

Verificarea și Testarea Sistemelor de Calcul

Temă de casă

November 19, 2018

Titlu: *Testarea Empirică*

Profesor: Ș.l. Dr. Ing. Nicolae Enescu

Student: Voiculescu Ioan-Valentin

Facultate: Automatică, Calculatoare și Electronică

Anul: IV

Specializarea: Calculatoare Română

Grupa: CR 4.H1 A

Contents

| | | |
|----------|--------------------------------------|----------|
| 1 | Codul Sursă^[1] | 3 |
| 1.1 | Implementarea | 4 |
| 1.1.1 | EmpiricalTestingConsole.cs | 4 |
| 1.1.2 | Program.cs | 5 |
| 1.1.3 | QuadraticEquation.cs | 7 |
| 1.1.4 | Coefficients.cs | 11 |
| 1.1.5 | Solution.cs | 13 |
| 1.1.6 | DecimalRound.cs | 14 |
| 1.1.7 | Constants.cs | 16 |
| 1.1.8 | InputProcessing.cs | 17 |
| 1.1.9 | OutputProcessing.cs | 21 |
| 1.1.10 | Input.cs | 23 |
| 1.1.11 | Output.cs | 23 |

1 Codul Sursă^[1]

1.1 Implementarea

1.1.1 EmpiricalTestingConsole.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public static class EmpiricalTestingConsole
10     {
11         private static Output _output;
12         static EmpiricalTestingConsole()
13         {
14             _output = new Output();
15         }
16         public static void Write(string[] lines)
17         {
18             _output.WriteData(lines);
19         }
20     }
21 }
```

1.1.2 Program.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             EmpiricalTestingConsole.Write(new string
14             [1] { "static void Main(string[] args)"
15             });
16
17             try
18             {
19                 EmpiricalTestingConsole.Write(new
20                 string[1] { "try" });
21                 InputProcessing inputProcessing = new
22                 InputProcessing(new Input(), new
23                 Output());
24                 EmpiricalTestingConsole.Write(new
25                 string[1] { "InputProcessing
26                 inputProcessing = new
27                 InputProcessing(new Input(), new
28                 Output())" });
29                 EmpiricalTestingConsole.Write(new
30                 string[1] { "inputProcessing=" +
31                 inputProcessing.ToString() });
32                 OutputProcessing outputProcessing =
33                 new OutputProcessing(new Output());
34                 EmpiricalTestingConsole.Write(new
35                 string[1] { "OutputProcessing
36                 outputProcessing = new
37                 OutputProcessing(new Output())" });
38                 EmpiricalTestingConsole.Write(new
39                 string[1] { "outputProcessing=" +
40                 outputProcessing.ToString() });
41                 QuadraticEquation quadraticEquation =
42                 new QuadraticEquation(
43                 inputProcessing.GetData());
44                 EmpiricalTestingConsole.Write(new
45                 string[1] { "QuadraticEquation
```

```

        quadraticEquation = new
        QuadraticEquation(inputProcessing.
        GetData())" });
25      EmpiricalTestingConsole.Write(new
        string[1] { "quadraticEquation=" +
        quadraticEquation.ToString() });
26      outputProcessing.PutData(
        quadraticEquation.
        SolveWithRealSolutions());
27      EmpiricalTestingConsole.Write(new
        string[1] { "outputProcessing.
        PutData(quadraticEquation.
        SolveWithRealSolutions())" });
28    }
29    catch (Exception e)
30    {
31      EmpiricalTestingConsole.Write(new
        string[1] { "catch (Exception e)"
        });
32      Console.WriteLine(e.Message);
33      EmpiricalTestingConsole.Write(new
        string[1] { "Console.WriteLine(e.
        Message)" });
34    }
35  }
36 }
37 }

```

1.1.3 QuadraticEquation.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class QuadraticEquation
10     {
11         //ax^2 + bx + c = 0
12         public Coefficients Coefficients { private set
13             ; get; }
14         public double Delta
15         {
16             get
17             {
18                 EmpiricalTestingConsole.Write(new
19                     string[2] { "public double Delta",
20                         "get" });
21                 return (double)(Math.Pow(Coefficients.
22                     B, 2) - 4 * Coefficients.A *
23                     Coefficients.C);
24             }
25         }
26     }
27     public QuadraticEquation (Coefficients
28         coefficients)
29     {
30         EmpiricalTestingConsole.Write(new string
31             [1] { "public QuadraticEquation (
32                 Coefficients coefficients)" });
33         EmpiricalTestingConsole.Write(new string
34             [1] { "coefficients=" + coefficients.
35                 ToString() });
36         EmpiricalTestingConsole.Write(new string
37             [3] { "A=" + coefficients.A, "B=" +
38                 coefficients.B, "C=" + coefficients.C
39             });
40         if (coefficients != null)
41         {
42             EmpiricalTestingConsole.Write(new
43                 string[1] { "if (coefficients !=
44                     null)" });
45             Coefficients = coefficients;
46         }
47     }
48 }
```

```

30         EmpiricalTestingConsole.Write(new
            string[1] { "Coefficients =
                coefficients" });
31         EmpiricalTestingConsole.Write(new
            string[1] { "Coefficients=" +
                Coefficients.ToString() });
32         EmpiricalTestingConsole.Write(new
            string[3] { "A=" + Coefficients.A,
                "B=" + Coefficients.B, "C=" +
                Coefficients.C });
33     }
34     else
35     {
36         EmpiricalTestingConsole.Write(new
            string[1] { "else" });
37         throw new Exception("'coefficients'
            cannot be null");
38     }
39 }
40 public Solution SolveWithRealSolutions()
41 {
42     EmpiricalTestingConsole.Write(new string
        [1] { "public Solution
            SolveWithRealSolutions()" });
43     Solution solution = null;
44     EmpiricalTestingConsole.Write(new string
        [1] { "Solution solution = null" });
45     if (Delta.Equals(0))
46     {
47         EmpiricalTestingConsole.Write(new
            string[1] { "if (Delta.Equals(0))"
                });
48         solution = new Solution(
49             (double)(-Coefficients.B) / (
                double)(2 * Coefficients.A),
50             (double)(-Coefficients.B) / (
                double)(2 * Coefficients.A)
51         );
52         EmpiricalTestingConsole.Write(new
            string[4]
53         {
54             "solution = new Solution(",
55             "(double)(-Coefficients.B) / (
                double)(2 * Coefficients.A)",
56             "(double)(-Coefficients.B) / (
                double)(2 * Coefficients.A)",

```



```

57         ")"
58     });
59     EmpiricalTestingConsole.Write(new
        string[1] { "solution=" + solution.
            ToString() });
60     EmpiricalTestingConsole.Write(new
        string[2] { "solution.Root1=" +
            solution.Root1.ToString(), "
            solution.Root2=" + solution.Root2.
            ToString() });
61 }
62 if (Delta > 0)
63 {
64     EmpiricalTestingConsole.Write(new
        string[1] { " if(Delta > 0)" });
65     solution = new Solution(
66         (double)(-Coefficients.B + Math.
            Sqrt(Delta)) / (double)(2 *
            Coefficients.A),
67         (double)(-Coefficients.B - Math.
            Sqrt(Delta)) / (double)(2 *
            Coefficients.A)
68     );
69     EmpiricalTestingConsole.Write(new
        string[4]
70     {
71         "solution = new Solution(",
72         "(double)(-Coefficients.B + Math.
            Sqrt(Delta)) / (double)(2 *
            Coefficients.A)",
73         "(double)(-Coefficients.B - Math.
            Sqrt(Delta)) / (double)(2 *
            Coefficients.A)",
74         ")"
75     });
76     EmpiricalTestingConsole.Write(new
        string[1] { "solution=" + solution.
            ToString() });
77     EmpiricalTestingConsole.Write(new
        string[2] { "solution.Root1=" +
            solution.Root1.ToString(), "
            solution.Root2=" + solution.Root2.
            ToString() });
78 }
79 if (Delta < 0)
80 {

```

```
81         EmpiricalTestingConsole.Write(new
            string[1] { " if (Delta < 0)" });
82         throw new Exception("the equation do
            not have a real solution");
83     }
84     return solution;
85 }
86 }
87 }
```

1.1.4 Coefficients.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class Coefficients
10     {
11         public int A { private set; get; }
12         public int B { private set; get; }
13         public int C { private set; get; }
14         public Coefficients(int coefficientA, int
15             coefficientB, int coefficientC)
16         {
17             EmpiricalTestingConsole.Write(new string
18                 [1] {"public Coefficients(int
19                     coefficientA, int coefficientB, int
20                     coefficientC)"});
21             EmpiricalTestingConsole.Write(new string
22                 [1] { "coefficientA=" + coefficientA.
23                     ToString() });
24             EmpiricalTestingConsole.Write(new string
25                 [1] { "coefficientB=" + coefficientB.
26                     ToString() });
27             EmpiricalTestingConsole.Write(new string
28                 [1] { "coefficientB=" + coefficientB.
29                     ToString() });
30             if (coefficientA == 0)
31             {
32                 EmpiricalTestingConsole.Write(new
33                     string[1] { "if (coefficientA == 0)
34                         " });
35                 throw new Exception("The 'a'
36                     coefficient cannot equal 0");
37             }
38             A = coefficientA;
39             EmpiricalTestingConsole.Write(new string
40                 [1] { "A = coefficientA" });
41             EmpiricalTestingConsole.Write(new string
42                 [1] { "A=" + A.ToString()});
43             B = coefficientB;
44             EmpiricalTestingConsole.Write(new string
```

```

30         [1] { "B = coefficientB" });
        EmpiricalTestingConsole.Write(new string
31         [1] { "B=" + B.ToString() });
        C = coefficientC;
32        EmpiricalTestingConsole.Write(new string
        [1] { "C = coefficientC" });
33        EmpiricalTestingConsole.Write(new string
        [1] { "C=" + C.ToString() });
34    }
35 }
36 }

```

1.1.5 Solution.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class Solution
10     {
11         public double Root1 { private set; get; }
12         public double Root2 { private set; get; }
13         public Solution(double root1, double root2)
14         {
15             EmpiricalTestingConsole.Write(new string
16                 [1] { "public Solution(double root1,
17                     double root2)" });
18             EmpiricalTestingConsole.Write(new string
19                 [2] { "root1=" + root1.ToString(), "
20                     root2=" + root2.ToString() });
21             Root1 = root1;
22             EmpiricalTestingConsole.Write(new string
23                 [1] { "Root1 = root1" });
24             EmpiricalTestingConsole.Write(new string
25                 [1] { "Root1=" + Root1.ToString() });
26             Root2 = root2;
27             EmpiricalTestingConsole.Write(new string
28                 [1] { "Root2 = root2" });
29             EmpiricalTestingConsole.Write(new string
30                 [1] { "Root2=" + Root2.ToString() });
31         }
32     }
33 }
```

1.1.6 DecimalRound.cs

```
1 using System;
2
3 namespace Lab1
4 {
5     public static class Decimals
6     {
7         public static string[] Round(Solution solution
8             , int numberDecimals)
9         {
10             EmpiricalTestingConsole.Write(new string
11                 [1] { "public static string[] Round(
12                     Solution solution, int numberDecimals)"
13                 });
14             EmpiricalTestingConsole.Write(new string
15                 [1] { "solution=" + solution.ToString()
16                 });
17             EmpiricalTestingConsole.Write(new string
18                 [2] { "solution.Root1=" + solution.
19                     Root1.ToString(), "solution.Root2=" +
20                     solution.Root2.ToString() });
21             EmpiricalTestingConsole.Write(new string
22                 [1] { "numberDecimals=" +
23                     numberDecimals.ToString() });
24             string[] roots;
25             EmpiricalTestingConsole.Write(new string
26                 [1] { "string[] roots" });
27             roots = new string[2]
28             {
29                 (Math.Truncate(Math.Pow(10,
30                     numberDecimals) * solution.Root1) /
31                     (double) Math.Pow(10,
32                     numberDecimals)).ToString(),
33                 (Math.Truncate(Math.Pow(10,
34                     numberDecimals) * solution.Root2) /
35                     (double) Math.Pow(10,
36                     numberDecimals)).ToString(),
37             };
38             EmpiricalTestingConsole.Write(new string
39                 [5]
40             {
41                 "roots = new string[2]",
42                 "{",
43                 "(Math.Truncate(Math.Pow(10,
44                     numberDecimals) * solution.Root1) /
```

```

25         (double) Math.Pow(10,
            numberDecimals)).ToString(),",
        "(Math.Truncate(Math.Pow(10,
            numberDecimals) * solution.Root2) /
            (double) Math.Pow(10,
            numberDecimals)).ToString(),",
26         "};"
27
28     });
29     EmpiricalTestingConsole.Write(new string
        [2] { "roots[0]=" + roots[0], "roots
            [1]=" + roots[1] });
30     return roots;
31 }
32 }
33 }

```

1.1.7 Constants.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public static class Constants
10     {
11         public const int CoefficientAMinimumValue =
12             -30;
13         public const int CoefficientAMaximumValue =
14             70;
15
16         public const int CoefficientBMinimumValue =
17             -50;
18         public const int CoefficientBMaximumValue =
19             10;
20
21         public const int CoefficientCMinimumValue = 0;
22         public const int CoefficientCMaximumValue =
23             200;
24     }
25 }
```


1.1.8 InputProcessing.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Linq.Expressions;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Lab1
9  {
10     public class InputProcessing
11     {
12         private Input _input;
13         private Output _output;
14         public InputProcessing(Input input, Output
            output)
15         {
16             EmpiricalTestingConsole.Write(new string
                [1] { "public InputProcessing(Input
                    input, Output output)" });
17             EmpiricalTestingConsole.Write(new string
                [2] { "input=" + input.ToString(), "
                    output=" + output.ToString() });
18             _input = input;
19             EmpiricalTestingConsole.Write(new string
                [1] { " _input = input" });
20             EmpiricalTestingConsole.Write(new string
                [1] { " _input=" + _input.ToString() });
21             _output = output;
22             EmpiricalTestingConsole.Write(new string
                [1] { " _output = output" });
23             EmpiricalTestingConsole.Write(new string
                [1] { " _output=" + _output.ToString()
                    });
24         }
25         public Coefficients GetData()
26         {
27             EmpiricalTestingConsole.Write(new string
                [1] { "public Coefficients GetData()"
                    });
28             Coefficients coefficients = new
                Coefficients(
29                 SetCoefficient('a', Constants.
                    CoefficientAMinimumValue, Constants
                    .CoefficientAMaximumValue),
```

```

30         SetCoefficient('b', Constants.
           CoefficientBMinimumValue, Constants
           .CoefficientBMaximumValue),
31         SetCoefficient('c', Constants.
           CoefficientCMinimumValue, Constants
           .CoefficientCMaximumValue)
32     );
33     EmpiricalTestingConsole.Write(new string
34     [5]
35     {
36         "Coefficients coefficients = new
           Coefficients(",
37         "SetCoefficient('a', Constants.
           CoefficientAMinimumValue, Constants
           .CoefficientAMaximumValue)",
38         "SetCoefficient('b', Constants.
           CoefficientBMinimumValue, Constants
           .CoefficientBMaximumValue)",
39         "SetCoefficient('c', Constants.
           CoefficientCMinimumValue, Constants
           .CoefficientCMaximumValue)",
40         ")",
41     });
42     EmpiricalTestingConsole.Write(new string
43     [1] { "coefficients=" + coefficients.
44     ToString() });
45     EmpiricalTestingConsole.Write(new string
46     [3]
47     {
48         "coefficients.A=" + coefficients.A,
49         "coefficients.B=" + coefficients.B,
50         "coefficients.C=" + coefficients.C
51     });
52     return coefficients;
53 }
54 private int SetCoefficient(char c, int
55     minimumValue, int maximumValue)
56 {
57     EmpiricalTestingConsole.Write(new string
58     [1] { "private int SetCoefficient(char
59     c, int minimumValue, int maximumValue)"
60     });
61     EmpiricalTestingConsole.Write(new string
62     [3]
63     {
64         "c=" + c,

```

```

56         "minimumValue=" + minimumValue.
           ToString(),
57         "maximumValue=" + maximumValue.
           ToString()
58     });
59     int coefficient;
60     EmpiricalTestingConsole.Write(new string
        [1] { "int coefficient" });
61     if (_output != null)
62     {
63         EmpiricalTestingConsole.Write(new
            string[1] { "if (_output != null)"
        });
64         string line = c + "=";
65         EmpiricalTestingConsole.Write(new
            string[1] { "string line = c + " +
            "=" });
66         EmpiricalTestingConsole.Write(new
            string[1] { "line=" + line });
67         _output.WriteData(new string[1] {line
        });
68         EmpiricalTestingConsole.Write(new
            string[1] { "_output.WriteData(new
            string[1] {line})" });
69     }
70     try
71     {
72         EmpiricalTestingConsole.Write(new
            string[1] { "try" });
73         coefficient = Int32.Parse(_input.
            ReadDataByLine());
74         EmpiricalTestingConsole.Write(new
            string[1] { "coefficient = Int32.
            Parse(_input.ReadDataByLine())" });
75         EmpiricalTestingConsole.Write(new
            string[1] { "coefficient=" +
            coefficient });
76     }
77     catch (FormatException)
78     {
79         EmpiricalTestingConsole.Write(new
            string[1] { "catch (FormatException
            )" });
80         throw new Exception(c+" is not integer
            ");
81     }

```

```

82         if(!(coefficient>=minimumValue &&
83             coefficient<=maximumValue))
84         {
85             EmpiricalTestingConsole.Write(new
86                 string[1] { "if(!(coefficient>=
87                     minimumValue && coefficient<=
88                     maximumValue))" });
89             throw new Exception(c + " out of range
90                 ");
91         }
92     }
93 }

```

1.1.9 OutputProcessing.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Lab1
8  {
9      public class OutputProcessing
10     {
11         private Output _output;
12         public OutputProcessing(Output output)
13         {
14             EmpiricalTestingConsole.Write(new string
15                 [1] { "public OutputProcessing(Output
16                     output)" });
17             _output = output;
18             EmpiricalTestingConsole.Write(new string
19                 [1] { "_output = output" });
20             EmpiricalTestingConsole.Write(new string
21                 [1] { "_output=" + output.ToString() })
22             ;
23         }
24         public void PutData(Solution solution)
25         {
26             EmpiricalTestingConsole.Write(new string
27                 [1] { "public void PutData(Solution
28                     solution)" });
29             EmpiricalTestingConsole.Write(new string
30                 [1] { "solution=" + solution.ToString()
31                     });
32             EmpiricalTestingConsole.Write(new string
33                 [2] { "solution.Root1=" + solution.
34                     Root1.ToString(), "solution.Root2=" +
35                     solution.Root2.ToString() });
36             string line1, line2;
37             EmpiricalTestingConsole.Write(new string
38                 [1] { "string line1, line2" });
39             string[] newSolution = Decimals.Round(
40                 solution, 2);
41             EmpiricalTestingConsole.Write(new string
42                 [1] { "string[] newSolution = Decimals.
43                     Round(solution, 2)" });
44             EmpiricalTestingConsole.Write(new string
```

```

[2] { "newSolution[0]=" + newSolution
[0], "newSolution[1]=" + newSolution[1]
});
29 line1 = "x1=" + newSolution[0];
30 EmpiricalTestingConsole.Write(new string
[1] { "line1 = "+"x1 = "+" +
newSolution[0]" });
31 EmpiricalTestingConsole.Write(new string
[1] { "line1=" + line1 });
32 line2 = "x2=" + newSolution[1];
33 EmpiricalTestingConsole.Write(new string
[1] { "line2 = "+"x2 = "+" +
newSolution[1]" });
34 EmpiricalTestingConsole.Write(new string
[1] { "line2=" + line2 });
35 _output.WriteData(new string[2] {line1,
line2});
36 EmpiricalTestingConsole.Write(new string
[1] { "_output.WriteData(new string[2]
{line1, line2})" });
37 }
38 }
39 }

```

1.1.10 Input.cs

```
1 using System;
2
3 namespace Lab1
4 {
5     public class Input
6     {
7         public virtual string ReadDataByLine()
8         {
9             EmpiricalTestingConsole.Write(new string
10                [1] { "public virtual string
11                    ReadDataByLine()" });
12             return Console.ReadLine();
13         }
14     }
15 }
```

1.1.11 Output.cs

```
1 using System;
2
3 namespace Lab1
4 {
5     public class Output
6     {
7         public virtual void WriteData(string[] lines)
8         {
9             foreach (var line in lines)
10             {
11                 Console.WriteLine(line);
12             }
13         }
14     }
15 }
```

References

- [1] [https://github.com/vioan12/
Verificarea-si-Testarea-Sistemelor-de-Calcul](https://github.com/vioan12/Verificarea-si-Testarea-Sistemelor-de-Calcul)