# [.NET][CZ 15:15][1] Problem Plecakowy

Bartosz Włodarczyk 275470

2025-03-20 15:15 : 16:55

Link repozytorium na: GITHUB Problem Plecakowy.

## 1 Opis projektu

Program składa się z trzech klas, reprezentujących pojedynczy przedmiot (`class Przedmiot`), plecak zabranych przedmiotów (`class Result`) oraz wszystkie przedmioty (`class Problem`).
Program w metodzie `Solve` rozwiązuje problem problem plecakowy algorytmem zachłannym, tzn. sortuje przedmioty od naj-$\frac{\text{cena}}{\text{masa}}$-szego, następnie iterują po każdym elemencie sprawdzając czy się zmieści.
Do programu zrobiono testy oraz graficzne GUI.

### 1.1 Opis najważniejszych klas/metod

#### 1.1.1 `class Przedmiot` Pojedynczy przedmiot

```
12          public int nr_przedmiotu {get;}    // Numer przedmiotu
13          public int waga {get;}             // Waga pojedynczego przedmiotu
14          public int wartosc {get;}          // Wartość pojedynczego przedmiotu
15          public decimal Wartosc_Masa {get;} // "cenność" Wartość / Masa
```

Konstruktor inicjalizuje składowe klasy, istnieje metoda `ToString()`.

#### 1.1.2 `class Result` Plecak zabranych przedmiotów

```
42          public List<int> ListaPrzedmiotow {get;} // Lista id spakowanych przedmiotów
43          private int SumaWartosci;           // Suma wartości spakowanych przedmiotów
44          private int SumaMas;                // Suma mas      spakowanych przedmiotów
45          public int get_SumaMas() {return SumaMas;}      // Pobiera wartość w testach
46          private int Pojemnosc;     // Maksymalna suma mas jaka zmieści się w plecaku
```

Konstruktor inicjalizuje składowe klasy, istnieje metoda `ToString()` oraz osobna dla GUI metoda `ToThreeStrings()` (GUI nie czytało znaków nowej linii, funkcja zwraca tablice z wartościami jako kolejne linie.).

```
55          public bool DodajPrzedmiot(int nr, int wartosc, int masa)
```

Dodaje przedmioty do listy, sprawdzając pojemność oraz aktualizując składowe klasy.

#### 1.1.3 `class Problem` Wszystkie przedmioty

```
92          private int liczba_przedmiotow;    // Liczba wszystkich wygenerowanych przedmiotów
93          public List<Przedmiot> ListaPrzedmiotow { get; } // Lista w kolejności generowania
94          public List<Przedmiot> lista_sort { get; set; }  // Lista w kolejności "cenność"
95
96          public Result Solve(int capacity)
```

Konstruktor inicjalizuje składowe klasy, przypisuje wartości w zakresie $[0; 20]$ (dla masy, wartości) do kolejnych przedmiotów. Klasa posiada metodę `ToString()` oraz osobną dla GUI metodę `ToFewStrings()` (Umożliwia wyświetlanie nowej linii).

#### 1.1.4 `public Result Solve(int capacity)`

Metoda klasy `Problem` tworząca plecak (`class Result`), sortująca po cenności przedmioty, na koniec próbuje dodać do plecaka każdy przedmiot (metoda `DodajPrzedmiot`). Zwraca wypełniony plecak (`class Result`).
Cały kod w sekcji: 3.1.

### 1.1.5 `internal class Program`

Klasa `internal class Program` → odpowiada za komunikacje z użytkownikiem.
Wczytuje `ilosc_przedmiotow` oraz `seed`, do inicjalizacji klasy `Problem`.
Wypisuje wszystkie przedmioty praz pyta o wielkość plecaka. Kończy wypusjąc zawartość plecka.

### 1.1.6 `public class UnitTest1` Testy

Klasa `public class UnitTest1` → zawiera wszystkie testy.
Czy poprawnie wylicza cenność przedmiotów

```
7         [TestMethod]
8         public void TestMethod1()
```

I Sprawdzenie, czy jeśli co najmniej jeden przedmiot spełnia ograniczenia, to zwrócono co najmniej jeden element

```
16        public void TEST01 ()
```

II Sprawdzenie, czy jeśli żaden przedmiot nie spełnia ograniczeń, to zwrócono puste rozwiązanie.

```
29        public void TEST02 ()
```

III Sprawdzenie poprawności wyniku dla konkretnej instancji.
1. Czy jest jakiś przedmiot który by się jeszcze zmieścił ale go nie dodano
2. Czy poprawnie sumuje
3. Czy Suma zapakowanych przedniotów się zmieściłą do plecaka

```
48        public void TEST03 ()
```

IV Sprawdzanie czyj są uszeregowane od naj cenniejszych/masa do najmniej.

```
80        public void TEST04 ()
```

V Czy dodaje odpowiednią ilość przedmiotów

```
103       public void TEST05 ()
```

### 1.1.7 `public partial class Form1 : Form` GUI

Klasa `public partial class Form1 : Form` → zawiera kod tworzący GUI.
`Label` to nie edytowalny napis w GUI, opis pola edytowalnego `TextBox`. `Button` to pole wykrywające wciśnięcia. `ListBox` to nieedytowalne "duże" pole tekstowe.

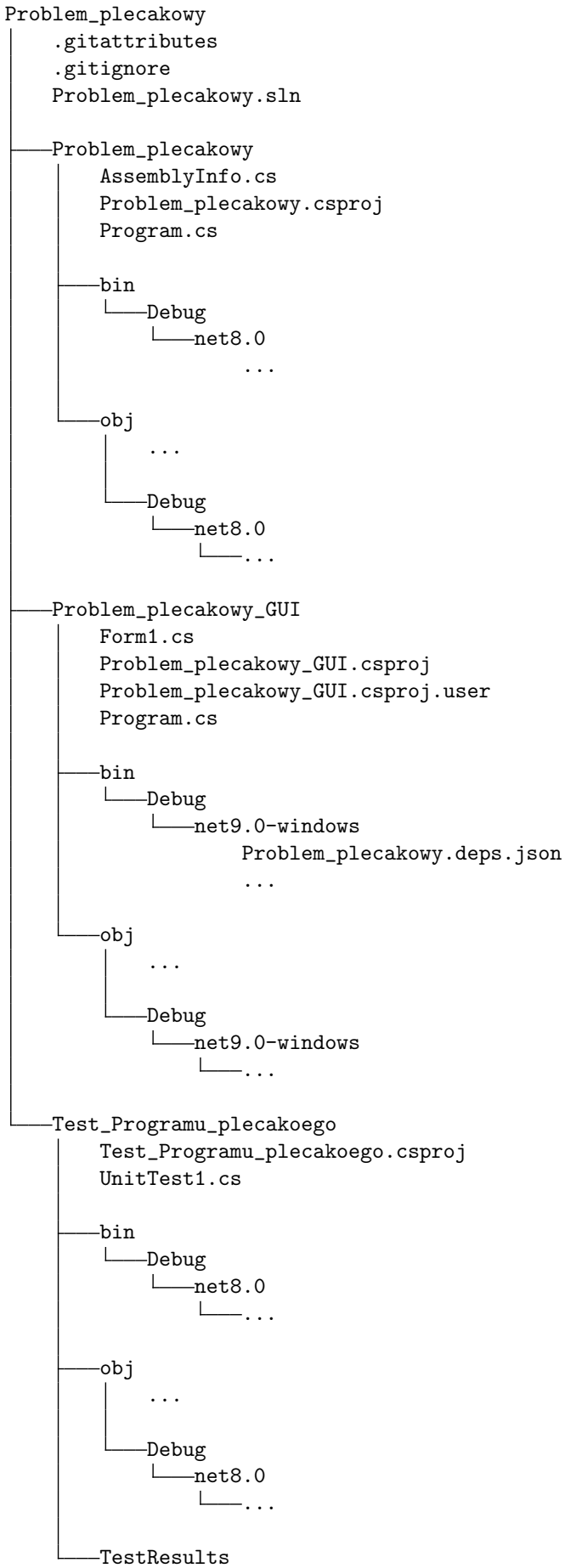
```
9         private Label Opis_liczba_przedmiotow;    private TextBox Input_liczba_przedmiotow;
10        private Label Opis_seed;                  private TextBox Input_seed;
11        private Label Opis_pojemnosc_plecaka;     private TextBox Input_pojemnosc_plecaka;
12
13        private Button przycisk_start;
14        private ListBox Pole_textowe_zawartosc; // Zawartość plecaka
15        private ListBox Pole_textowe_parametry; // Podane parametry przez użytkowika do TextBox

19        public Form1()
```

Metoda, nadaje wartość zmiennym `private` klasy.
Na koniec dodaje zmienne do interfejsu metodą `this.Controls.Add( <Zmienna> );`

## 1.2 Opis działania

Program przyjmuje parametry wejściowe od użytkownika (seed, liczba przedmiotów, pojemność plecaka). Inicjalizuje przedmioty, tworząc dla nich od razu parametr cenność. Sortuje je w kolejności od najcenniejszego. Program podejmuje w kolejności dodać każdy przedmiot do plecaka. Program informuje użytkownika o rezultacie.

## 2 Screen drzewa projektu



# 3 Screen kluczowego fragmentu programu na daną ocenę

## 3.1 Zadanie 1.

```
public Result Solve(int capacity)
{
    if (capacity <= 0) { Console.Write("ERROR, za mały plecak!\n"); }
    Result rozwiazanie = new Result(capacity); // [1.] Tworzy plecak.
    /* lista_sort = new List<Przedmiot>(ListaPrzedmiotow);
    Skrócona inicjalizacja:                                  */
    lista_sort = [.. ListaPrzedmiotow]; // [2.] Sortująca po cenności
    lista_sort.Sort((a, b) => b.Wartosc_Masa.CompareTo(a.Wartosc_Masa));
    foreach (Przedmiot tmp in lista_sort) { // [3.] Próbuje dodać każdy przedmiot
        //Console.WriteLine(tmp); /* DISPLAY posortowane */
        rozwiazanie.DodajPrzedmiot(tmp.nr_przedmiotu, tmp.wartosc, tmp.waga);
    }
    return rozwiazanie;
}
```

Opis klasy: 1.1.4.

## 3.2 Zadanie 2.

```
[ TestMethod ] // IV Sprawdzanie czy są uszeregowane od naj cenniejszych/masa do najmniej
public void TEST04 ()
{
    for(int i = 10; i<100; i += 5) // [1.] Dla 20 różnych ilości przedmiotów, losowych seedów.
    {
        int ilosc_przedmiotow = i;
        int seed = new Random().Next();
        Problem plecakowy = new Problem(seed, ilosc_przedmiotow); // [2.] Tworzy przedmioty
        int Pojemnosc = i*3; // [3.] Pojemność = 3 * ilość przedmiotów
        plecakowy.Solve(Pojemnosc); // [4.] Assert poniżej, Sprawdza czy po posortowaniu listy są różne
        Assert.AreNotSame(plecakowy.lista_sort, plecakowy.ListaPrzedmiotow, " 04 - porównanie list");
        decimal poprzednia_cennosc = plecakowy.lista_sort.First().Wartosc_Masa;
        foreach (Przedmiot rzecz in plecakowy.lista_sort)
        {
            Assert.IsTrue(poprzednia_cennosc >= rzecz.Wartosc_Masa,
                $"04 - Zła kolejnośc {poprzednia_cennosc} < {rzecz.Wartosc_Masa}");
            poprzednia_cennosc = rzecz.Wartosc_Masa;
        } // [5.] Sprawdza czy każdy kolejny jest mniej cenny.
    }
}




[ TestMethod ] // V Czy dodaje odpowiednią ilość przedmiotów
public void TEST05 ()
{
    for(int i = 10; i<100; i += 5) // [1.] Dla 20 różnych ilości przedmiotów, losowych seedów.
    {
        int ilosc_przedmiotow = i;
        int seed = new Random().Next();
        Problem plecakowy = new Problem(seed, ilosc_przedmiotow); // [2.] Tworzy przedmioty
        Assert.AreEqual(plecakowy.ListaPrzedmiotow.Count, ilosc_przedmiotow, " 05 liczba przedmiotów");
    } // [3.] Porównuje liczbę przedmiotów, tą w strukturze z prawdziwą.

}
```

### 3.3 Zadanie 3.

```
public partial class Form1 : Form
{
    private Label Opis_liczba_przedmiotow;    private TextBox Input_liczba_przedmiotow;
    private Label Opis_seed;                  private TextBox Input_seed;
    private Label Opis_pojemnosc_plecaka;     private TextBox Input_pojemnosc_plecaka;

    private Button przycisk_start;
    private ListBox Pole_textowe_zawartosc;
    private ListBox Pole_textowe_parametry;



    public Form1()
    {
        this.Text = "[.NET][CZ 15:15][LAB 1] \"Problem Plecakowy\"";
        this.Size = new System.Drawing.Size(620, 420);

        Opis_liczba_przedmiotow  = new Label {Text="Ilość przedmiotów:", Left=20, Top=20, Width=150};
        Opis_seed                = new Label {Text="Podaj seed:",        Left=20, Top=45, Width=150};
        Opis_pojemnosc_plecaka   = new Label {Text="Pojemność plecaka:", Left=20, Top=70, Width=150};
        Input_liczba_przedmiotow = new TextBox {Left=180, Top=20, Width=100, Text="10"};
        Input_seed               = new TextBox {Left=180, Top=45, Width=100, Text= "0"};
        Input_pojemnosc_plecaka  = new TextBox {Left=180, Top=70, Width=100, Text= "8"};

        przycisk_start = new Button {Text="Uruchom", Left=20, Top=95, Width=260};
        przycisk_start.Click += przycisk_start_funkcja!;

        Pole_textowe_parametry = new ListBox {Text="Parametry", Left= 20, Top=120, Width=260, Height=140};
        Pole_textowe_zawartosc = new ListBox {Text="Parametry", Left=320, Top=20,  Width=260, Height=360};

        this.Controls.Add(Opis_liczba_przedmiotow);  this.Controls.Add(Input_liczba_przedmiotow);
        this.Controls.Add(Opis_seed);                this.Controls.Add(Input_seed);
        this.Controls.Add(Opis_pojemnosc_plecaka);   this.Controls.Add(Input_pojemnosc_plecaka);
        this.Controls.Add(przycisk_start);
        this.Controls.Add(Pole_textowe_zawartosc);
        this.Controls.Add(Pole_textowe_parametry);
    }

    private void przycisk_start_funkcja(object sender, EventArgs e)
    {
        try
        {
            int iloscPrzedmiotow = int.Parse(Input_liczba_przedmiotow.Text);
            int seed             = int.Parse(              Input_seed.Text);
            int pojemnosc        = int.Parse( Input_pojemnosc_plecaka.Text);

            Problem plecakowy = new Problem(seed, iloscPrzedmiotow);
            Pole_textowe_parametry.Items.Clear();
            Pole_textowe_parametry.Items.Add($"Liczba wszystkich przedmiotów: {iloscPrzedmiotow}");
            Pole_textowe_parametry.Items.Add($"Seed generatora liczb losowych: {seed}");
            Pole_textowe_parametry.Items.Add($"Całkowita pojemność plecaka: {pojemnosc}");
            Pole_textowe_parametry.Items.Add($"");
            string[] tmp = plecakowy.Solve(pojemnosc).ToThreeStrings();
            Pole_textowe_parametry.Items.Add(tmp[0]);
            Pole_textowe_parametry.Items.Add(tmp[1]);
            Pole_textowe_parametry.Items.Add(tmp[2]);
            Pole_textowe_parametry.Items.Add(tmp[3]);

```

```
65              Pole_textowe_zawartosc.Items.Clear();
66              foreach(string str in plecakowy.ToFewStrings()) Pole_textowe_zawartosc.Items.Add(str);
67
68          }
69          catch (Exception ex)
70          {
71              MessageBox.Show("Zły format wprowadzonych danych\nProgram przyjmuje tylko int!\nBłąd: "
72                          + ex.Message, "Złe dane!", MessageBoxButtons.OK, MessageBoxIcon.Warning);
73          }
74      }
75  }
```