Dokumentacja techniczna projektu Decompressor

[Skład zespołu 1](#_Toc1812533312)

[Opis projektu 2](#_Toc1839921526)

[Użyte technologie 3](#_Toc1643069348)

[Sposób działania 3](#_Toc1131766726)

[Testowanie oprogramowania 7](#_Toc667938700)

# Skład zespołu

* Mikołaj Pątkowski
* Marcin Wardak
* Paweł Wróblewski

# Opis projektu

Projekt ma na celu utworzenie dekompresora plików. Pliki są zakodowane przy pomocy kompresora gr. 6 .

Link do repozytorium: https://github.com/Namlev1/jimp2\_dcmp

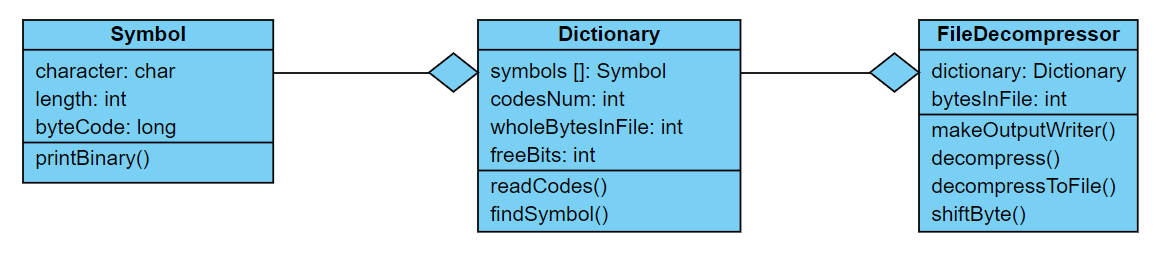
# Użyte technologie

* Język Java
* Maven
* JavaFX

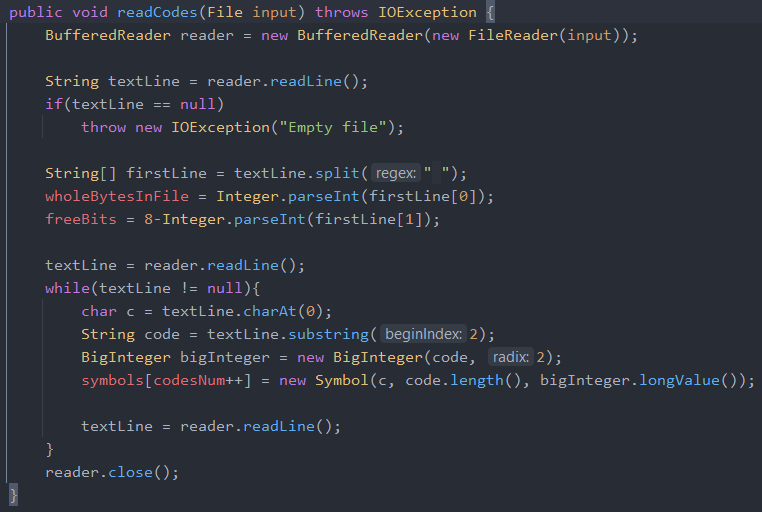
# Sposób działania

W celu dekompresji pliku zostały utworzone 3 klasy:

1. Symbol
2. Dictionary
3. FileDecompressor

Każdy kod przechowywany jest jako Symbol. Jego znak zawarty jest w character, jego kod zapisany jest w zmiennej typu long. Ponieważ typ long nie rozróżnia liczb 0b001 od 0b000001, zastosowana jest zmienna length.

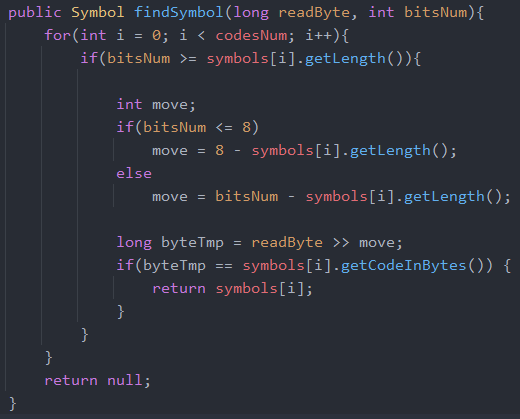
Dictionary posiada informacje o wszystkich symbolach. Funkcja readCodes(File input) czyta plik code.txt (tworzony przez kompresor) i zapisuje potrzebne informacje w tablicy obiektów Symbol. Zmienna codesNum zawiera liczbę kodów w słowniku Dictionary.

Pierwsza linia code.txt zawiera informację, ile całych bajtów jest zawartych w pliku oraz ile bitów jest zajętych w ostatnim bajcie. Następnie zostają wczytane kody, w formacie

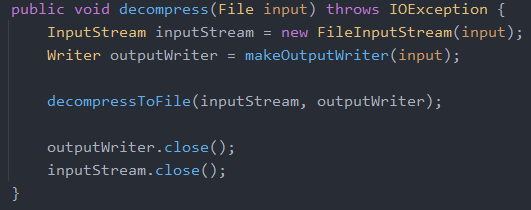
*”[symbol] [kod]”*

np. E 101.

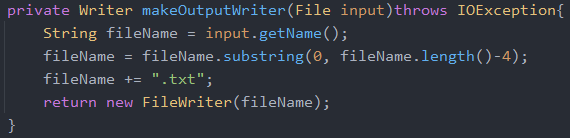
Funkcja findSymbol(long readByte, int bitsNum) zwraca odpowiedni Symbol, jeżeli znajduje się w słowniku, lub null w innym przypadku. Kod musi być niedłuższy niż bitsNum (po to, żeby program nie zwracał kodu 0b1000, gdy ma rozpatrywać tylko 2 ostatnie bity). Następnie odkodowywany bajt jest skracany do adekwatnej długości, żeby ignorować początkowe bity (np. 0b10100011 -> 0b0011, gdy porównywany kod ma długość 4) oraz jest porównywany do danego kodu.



Obiekt FileDecompressor jest odpowiedzialny za odkodowanie pliku output.bin w oparciu o dostarczony w constructorze obiekt Dictionary. Główną funkcją wywoławczą jest decompress(File input).



makeOutputWriter zwraca FileWriter z odpowiednią nazwą pliku.



Najważniejszą funkcją jest decompressToFile(InputStream inputStream, Writer outputWriter).

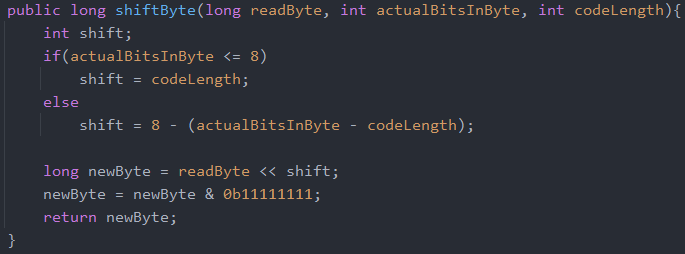
Z każdą iteracją for następuje pobranie nowego bajtu. Zmienna bitsNumber mówi, ile nieprzeczytanych bitów znajduje się jeszcze w rozpatrywanym bajcie (lub bajtach). W pętli while następuje odkodowanie bajtu.

Jeżeli funkcja dictionary.findSymbol() zwróci adekwatny Symbol, następuje dopisanie jego znaku do pliku wyjściowego, odkodowywany bajt zostaje przesunięty, a informacja o rozpatrywanych bitach zostaje odpowiednio zmniejszona. W przeciwnym wypadku, tzn. gdy dla danego bajtu i odpowiedniej jego długości nie znajduje się odpowiadający mu Symbol, bajt zostaje przesunięty o 8 bitów w lewo, a w następnej iteracji zostanie do niego dopisany kolejny bajt np. 0b010 -> 0b010\_11001100 (nie istnieje kod dla 0b010, ale być może dla 0b01011 już tak).

Funkcja shiftByte() przesuwa rozpatrywany bajt o długość odczytanego kodu. Robi to w następujący sposób:

1. Wylicza przesunięcie bitowe (jeżeli bajt to tak naprawdę więcej niż 1 bajt, np. 0b010\_11000000, korzysta z innej formuły)
2. Przesuwa bajt w lewo
3. Stosuje maskę tak, aby z danej liczby zostało jedynie 8 bitów.

Odczytywany bajt zawsze będzie skrócony do długości jednego bajta, gdyż niemożliwa jest sytuacja, że w bajcie np. 0b010\_11001100 zostanie wykryty kod 0b01, ponieważ zostałby on wykryty wcześniej jeszcze przed dołączeniem do pierwszego bajta drugiego.



Przykład 1:

Bajt: 0b11001111 (długość 8)

Kod: 0b00001100 (długość 4)

Shift: 4

0b11001111 -[4]-> 0b00001100\_11110000 -[maska]-> 0b11110000 (przesunięty bajt, długość 4)

Przykład 2:

Bajt: 0b01011100\_00001110\_11000000 (długość 24)

Kod: 0b00000000\_10111000\_00011101 (długość 17)

Shift: 1

0b01011100\_00001110\_11000000 -[1]-> 0b00000000\_10111000\_00011101\_10000000 -[maska]-> 0b10000000 (nowy bajt, długość 7)

# Testowanie oprogramowania

Testy odbywają się na losowy generowanych plikach tekstowych o długości 100, 500, 1000 znaków. Klasa TestDecompressor jest odpowiedzialna za tworzenie losowych plików, kompresowanie ich, dekompresję oraz porównanie zawartości.