Kacper Malinowski, 263518

Automatyka i Robotyka, 6. Sem, spec. ARP

LAB 1 – Knapsack Problem

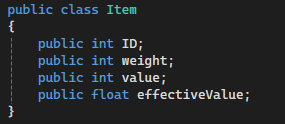
Link do repozytorium: <https://github.com/kacper-malinowski/LAB1Malinowski>

# Opis projektu

Program służy do generowania i rozwiązywania problemu plecakowego metodą aproksymacyjną. Algorytm losowo generuje zestaw przedmiotów na podstawie danych wejściowych użytkownika, następnie sortuje je względem ich „efektywnej wartości”, czyli ilorazu wartości i wagi, po czym zabiera przedmioty z góry listy dopóki plecak nie zostanie wypełniony lub przedmioty się nie skończą.

## Klasy:

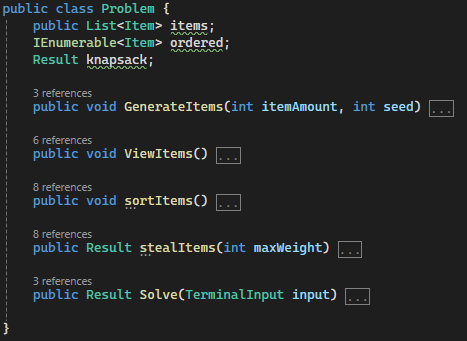
### Item



Rysunek 1 Klasa Item

Klasa Item zawiera dane odnośnie konkretnego przedmiotu, jego ID, wagę, wartość oraz efektywną wartość, według której lista przedmiotów będzie sortowana.

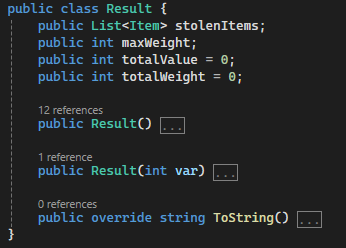
### Problem



Rysunek 2 Klasa Problem

Klasa problem zawiera listy służące przechowywaniu listy przedmiotów – posortowanej i nieposortowanej, oraz zmienną typu Result w której pojawi się rozwiązanie wygenerowane przez algorytm. Klasa ta zawiera także metody odpowiedzialne za tworzenie instancji problemu oraz jej rozwiązywanie.

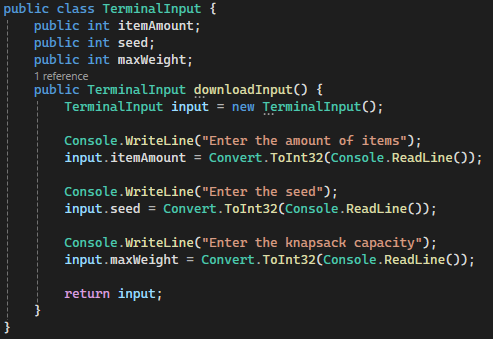
### Result



Rysunek 3 Klasa Result

Klasa Result przechowuje rozwiązanie problemu plecakowego – listę „ukradzionych” przedmiotów, maksymalną i wykorzystaną pojemność plecaka oraz wartość przedmiotów w nim się znajdujących. Klasa zawiera konstruktor oraz przeciążenie funkcji ToString(), aby uprościć wyświetlanie wyników problemu.

### TerminalInput



Rysunek 4 Klasa TerminalInput

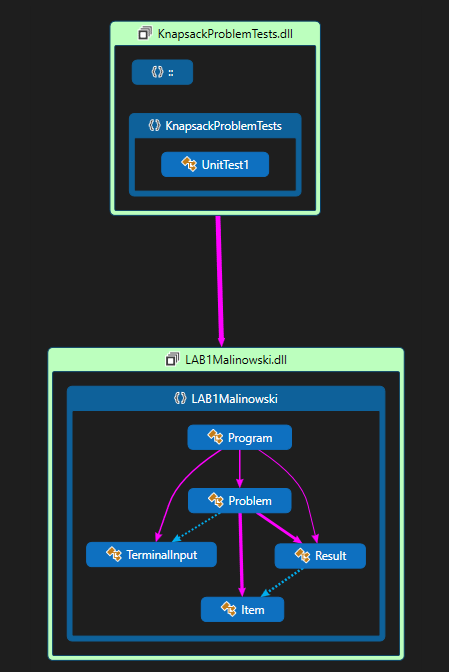
Klasa TerminalInput obsługuję wprowadzanie danych wejściowych przez użytkownika poprzez terminal. Metoda wczytująca działa niezależnie od algorytmu rozwiązywania co umożliwia jego zmianę na przykładowo interfejs graficzny.

# Opis testów

Program zawiera 8 testów sprawdzających poprawność działania algorytmu.

* CommonInstance() – Typowy przypadek z poprawnymi danymi
* EmptyInstance() – Przypadek w którym nie mamy żadnych przedmiotów do umieszczenia w plecaku.
* DoesTheItemOrderMatter() – Przypadek zawierający dwie listy z tymi samymi przedmiotami jednak ułożonymi w innej kolejności.
* CustomInstance() – Instancja problemu z ręcznie wpisanymi danymi, dla których ręcznie obliczono prawidłowe rozwiązanie.
* NoSpaceInBackpack() – Plecak nie maksymalną pojemność równą 0 – nie można w nim umieścić żadnego przedmiotu.
* NoItemsFitBackpack() – Maksymalna pojemność plecaka jest mniejsza niż masa jakiegokolwiek przedmiotu na liście.
* BackpackTooLarge() – Pojemność plecaka jest znacznie większa niż suma przedmiotów na liście.

# Drzewo projektu



# Kluczowa część algorytmu



