

2022

validación y verificación

Ejercicio

Andy Yamada	6481
Jhon Hilasaca	6608
Wellington Cabezas	6683
Jhonnathan Castillo	6588





ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



Problemática: Desarrollo de un programa obtenga las raíces de polinomio de grado 2

Requisitos:

- RF1. *Ingresar por teclado el primer término de la ecuación.*
- RF2. *Ingresar por teclado el segundo término de la ecuación.*
- RF3. *Ingresar por teclado el tercer término de la ecuación.*
- RF4. *Validar el ingreso de cada término, donde no se admite letras o caracteres especiales.*
- RF5. *Validar el ingreso del primer término, que no sea "0"*
- RF6. *Cada término ingresado debe ser un número real.*
- RF7. *El Sistema debe calcular las raíces imaginarias, en caso de que existan.*
- RF8. *El Sistema debe indicar que se calcularon las raíces imaginarias, en el caso de hacerlo*
- RF9. *El Sistema debe calcular las raíces reales, en caso de que existan.*
- RF10. *El Sistema debe indicar que se calcularon las raíces reales, en el caso de hacerlo.*

Código:

Clase base

```
package com.mycompany.raicescuadraticas;
```

```
import java.util.Scanner;
```

```
import java.text.DecimalFormat;
```

```
public class base {
```

```
    double a;
```

```
    double b;
```

```
    double c;
```

```
    double raiz1 =0;
```

```
    double raiz2=0;
```

```
    Scanner lectura = new Scanner (System.in);
```

```
/*
```

```
* comprobador de numeros
```



* */

```
public boolean comprobar_num(String a) {  
    try {  
        Double.parseDouble(a);  
        return true;}  
    catch(NumberFormatException nfe){  
        System.out.println("Dato erroneo ingresar un numero permitido");  
        return false;}  
}  
  
public void constructor(){  
    String cadena;  
    do{  
        do {  
            System.out.println("Ingresar valor 1 (el valor debe ser diferente de 0)");  
            cadena=lectura.next();  
        }while(!comprobar_num(cadena));  
        while(Double.parseDouble(cadena)==0);  
        a = Double.parseDouble(cadena);  
        do {  
            System.out.println("Ingresar valor 2");  
            cadena=lectura.next();  
        }while(!comprobar_num(cadena));  
        b = Double.parseDouble(cadena);  
        do {  
            System.out.println("Ingresar valor 3");  
            cadena=lectura.next();
```



```
    }while(!comprobar_num(cadena));  
  
    c = Double.parseDouble(cadena);  
  
}
```

```
public double discriminante(double a1,double b1,double c1) {  
  
    double r;  
  
    r= ((Math.pow(b1,2))-(4*a1*c1));  
  
    return r;  
  
}
```

```
public void raices(double a1,double b1,double c1) {  
  
    double disc= discriminante(a1, b1, c1);  
  
    if(disc<0){  
  
        raiz1=(b1+Math.sqrt(disc*-1))/(2*a);  
        raiz2=(b1-Math.sqrt(disc*-1))/(2*a);  
  
    }else{  
  
        raiz1=(b1+Math.sqrt(disc))/(2*a);  
        raiz2=(b1-Math.sqrt(disc))/(2*a);  
  
    }  
  
}
```

```
public void resultado(double a1,double b1,double c1) {
```



```
double disc= discriminante(a1, b1, c1);

DecimalFormat decimal=new DecimalFormat ("#.####");

if(disc<0){

    System.out.println("x1="+decimal.format(raiz1)+" i
x2="+decimal.format(raiz2)+" i");

}else{

    System.out.println("x1="+decimal.format(raiz1)+"
x2="+decimal.format(raiz2));

}

}

}
```

Clase main

```
package com.mycompany.raicescuadraticas;

public class RaicesCuadraticas {

    public static void main(String[] args) {

        base arte=new base();

        arte.constructor();

        arte.raices(arte.a, arte.b, arte.c);

        arte.resultado(arte.a, arte.b, arte.c);

    }

}
```



```
}  
  
}
```

Pruebas Unitarias:

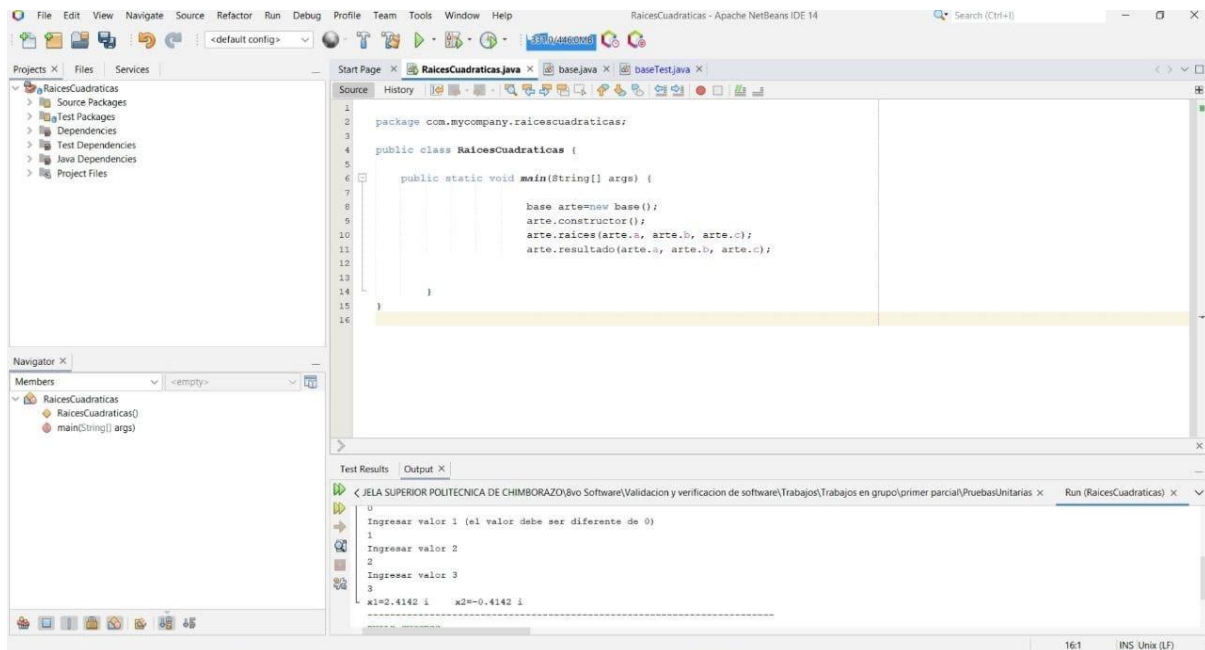
Pruebas Unitarias comprobando RF1

No.casos de prueba: 1

No.1 Ingreso del valor del primer termino

Pruebas exitosas: 1

Observacion: Al momento de la ejecucion se comprueba que el Sistema acepta los valores ingresados



Pruebas Unitarias comprobando RF2

No.casos de prueba: 1

No.1 Ingreso del valor del segundo termino

Pruebas exitosas: 1

Observacion: Al momento de la ejecucion se comprueba que el Sistema acepta los valores ingresados



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



The screenshot shows the Apache NetBeans IDE interface. The main editor displays the `RaicesCuadraticas.java` file with the following code:

```
1 package com.mycompany.raicescuadraticas;
2
3 public class RaicesCuadraticas {
4
5     public static void main(String[] args) {
6
7         base arte=new base();
8         arte.constructor();
9         arte.raices(arte.a, arte.b, arte.c);
10        arte.resultado(arte.a, arte.b, arte.c);
11    }
12
13 }
14
15
16
```

The left sidebar shows the Project Explorer with the following structure:

- Projects X
- Files
- Services
- ▼ RaicesCuadraticas
 - Source Packages
 - Test Packages
 - Dependencies
 - Test Dependencies
 - Java Dependencies
 - Project Files

The bottom pane shows the Test Results and Output. The output window displays the following text:

```
U
Ingresar valor 1 (el valor debe ser diferente de 0)
1
Ingresar valor 2
2
Ingresar valor 3
3
x1=2.4142 i x2=-0.4142 i
```

Pruebas Unitarias comprobando RF3

No.casos de prueba: 1

No.1 Ingreso del valor del tercer termino

Pruebas exitosas: 1

Observacion: Al momento de la ejecucion se comprueba que el Sistema acepta los valores ingresados

This screenshot is identical to the one above, showing the same code and output in the Apache NetBeans IDE.



Validacion si un valor Ingresado es número

No.casos de prueba: 3

No.1 Ingreso de letras

No2.Ingreso de letras y números

No3.Ingreso de unicamente numeros

Pruebas exitosas: 3

Observacion: Al momento de la ejecucion nos dio 2 errores (como se esperaba) porque los 2 primeros casos contienen letras y dichos valores no son posibles.

El ultimo caso es positivo por que solo contienen numeros:

```
baseTest.java
10 oPrueba=new base();
11 };
12
13
14 @Test
15 public void testcomprobar_num1() {
16     escena();
17     String cadena= "z";
18     assertTrue(oPrueba.comprobar_num(cadena));
19 }
20 @Test
21 public void testcomprobar_num2() {
22     escena();
23     String cadena= "12kz";
24     assertTrue(oPrueba.comprobar_num(cadena));
25 }
26 @Test
27 public void testcomprobar_num3() {
28     escena();
29     String cadena= "1235.12";
30     assertTrue(oPrueba.comprobar_num(cadena));
31 }
32
33
34
```

Failure Trace

```
java.lang.AssertionError
    at par.baseTest.testcomprobar_num1(baseTest.java:18)
```

Console

```
<terminated> baseTest [JUnit] C:\Users\USUARIO\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\java.exe (29 sep. 2022 10:10:50 p. m. - 10
Dato erroneo ingresar un numero permitido
Dato erroneo ingresar un numero permitido
```

Pruebas Unitarias comprobando RF6

No.casos de prueba: 1

No.1 Ingreso del valor 0

Pruebas exitosas: 1

Observacion: Al momento de la ejecucion se nota que el sistema no se admite el valor de imaginario al momento de ingresar los valores



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



The screenshot shows an IDE with the following components:

- Source Editor:** Contains the following Java code:

```
1 package com.myccompany.raicescuadraticas;
2
3
4 public class RaicesCuadraticas {
5
6     public static void main(String[] args) {
7
8         base arte=new base();
9         arte.constructor();
10        arte.raices(arte.a, arte.b, arte.c);
11        arte.resultado(arte.a, arte.b, arte.c);
12
13    }
14
15 }
16
```
- Navigator:** Shows the project structure with 'RaicesCuadraticas' and its members 'RaicesCuadraticas()' and 'main(String[] args)'.
- Test Results / Output:** Shows the execution output for 'Run (RaicesCuadraticas)'. The output includes:

```
< JELA SUPERIOR POLITÉCNICA DE CHIMBORAZO\Software\Validación y verificación de software\Trabajos\Trabajos en grupo\primer parcial\Pruebas Unitarias > Run (RaicesCuadraticas) >
Building RaicesCuadraticas 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ RaicesCuadraticas ---
Ingresar valor 1 (el valor debe ser diferente de 0)
31
Dato erroneo ingresar un numero permitido
Ingresar valor 1 (el valor debe ser diferente de 0)
```

Pruebas Unitarias comprobando RF5 (RF1. Validar el ingreso del primer término, que no sea "0")

No.casos de prueba: 1

No.1 Ingreso del valor 0

Pruebas exitosas: 1

Observacion: Al momento de la ejecución se nota que el sistema no se admit el valor de 0 en el primer valor y vuelve a solitar el primer valor



```
1 package com.myccompany.raicescuadraticas;
2
3
4 public class RaicesCuadraticas {
5
6     public static void main(String[] args) {
7
8         base arte=new base();
9         arte.constructor();
10        arte.raices(arte.a, arte.b, arte.c);
11        arte.resultado(arte.a, arte.b, arte.c);
12
13    }
14
15
16 }
```

Test Results Output X

< LA SUPERIOR POLITECNICA DE CHIMBORAZO\Bvo Software(Validacion y verificacion de software)\Trabajos\Trabajos en grupo\primer parcial\PruebasUnitarias x Run (RaicesCuadraticas) x

Building RaicesCuadraticas 1.0-SNAPSHOT

-----[jar]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ RaicesCuadraticas ---

Ingresar valor 1 (el valor debe ser diferente de 0)

0

Ingresar valor 1 (el valor debe ser diferente de 0)

Pruebas Unitarias comprobando el valor del discriminante

No.casos de prueba: 3

No.1 Ingreso de valores para obtener un valor negativo

No2. Ingreso de valores para obtener el elemento 0

No3. Ingreso de valores para obtener un valor positivo

Pruebas exitosas: 3

Observacion: Al momento de la ejecucion no se ejecuto los errores porque se obserba:

cuando el discriminante debe ser negativo Resultado: Correcto

cuando el discriminante es 0 Resultado: Correcto

Cuando el discriminante debe ser positivo Resultado: Correcto



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



```
1 package par;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class baseTest {
8     public base oPruebas;
9     public void escena() {
10         oPrueba=new base();
11     };
12
13     @Test
14     public void testDiscriminante1() {
15         escena();
16         double disc= oPrueba.discriminante(1 , 2, 3);
17         assertEquals(-8,disc,0);
18     }
19
20     @Test
21     public void testDiscriminante2() {
22         escena();
23         double disc= oPrueba.discriminante(1 , 10, 25);
24         assertEquals(0,disc,0);
25     }
26
27     @Test
```

Pruebas Unitarias comprobando el valor de la raiz

No.casos de prueba: 2

No.1 Optener raices imaginarias

No2. Optener raices reales

Pruebas exitosas: 2

Observacion: Al momento de la ejecucion no se ejecuto los errores porque se obserba:

cuando las raices deben ser imaginaries Resultado: Correcto

cuando las raices deben ser reales Resultado: Correcto



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



```
package com.mycompany.raicescuadraticas;

public class RaicesCuadraticas {

    public static void main(String[] args) {

        base arte=new base();
        arte.constructor();
        arte.raices(arte.a, arte.b, arte.c);
        arte.resultado(arte.a, arte.b, arte.c);

    }

}
```

Test Results Output

JELA SUPERIOR POLITECNICA DE CHIMBORAZO\Bvo Software(Validacion y verificacion de software)\Trabajos\Trabajos en grupo\primer parcial\PruebasUnitarias > Run (RaicesCuadraticas) X

Ingresar valor 1 (el valor debe ser diferente de 0)
1
Ingresar valor 2
4
Ingresar valor 3
4
x1=2 x2=2
BUILD SUCCESS

(Raices reales)

```
package com.mycompany.raicescuadraticas;

public class RaicesCuadraticas {

    public static void main(String[] args) {

        base arte=new base();
        arte.constructor();
        arte.raices(arte.a, arte.b, arte.c);
        arte.resultado(arte.a, arte.b, arte.c);

    }

}
```

Test Results Output

JELA SUPERIOR POLITECNICA DE CHIMBORAZO\Bvo Software(Validacion y verificacion de software)\Trabajos\Trabajos en grupo\primer parcial\PruebasUnitarias > Run (RaicesCuadraticas) X

--- exec-maven-plugin:3.0.0:exec (default-cli) @ RaicesCuadraticas ---
Ingresar valor 1 (el valor debe ser diferente de 0)
1
Ingresar valor 2
2
Ingresar valor 3
3
x1=2.4142 i x2=-0.4142 i

(Raices Imaginarias)