

# Specyfikacja implementacyjna programu realizującego kompresję plików wykorzystującego algorytm Huffmana

Autorzy: Adrian Chmiel, Mateusz Tyl

08.04.2023

Historia zmian dokumentu:

Autor:	Data:	Opis zmiany:	Wersja dokumentu
Adrian Chmiel	10.06.2023	Dodanie pierwszego szkicu	1.0

## Cel dokumentu

Celem tego dokumentu jest przedstawienie informacji o sposobie działania programu od strony technicznej poprzez analizę każdego pliku z osobna oraz przedstawienie związków między nimi.

## Informacje ogólne

Program realizuje dekompresję plików pochodzących z kompresora przygotowanego w ramach poprzedniego projektu. Został on w całości napisany w języku Java. Dla prawidłowego działania wszystkich komponentów programu zalecane jest korzystanie z wersji języka Java 18 oraz wersji JDK 20.

Program jest przystosowany do pracy i kompilacji zarówno na systemach Unixowych, jak i systemie Microsoft Windows. Do dyspozycji programisty przygotowano szeroki zakres testów różnej złożoności do wykonania..

## Omówienie kodu źródłowego

Program oferuje kilka modułów, które wzajemnie współpracują ze sobą. Niektóre z nich pracują jednak niezależnie od innych.

## Katalog główny

- Dekompresor.jar -> archiwum zawierające wszystkie klasy znajdujące się w package *dekompresor*

- README.md -> plik zawierający informacje o projekcie przygotowany pod repozytorium na GitHubie

## Package "dekompresor"

Zawiera on kod źródłowy klas, które składają się potem na archiwum *Dekompresor.jar*

- BitsAnalyze -> zawiera funkcje służące do analizy kolejnych bitów zawartych w kolejnych znakach przy dekompresji oraz analizuje zapisany słownik w pliku. Funkcje korzystają z pomocniczej klasy DNode odwzorowującej drzewo binarne. Zawarte są w niej następujące tryby działania (opisane przy pomocy komentarzy w kodzie): dictRoad - tryb uaktywnia się, gdy przemieszczamy się w drzewie, dictWord - uaktywniany, gdy znajdziemy się w liściu w celu odczytania znaku/słowa, bitsToWords - służy do odczytywania skompresowanych danych po odczytaniu całości słownika
- Buffer -> klasa zawierająca pomocnicze pola pod tworzenie różnego rodzaju buforów
- Controller -> klasa sprawdzająca argumenty podane na wejściu oraz poprawność podanych plików
- Decompressor -> klasa zawierająca główne funkcje dekompresujące, które wywołują inne metody pomocnicze oraz zapisują do pliku wyjściowego odczytane znaki
- Decrypt -> klasa wykonująca proste odszyfrowanie dla plików zaszyfrowanych i nieskompresowanych
- DNode -> klasa tworząca pomocnicze drzewo służące do prawidłowego odczytywania słownika podczas dekompresji
- FileManager -> klasa abstrakcyjna wspierająca prawidłowe zarządzanie oraz przygotowywanie plików
- Flags -> klasa pomocnicza służąca do sprawnego odczytania flag dla danego pliku
- Main -> uruchamia cały proces dekompresji
- Mode -> klasa pomocnicza służąca do przechowywania aktualnego trybu
- SendDataToGUI -> klasa zawierająca metodę odpowiadającą za uzupełnianie pliku *data* odpowiednimi wartościami, które są następnie pobierane przez GUI
- Settings -> klasa pomocnicza służąca do przechowywania wybranych przez użytkownika ustawień przekazanych w argumentach

- Utils -> klasa zawierająca pomocnicze metody do sprawdzania poprawności pliku oraz wyświetlania pomocy programu na standardowy strumień

## Package "dekompresorgui"

W tej części zawarty jest opis klas odpowiadający za graficzne przedstawienie okienka oraz ewentualnej wizualizacji słownika zrealizowany przy pomocy JavaFX.

- AnalyzeDataFromDecompressor -> klasa analizująca plik *data* tworzony w trakcie działania programu
- CLine -> klasa tworząca linię o określonych parametrach między podanymi koordynatami (w przypadku tego programu dwa środki kwadratów tj. węzłów)
- CObject -> interface implementowany przez CLine oraz CRect
- CRect -> klasa tworząca kwadrat o określonych parametrach będący danym węzłem w drzewie, w przypadku liścia zawiera również literkę odpowiadającą danemu kodowi
- CTree -> klasa realizująca obrazowanie słownika w sposób podobny do wspomnianego wcześniej DNode
- DekompresorGUI -> klasa tworząca główne okienko umożliwiające wybór pliku, ustawień itd.
- DictTree -> klasa tworząca okienko z wizualizacją całego słownika

## Dodatkowe pliki tworzone w trakcie działania programu

- data -> plik pomocniczy służący do komunikacji dekompresora z graficznym okienkiem
- tree -> plik pomocniczy przechowujący informacje o tym, jak tworzyć drzewo

System tworzenia drzewa jest bardzo podobny do systemu zapisu słownika wyjaśnionego w specyfikacji funkcjonalnej, lecz zamiast pojedynczych bitów 00, 01, 10, 11 odpowiednio zapisuję pojedyncze bajty znakami '0', '1', '2', '3'. Sama wizualizacja jest wyświetlania jedynie dla plików skompresowanych 8-bitowo, a więc informacje zawarte w samym pliku *tree* są akuratne również tylko i wyłącznie dla takiej kompresji.