

Programowanie w JAVA

Lab. 5 – Testowanie kodu

Do każdego algorytmu z LAB 3 (Algorytmy) stwórz testy jednostkowe z wykorzystaniem JUnit. Testy zamknij w odpowiednich `TestSuite`. Wykorzystaj mechanizmy IDE z którego korzystasz do uruchomienia testów i prezentacji wyników. Postaraj zapewnić się 100% pokrycie kodu. Poza typowymi testami poprawności działania algorytmów, zaimplementuj sprawdzanie wyjątków (<https://github.com/junit-team/junit4/wiki/exception-testing>) oraz testy czasu wykonywania (<https://github.com/junit-team/junit4/wiki/timeout-for-tests>). Testy muszą mieć sens. Postaraj zastosować jak najwięcej różnych asercji.

1. Teoria:
 - a. Jakie dwa rodzaje testów wyróżniamy i czym się one cechują (testy automatyczne i manualne)
 - b. Czym są testy jednostkowe i do czego służą?
 - c. Czym jest `TestSuite` w JUnit?
 - d. Czym jest `Mock Objects`? Gdzie się go stosuje?
 - e. Na czym polega `Test fixture` w JUnit. Jakich adnotacji możemy użyć.
 - f. Poprawne nazewnictwo testów
<https://dzone.com/articles/7-popular-unit-test-naming>
 - g. Na czym polega `Test Driven Development (TDD)`?
 - h. Czym jest pokrycie kodu (ang. `Code coverage`)?
 - i. Jak zbudowane są testy JUnit? (adnotacje i nazwy metod)
2. Wskazówki
 - a. Testowanie kodu w IntelliJ Idea: <https://www.jetbrains.com/help/idea/configuring-testing-libraries.html>
 - b. Analiza pokrycia testami: <https://www.jetbrains.com/help/idea/code-coverage.html>

Po uzyskaniu zaliczenia na zajęciach, prześlij źródła w archiwum **zgodnie z konwencją nazewnictwą** (patrz Lab0.pdf) do chmury na adres:

<https://cloud.kisim.eu.org/s/5PJXWTywQX4cxMD>

najpóźniej do następnych zajęć.