Lab04 - Testy jednostkowe

Cel ćwiczenia: weryfikacja poprawności oprogramowania z wykorzystaniem testów jednostkowych na przykładzie biblioteki JUnit5

Zadanie

Dla każdego zadania z Lab03 stwórz testy jednostkowe przy użyciu biblioteki JUnit. Testy zamknij w odpowiednich TestSuite. Wykorzystaj mechanizmy wbudowane w IDE do uruchomienia testów i prezentacji wyników. Postaraj zapewnić się 100% pokrycie kodu. Poza typowymi testami poprawności działania algorytmów, zaimplementuj sprawdzanie wyjątków oraz testy czasu wykonywania. Testy muszą mieć sens. Postaraj zastosować jak najwięcej różnych asercji.

Niezbędne informacje znajdują się na [stronie JUnit](https://junit.org/junit5/) lub ich [GitHubie](https://github.com/junit-team/junit5/wiki).

Wykorzystaj Mavena do dołączania zależności oraz uruchamiania i testowania projektu.

Wskazówki:

1. [Testowanie kodu w Intellij Idea](https://www.jetbrains.com/help/idea/configuring-testing-libraries.html)

2. [Analiza pokrycia testami](https://www.jetbrains.com/help/idea/code-coverage.html)

3. [Pisanie testów jednostkowych](https://junit.org/junit5/docs/current/user-guide/#writing-tests)

4. [Wykonywanie testów jednostkowych](https://junit.org/junit5/docs/current/user-guide/#running-tests)

Teoria:

1. Jakie dwa rodzaje testów wyróżniamy i czym się one cechują (testy automatyczne i manualne)

2. Czym są testy jednostkowe i do czego służą?

3. Czym jest TestSuite w JUnit? [Przykład](https://howtodoinjava.com/junit5/junit5-test-suites-examples/)

4. Czym jest Mock Objects? Gdzie się go stosuje?

5. Na czym polega Test fixture w JUnit. Jakich adnotacji możemy użyć.

6. [Poprawne nazewnictwo testów](https://dzone.com/articles/7-popular-unit-test-naming)

7. Na czym polega Test Driven Development (TDD)?

8. Czym jest pokrycie kodu (ang. Code coverage)?

9. Jak zbudowane są testy JUnit? (adnotacje i nazwy metod)