Politechnika
Wrocławska

Wydział Informatyki i
Telekomunikacji

Platformy programistyczne .Net i Java - LAB1-2

Politechnika Wrocławska

Wydział Informatyki i telekomunikacji

Kierunek: Informatyczne systemy automatyki

grupa nr 2

github.com/wernexnrs/264254-.NET-i-Java

Dawid Popławski - 264254

Termin zajęc: Środa godz. $17^{\underline{05}}$ - $18^{\underline{45}}$

Prowadzący: mgr inż. Michał Jaroszczuk

Spis treści

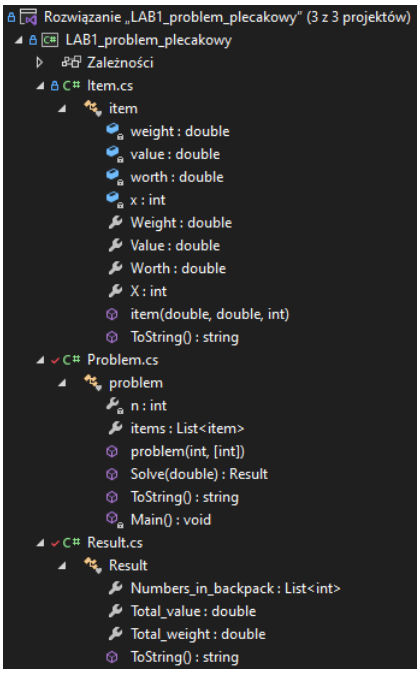
Opis projektu

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tempus nisl enim, venenatis rutrum mauris molestie in. Duis pretium et mi vitae pretium. Nullam malesuada nunc ac lectus viverra, a euismod purus iaculis. Aenean et nisi sit amet eros tristique mollis. Mauris sed turpis ex. In in odio sit amet nisl varius varius. Aliquam faucibus ante non risus vulputate, ornare commodo quam commodo.

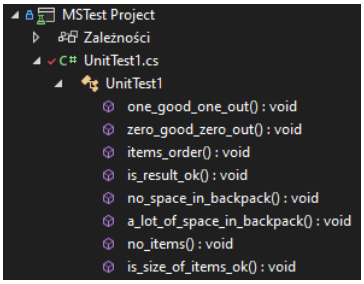
Aliquam et faucibus elit, eget pulvinar enim. Phasellus fermentum lorem eget leo vehicula, eu euismod justo convallis. Vestibulum molestie faucibus nunc eu efficitur. Mauris egestas orci lorem, eu dapibus erat volutpat ut. Vestibulum tincidunt pharetra mi, id scelerisque est dictum ut. In euismod aliquam turpis, at elementum odio pulvinar eu. Duis sit amet lorem aliquam, mollis mi quis, tincidunt ante. Sed congue eu ipsum bibendum rutrum. Cras tincidunt vulputate dui, quis elementum felis accumsan ut. Aenean a pellentesque erat, nec rhoncus ex. Nam tempor fermentum sollicitudin.

Praesent maximus finibus purus at blandit. Aliquam pellentesque dapibus libero vel volutpat. Donec venenatis tortor erat, a consequat eros aliquet vitae. In consequat ornare leo, et tristique sem commodo mollis. Nunc et tellus tellus. In mollis euismod egestas. Integer ante tellus, varius sit amet tempus quis, efficitur quis nulla. Cras eu augue nec diam lacinia porttitor. Fusce lacinia ligula libero, ac placerat augue sollicitudin vitae. Integer sed rhoncus magna, a rutrum ex.

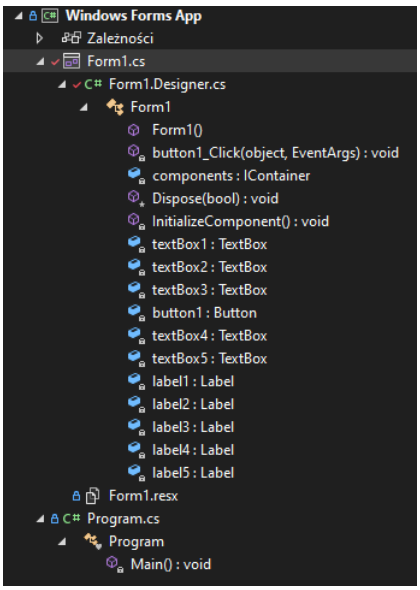
Drzewa projektów



Rys. 2.1: *Drzewo projektu aplikacji konsolowej.*



Rys. 2.2: *Drzewo projektu testów jednostkowych.*



Rys. 2.3: *Drzewo projektu aplikacji okienkowej.*

Kluczowe części kodu

```
Odwołania: 10 | 7/7 sprawdzanie
public Result Solve(double capacity)
{
    items = [.. items.OrderByDescending(item => item.Worth)];
    Result result = new();
    foreach (item element in items)
    {
        if (element.Weight <= capacity)
        {
            element.X = 1;
            capacity -= element.Weight;
            result.Total_value += element.Value;
            result.Total_weight += element.Weight;
            result.Numbers_in_backpack.Add(items.IndexOf(element));
        }
        if (capacity < items.Min(item => item.Weight)) break;
    }
    return result;
}
```

Rys. 3.1: *Kod rozwiązujący problem plecakowy.*

```
using LAB1_problem_plecakowy;

namespace Windows_Forms_App
{
    Odwołania: 3
    public partial class Form1 : Form
    {
        1 odwołanie
        public Form1()
        {
            InitializeComponent();
        }

        1 odwołanie
        private void button1_Click(object sender, EventArgs e)
        {
            bool textBox1Valid = int.TryParse(textBox1.Text, out int n);
            textBox1.BackColor = textBox1Valid ? System.Drawing.Color.Green : System.Drawing.Color.Red;

            bool textBox2Valid = int.TryParse(textBox2.Text, out int seed);
            textBox2.BackColor = textBox2Valid ? System.Drawing.Color.Green : System.Drawing.Color.Red;

            bool textBox3Valid = int.TryParse(textBox3.Text, out int capacity);
            textBox3.BackColor = textBox3Valid ? System.Drawing.Color.Green : System.Drawing.Color.Red;

            if (!textBox1Valid || !textBox2Valid || !textBox3Valid)
                return;

            problem mc = new(n, seed);
            Result result = mc.Solve(capacity);
            Console.WriteLine(mc.ToString());
            Console.WriteLine(result.ToString());

            textBox4.Text = result.ToString();
            textBox5.Text = mc.ToString();
        }
    }
}
```

Rys. 3.2: *Kod aplikacji okienkowej.*

Zrealizowanie testy jednostkowe:

- Sprawdzenie, czy jeśli co najmniej jeden przedmiot spełnia ograniczenia, to zwrócono co najmniej jeden element.
- Sprawdzenie, czy jeśli żaden przedmiot nie spełnia ograniczeń, to zwrócono puste rozwiązanie.
- Sprawdzenie, czy kolejność przedmiotów ma wpływa na znalezione rozwiązanie.
- Sprawdzenie poprawności wyniku dla konkretnej instancji.
- Sprawdzenie, czy jeżeli plecak nie ma miejsca to nie będzie w nim przedmiotów.
- Sprawdzenie, czy jeżeli plecak ma bardzo dużo miejsca to wszystkie przedmioty do niego trafiły.
- Sprawdzenie, czy algorytm działa gdy nie ma żadnych przedmiotów.
- Sprawdzenie, czy ilość wylosowanych przedmiotów zgadza się z ilością obiektów w liście.

```
[TestMethod]
| Odwołania: 0
public void one_good_one_out()
{
    /*Sprawdzenie, czy jeśli co najmniej jeden
     przedmiot spełnia ograniczenia,
     to zwrócono co najmniej jeden element.*/

    problem problem = new(15,2);
    int capacity = 10;
    Result result = problem.Solve(capacity);
    Debug.Assert(problem.items.Any(item => item.Weight <= capacity) && result.Numbers_in_backpack.Count >= 1);
}
[TestMethod]
| Odwołania: 0
public void zero_good_zero_out()
{
    /*Sprawdzenie, czy jeśli
     żaden przedmiot nie spełnia ograniczeń,
     to zwrócono puste rozwiązanie.*/

    problem problem = new(15, 2);
    double capacity = 0.0001;
    Result result = problem.Solve(capacity);
    Debug.Assert(problem.items.All(item => item.Weight > capacity) && result.Numbers_in_backpack.Count == 0);

}
[TestMethod]
| Odwołania: 0
public void items_order()
{
    /*Sprawdzenie, czy kolejność przedmiotów
     ma wpływa na znalezione rozwiązanie.*/
    double capacity = 10;
    problem problem = new(15, 2);
    Result result = problem.Solve(capacity);
    problem problem_reversed = problem;
    problem_reversed.items.OrderBy(item => item.Weight);
    Result result_reversed = problem_reversed.Solve(capacity);
    Debug.Assert(result_reversed != result);
}
```

Rys. 3.3: *Testy jednostkowe.*

```
[TestMethod]
| Odwołania: 0
public void is_result_ok()
{
    /*Sprawdzenie poprawności wyniku
     dla konkretnej instancji.*/

    double capacity = 10;
    List<item> items =[
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0),
    new item(1, 1, 0)];

    problem problem_test = new(8, 0){ items = items};

    Result result = problem_test.Solve(capacity);
    List<int> good_result = [0, 1, 2, 3, 4, 5, 6, 7];
    CollectionAssert.AreEqual(good_result, result.Numbers_in_backpack);
    Assert.AreEqual(8, result.Total_value);
    Assert.AreEqual(8, result.Total_weight);
}

[TestMethod]
| Odwołania: 0
public void no_space_in_backpack()
{
    /*Sprawdzenie, czy jeżeli plecak nie ma miejsca to nie będzie w nim przedmiotów.*/

    problem problem = new(15, 2);
    double capacity = 0;
    Result result = problem.Solve(capacity);
    Debug.Assert(result.Numbers_in_backpack.Count == 0);
}

[TestMethod]
| Odwołania: 0
public void a_lot_of_space_in_backpack()
{
    /*Sprawdzenie, czy jeżeli plecak ma bardzo dużo miejsca to wszystkie przedmioty do niego trafiły.*/

    problem problem = new(1000, 2);
    double capacity = int.MaxValue;
    Result result = problem.Solve(capacity);
    Debug.Assert(result.Numbers_in_backpack.Count == 1000);
}
```

Rys. 3.4: *Testy jednostkowe.*

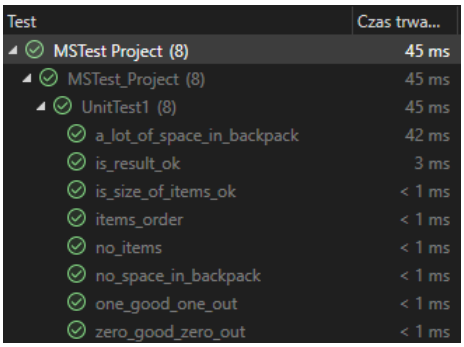
```
[TestMethod]
| Odwołania: 0
public void no_items()
{
    /*Sprawdzenie, czy algorytm działa gdy nie ma żadnych przedmiotów.*/

    problem problem = new(0, 2);
    double capacity = 20;
    Result result = problem.Solve(capacity);
    Debug.Assert(result.Numbers_in_backpack.Count == 0 && result.Total_weight == 0  && result.Total_value ==0);
}

[TestMethod]
| Odwołania: 0
public void is_size_of_items_ok()
{
    /*Sprawdzenie, czy ilość wylosowanych przedmiotów zgadza się z ilością obiektów w liście.*/

    problem problem = new(15, 2);
    Debug.Assert(problem.items.Count == 15);
}
```

Rys. 3.5: *Testy jednostkowe.*



Rys. 3.6: *Wynik testów jednostkowych.*