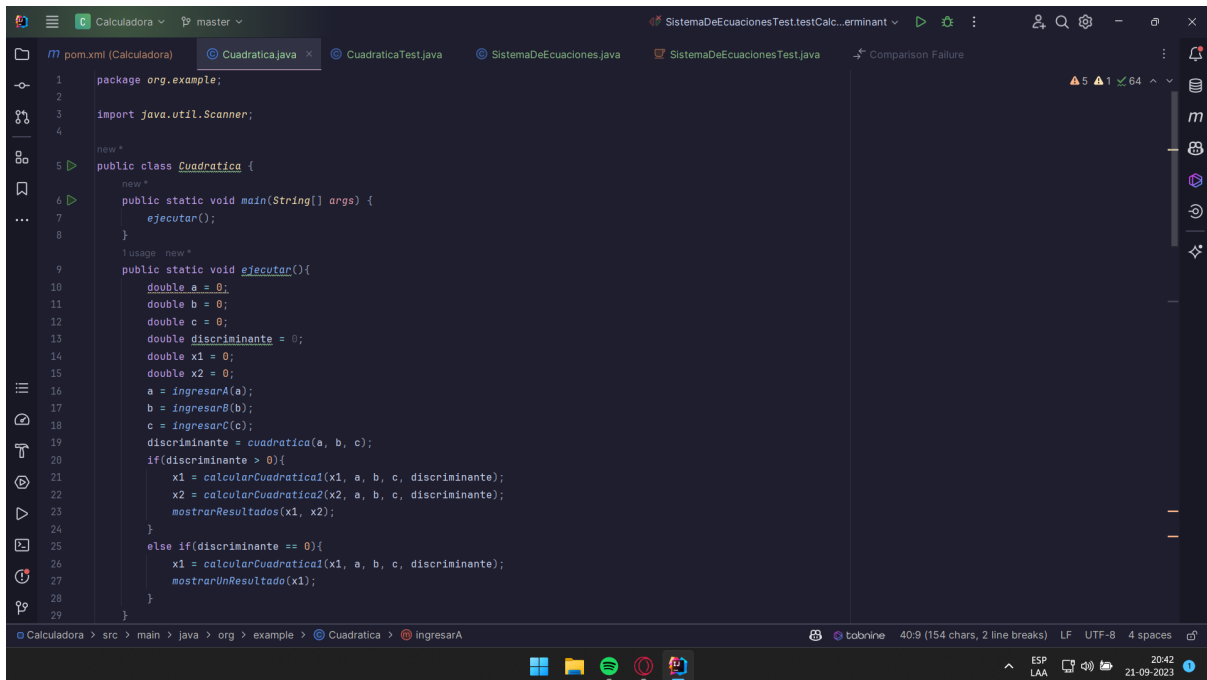
```
1 > import java.util.Scanner;
2 + GersonUA +2 *
3
4 public class Calculadora {
5     + GersonUA *
6     public static void main(String[] args) {
7         ejecutar();
8     }
9
10 2 usages + GersonUA +1 *
11 public static void ejecutar(){
12     Scanner sc = new Scanner(System.in);
13     double[] a = new double[2];
14     double[] b = new double[2];
15     boolean valorvalido = false;
16     boolean valorvalido2=false;
17     String respuesta;
18     String entrada;
19     do {
20         System.out.println("Bienvenido/a a la Calculadora de Ecuaciones de Recta");
21         try {
22             // TODO: write your code here
23         }
24     } while (true);
25 }
```

```
Run: Calculadora
?Desea calcular otra ecuación de la recta?
Ingrese si o no
no
Hasta pronto!
Process finished with exit code 0
```

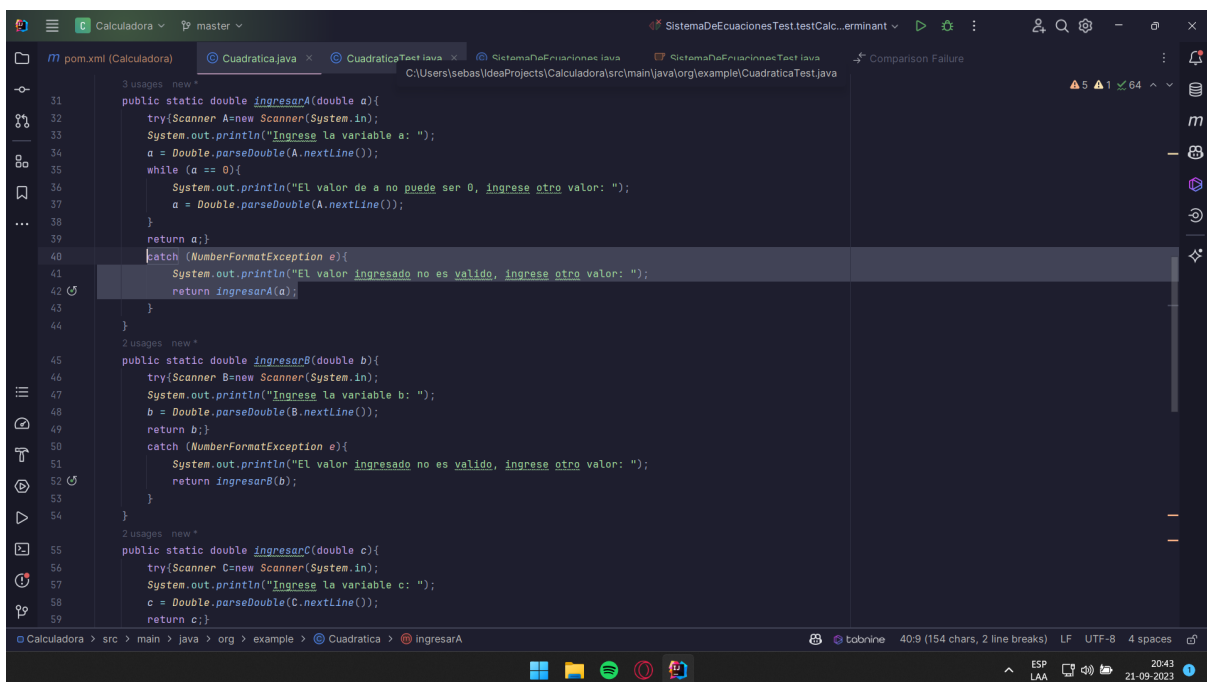




Sebastian Martinez



```
1 package org.example;
2
3 import java.util.Scanner;
4
5 new *
6 public class Quadratica {
7     new *
8     public static void main(String[] args) {
9         ejecutar();
10    }
11
12    usage new *
13    public static void ejecutar(){
14        double a = 0;
15        double b = 0;
16        double c = 0;
17        double discriminante = 0;
18        double x1 = 0;
19        double x2 = 0;
20        a = ingresarA(a);
21        b = ingresarB(b);
22        c = ingresarC(c);
23        discriminante = cuadratica(a, b, c);
24        if(discriminante > 0){
25            x1 = calcularCuadratica1(x1, a, b, c, discriminante);
26            x2 = calcularCuadratica2(x2, a, b, c, discriminante);
27            mostrarResultados(x1, x2);
28        }
29        else if(discriminante == 0){
30            x1 = calcularCuadratica1(x1, a, b, c, discriminante);
31            mostrarUnResultado(x1);
32        }
33    }
34 }
```



```
31 3 usages new *
32 public static double ingresarA(double a){
33     try{Scanner A=new Scanner(System.in);
34     System.out.println("Ingrese la variable a: ");
35     a = Double.parseDouble(A.nextLine());
36     while (a == 0){
37         System.out.println("El valor de a no puede ser 0, ingrese otro valor: ");
38         a = Double.parseDouble(A.nextLine());
39     }
40     return a;}
41
42 catch (NumberFormatException e){
43     System.out.println("El valor ingresado no es valido, ingrese otro valor: ");
44     return ingresarA(a);
45 }
46
47 2 usages new *
48 public static double ingresarB(double b){
49     try{Scanner B=new Scanner(System.in);
50     System.out.println("Ingrese la variable b: ");
51     b = Double.parseDouble(B.nextLine());
52     return b;}
53
54 catch (NumberFormatException e){
55     System.out.println("El valor ingresado no es valido, ingrese otro valor: ");
56     return ingresarB(b);
57 }
58
59 2 usages new *
60 public static double ingresarC(double c){
61     try{Scanner C=new Scanner(System.in);
62     System.out.println("Ingrese la variable c: ");
63     c = Double.parseDouble(C.nextLine());
64     return c;}
65 }
```

```
1 package org.example;
2
3 import java.sql.SQLException;
4 import java.util.Scanner;
5
6 new *
7 public class SistemaDeEcuaciones {
8     new *
9     public static void main(String[] args) {
10         System.out.println("Ingresa los valores de las variables de un sistema de ecuacion de la forma ax+by=c y dx+ey=f:");
11         ejecutar();
12     }
13
14     Usage new *
15     public static void ejecutar(){
16         double a = 0;
17         double b = 0;
18         double c = 0;
19         double d = 0;
20         double e = 0;
21         double f = 0;
22         double determinante = 0;
23         double x = 0;
24         double y = 0;
25         a = ingresarA(a);
26         b = ingresarB(b);
27         c = ingresarC(c);
28         d = ingresarD(d);
29         e = ingresarE(e);
30         f = ingresarF(f);
31         determinante = calculoDeterminante(a, b, c, d, e, f, determinante);
32         if(determinante != 0){
33             x = calculoSolucion1(b, c, e, f, determinante);
34         }
35     }
36 }
```

```
1 package org.example;
2
3 import org.example.Cuadratica;
4 import org.junit.jupiter.api.Test;
5 import java.io.ByteArrayInputStream;
6 import java.io.InputStream;
7 import static org.junit.jupiter.api.Assertions.*;
8
9 new *
10 public class CuadraticaTest {
11     new *
12     @Test
13     public void ingresarValidoTest() {
14         // Simular entrada de usuario (por ejemplo, "2.0\n")
15         String input = "2.0\n";
16         ByteArrayInputStream in = new ByteArrayInputStream(input.getBytes());
17         System.setIn(in);
18
19         // Ejecutar el método y verificar si devuelve el valor correcto
20         double a = Cuadratica.ingresarA(0.0);
21         assertEquals(expected: 2.0, a, delta: 0.0001); // Verifica que el valor ingresado sea igual a 2.0
22
23         // Restaurar la entrada estándar original
24         System.setIn(System.in);
25     }
26
27     new *
28     @Test
29     public void calcularCuadraticaTest() {
30         double x1 = Cuadratica.calcularCuadratica1(x: 0, a: 1.0, b: -3.0, c: 2.0, discriminante: 1.0);
31         assertEquals(expected: 2.0, x1, delta: 0.0001); // Verifica que el cálculo de la primera raíz sea correcto
32     }
33 }
```

Joaquin Silva

```
5 public class Calculadora {
30 public static double areaCirculo(double radio) {
31     return Math.PI * radio * radio;
32 }
33
34 public static double areaEsfera(double radio) {
35     return 4 * Math.PI * radio * radio;
36 }
37
38 public static double volumenEsfera(double radio) {
39     return Math.PI * Math.pow(radio, 3) * (double) 4 / 3;
40 }
41
42 public static double areaCubo(double lado) {
43     return 6 * lado * lado;
44 }
45
46 public static double volumenCubo(double lado) {
47     return lado * lado * lado;
48 }
49
50 public static double areaCono(double radio, double altura) {
51     return Math.PI * radio * (radio + (sqrt(radio * radio + altura * altura)));
52 }
53
54 public static double volumenCono(double radio, double altura) {
55     return (Math.PI * altura * radio * radio) / 3;
56 }
57
58 ✓ public static int pedirInt() {
59     Scanner teclado = new Scanner(System.in);
60 }
```

```

public class Calculadora {
    public static void llamarMetodos(int opcion) {
    }

}

public static double ingresarA(double a){
    try{Scanner A=new Scanner(System.in);
    System.out.println("Ingrese la variable a: ");
    a = Double.parseDouble(A.nextLine());
    while (a == 0){
        System.out.println("El valor de a no puede ser 0, ingrese otro valor: ");
        a = Double.parseDouble(A.nextLine());
    }
    return a;}
    catch (NumberFormatException e){
        System.out.println("El valor ingresado no es valido, ingrese otro valor: ");
        return ingresarA(a);
    }
}

public static double ingresarB(double b){
    try{Scanner B=new Scanner(System.in);
    System.out.println("Ingrese la variable b: ");
    b = Double.parseDouble(B.nextLine());
    return b;}
    catch (NumberFormatException e){
        System.out.println("El valor ingresado no es valido, ingrese otro valor: ");
        return ingresarB(b);
    }
}

public static double ingresarC(double c){
    try{Scanner C=new Scanner(System.in);
    System.out.println("Ingrese la variable c: ");
    c = Double.parseDouble(C.nextLine());

```

```

5  class CalculadoraTest {
6
7      @org.junit.jupiter.api.Test
8      void areaEsfera() {
9          assertEquals(Math.PI*4, Calculadora.areaEsfera(1));
10         assertEquals(Math.PI*2, Calculadora.areaEsfera(3));
11     }
12
13     @org.junit.jupiter.api.Test
14     void volumenEsfera() {
15         assertEquals((double) 4 / 3*Math.PI, Calculadora.volumenEsfera(1));
16         assertEquals(Math.PI, Calculadora.volumenEsfera(2));
17     }
18
19     @org.junit.jupiter.api.Test
20     void areaCubo() {
21         assertEquals(24, Calculadora.areaCubo(2));
22         assertEquals(30, Calculadora.areaCubo(3));
23     }
24
25     @org.junit.jupiter.api.Test
26     void volumenCubo() {
27         assertEquals(8, Calculadora.volumenCubo(2));
28         assertEquals(10, Calculadora.volumenCubo(3));
29     }
30
31     @org.junit.jupiter.api.Test
32     void areaCono() {
33         assertEquals(24*Math.PI, Calculadora.areaCono(3,4));
34         assertEquals(30, Calculadora.areaCono(3,3));
35     }

```



```

8      class CalculadoraTest {
83
84          @Test
85          public void ingresarAValidoTest() {
86              // Simular entrada de usuario (por ejemplo, "2.0\n")
87              String input = "hola\n";
88              InputStream in = new ByteArrayInputStream(input.getBytes());
89              System.setIn(in);
90
91              // Ejecutar el método y verificar si devuelve el valor correcto
92              double a = Calculadora.ingresarA(0.0);
93              assertEquals(2.0, a, 0.0001); // Verifica que el valor ingresado sea igual a 2.0
94
95              // Restaurar la entrada estándar original
96              System.setIn(System.in);
97          }
98          @Test
99          public void calcularCuadratica1Test() {
100              double x1 = Calculadora.calcularCuadratica1(0, 1.0, -3.0, 2.0, 1.0);
101              assertEquals(2.0, x1, 0.0001); // Verifica que el cálculo de la primera raíz sea correcto
102          }
103
104          @Test
105          public void calcularCuadratica2Test() {
106              double x2 = Calculadora.calcularCuadratica2(0, 1.0, -3.0, 2.0, 1.0);
107              assertEquals(1.0, x2, 0.0001); // Verifica que el cálculo de la segunda raíz sea correcto
108          }
109
110
111      }

```