

**Programowanie obiektowe**

*Cykliczna dystrybucja zbiorów, minimalizująca odchylenie standardowe*

Prowadzący: Autor:

mgr inż. Ewa Żesławska Samir Al-Azazi

66045

Kierunek: 3 IIZ/2021, grupa GP02

Rzeszów, r.a. 2022/2023

Spis treści

[1. Cele projektu 3](#__RefHeading___Toc706_448703101)

[2. Opis techniczny projektu 3](#__RefHeading___Toc1259_3016230123)

[3. Prezentacja warstwy użytkowej projektu 3](#__RefHeading___Toc710_448703101)

[4. Testy jednostkowe 4](#__RefHeading___Toc716_448703101)

[5. Repozytorium, system kontroli wersji 4](#__RefHeading___Toc718_448703101)

[6. Testy funkcjonalne 5](#__RefHeading___Toc720_448703101)

[7. Podsumowanie 6](#__RefHeading___Toc734_448703101)

[8. Literatura 6](#__RefHeading___Toc736_448703101)

1. **Cele projektu**

Celem aplikacji jest równy podział zadań pomiędzy zespoły, przy cyklicznym napływaniu zadań (dystrybuowany zbiór nie jest znany od samego początku). Wpisuje się to w nurt nowoczesnych metodyk pracy zwinnej. Może być używana w celu automatyzacji organizacji pracy, przez kierowników, menadżerów.

**Wymagania funkcjonalne**

* Dystrybucja zbioru produktu (Input) pomiędzy cele (Target).
* Dane muszą napływać z pliku CSV.
* Rozdystrybuowana baza musi być eksportowana do pliku CSV.
* Dane muszą być zapisane w bazie danych opartych o silnik SQL.
* Algorytm dystrybucji musi dążyć do jak najmniejszego odchylenia standardowego pomiędzy produktami celu, a wszystkimi celami, wszystkich produktów.
* Algorytm musi brać pod uwagę tylko 10 ostatnich iteracji.

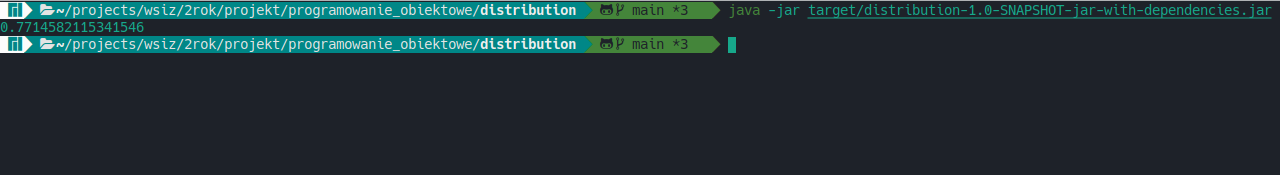
**Wymagania niefunkcjonalne**

* Aplikacja działa z terminala.
* Aplikacja nie wymaga żadnych argumentów konsoli.
* Aplikacja tworzona jest w języku Java
* Aplikacja nawiązuje połączenie z bazą danych i używa rekordów w niej zapisanych.

1. **Opis techniczny projektu**

* Środowisko programistyczne Javy: Java JDK 15
* Środowisko programistyczne: Visual Studio Code (Language Support For Java)
* Narzędzie do testów jednostkowych: Junit
* Menadżer projektu Java: Maven

1. **Prezentacja warstwy użytkowej projektu**

Aplikacja nie posiada warstwy graficznej. W zamyśle ma być używana jako narzędzie do automatyzacji (np. poprzez wywołania CRON).

W przypadku błędu, jest on wyświetlany i kierowany do *stdout.err*, co pozwala na logowanie błędów.

Aplikacja komunikuje się z wewnętrzną bazą danych, opartą na silniku SQLite. Przechowuje informacje o zespołach i rozdystrybuowanych zadaniach.

Przy starcie aplikacji, z zewnętrznego pliku CSV są zaczytywane nowe zadania do rozdystrybuowania. Po rozdystrybuowaniu zbioru zadań, wynik dystrybucji jest zapisywany do pliku *output.csv*.

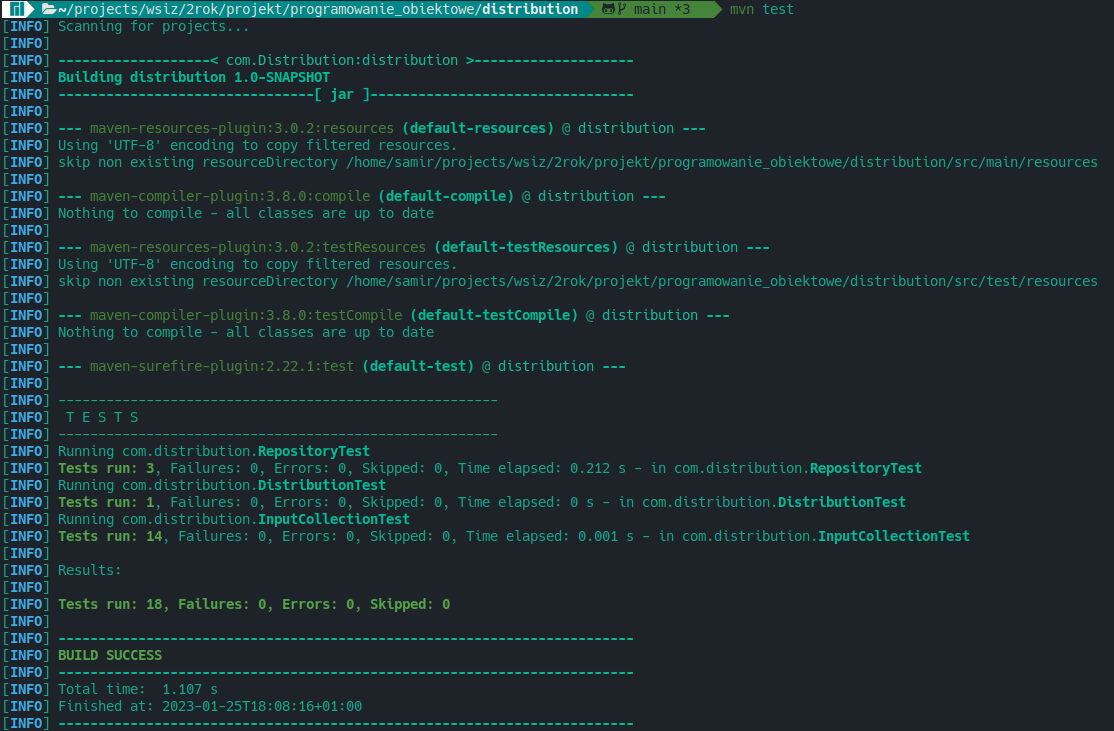
Algorytm dystrybucji prezentuje się następująco:

1. Dla każdego dystrybuowanego zadania.
2. Dla każdego zespołu, którego liczba zadań jest równa najmniejszej liczbie zadań spośród wszystkich zespołów.
3. Oblicz średnie odchylenie standardowe już rozdystrybuowanych zadań w zespole, od średniej dla wszystkich zespołów.
4. Oblicz średnie odchylenie standardowe dla zespołu, gdyby zadanie zostału mu przypisane.
5. Oblicz wskaźnik dopasowania zadania do zespołu, jako wartość bezwzględna różnicy między dwoma odchyleniami standardowymi obliczonymi w krokach 3 i 4.
6. Po obliczeniu wskaźników dopasowań danego zadania dla wszystkich zespołów, przypisz zadanie dla zespołu, dla którego wskaźnik dopasowania jest największy i usuń zadanie z zbioru zadań do rozdystrybuowania.
7. Jeśli zbiór zadań do rozdystrybuowania nie jest pusty, wróć do kroku 1.
8. **Testy jednostkowe**

Zaprojektowane zostały testy jednostkowe, z pomocą Junit we wsparciu Maven.

Testowane są działania

* kolekcji
* repozytorium (połączenie z bazą danych)
* proces dystrybucji



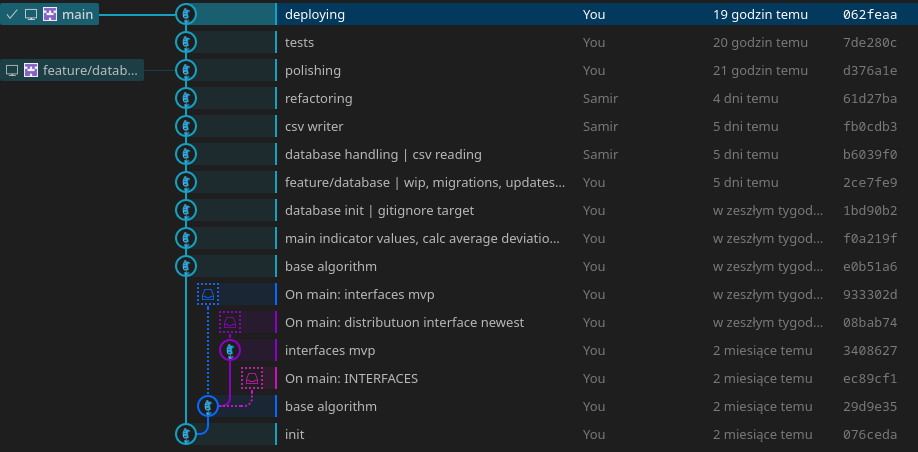
1. **Repozytorium, system kontroli wersji**

Projekt został zrealizowany z wykorzystaniem systemu kontroli wersji Git, na platformie GitHub.

Na Rysunku 1 przedstawiono zrzut ekranu pokazujący historię kosmitów.

Cały projekt jest zaprojektowany zgodnie z zasadami programowania obiektowego SOLID, Clean Code, Don’t Repeat Yourself, Keep It Simple Stupid.

Dokumentacja oraz projekt został umieszczony w repozytorium dostępnym pod adresem: https://github.com/aazsamir/po\_project\_distribution.

Rysunek 1. Graf komitów

1. **Testy funkcjonalne**

Stosując algorytm odwrotny do przedstawionego, utworzony został zbiór nierówno rozdystrybuowany.

Średnie wartości odchyleń to

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Skomplikowanie zadania** | **Wymagany czas do realizacji** | **Koszt wykonania** | **Wymagana liczba osób** | **Liczba zadań specjalnych** |
| 2.98 | 4.03 | 3.76 | 4.20 | 2.39 |

Natomiast konkretne wartości pomiędzy zespołami (każdy wiersz reprezentuje jeden zespół)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Skomplikowanie zadania** | **Wymagany czas do realizacji** | **Koszt wykonania** | **Wymagana liczba osób** | **Liczba zadań specjalnych** |
| 34 | 16 | 24 | 74 | 5 |
| 22 | 86 | 19 | 30 | 2 |
| 30 | 33 | 55 | 46 | 6 |
| 11 | 27 | 81 | 21 | 9 |
| 24 | 4 | 36 | 7 | 6 |
| 17 | 58 | 20 | 22 | 6 |
| 60 | 39 | 33 | 66 | 7 |
| 35 | 10 | 6 | 28 | 6 |

Następnie, kolejne zadania zostały rozdystrybuowane, tak, by średnie wartości odchyleń były jak najmniejsze.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Skomplikowanie zadania** | **Wymagany czas do realizacji** | **Koszt wykonania** | **Wymagana liczba osób** | **Liczba zadań specjalnych** |
| 2.08 | 1.65 | 1.51 | 1.50 | 2.41 |

A konkretne wartości w zespołach

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Skomplikowanie zadania** | **Wymagany czas do realizacji** | **Koszt wykonania** | **Wymagana liczba osób** | **Liczba zadań specjalnych** |
| 33 | 30 | 34 | 33 | 7 |
| 28 | 34 | 35 | 44 | 8 |
| 30 | 35 | 39 | 39 | 8 |
| 28 | 34 | 32 | 36 | 6 |
| 36 | 21 | 32 | 47 | 5 |
| 20 | 45 | 32 | 34 | 2 |
| 31 | 42 | 31 | 33 | 4 |
| 30 | 32 | 33 | 31 | 8 |

Odchylenie standardowe na przedstawionych danych, to odchylenie w ramach jednej kolumny. Jak widać po dystrybucji bazy za pomocą algorytmu, wartości w celach trzymają się bliżej wartości środkowej, niż jest to na początku, gdzie różnice potrafiły być bardzo duże.

Na czerwono zaznaczone zostały wartości minimalne, a na niebiesko wartości maksymalne.

Same różnice zaznaczonych wartości maksymalnych i minimalnych pokazują, że różnice pomiędzy zespołami się zmniejszyły, co jest następstwem faworyzowania jak najmniejszej wartości odchylenia standardowego poprzez algorytm.

1. **Podsumowanie**

Algorytm dystrybucji działa poprawnie, średnie odchylenia standardowe nie przekraczają wartości 2.5.

Całość jest zaprojektowana zgodnie z popularnymi wzorcami projektowania.

Zabrakło tylko obsługi parametrów wejściowych z konsoli bądź z zmiennych środowiskowych systemu.

1. **Literatura**

1. Martin Fowler, Architektura systemów zarządzania przedsięborstwem. Wzorce projektowe.

2. Robert C. Martin, Czysty kod. Podręcznik dobrego programisty.