Projektowanie obiektowe

Laboratorium 5
Testy jednostkowe

Maj 2020

1 Opis

1. Celem laboratorium jest zapoznanie sie z technikami tworzenia testów jednostkowych z wykorzystaniem mocków oraz elementami BDD i TDD.
2. W ramach laboratorium nastapi rozszerzenie funkcjonalności istniejacego systemu do obsługi zamówień w sklepie internetowym wraz z utworzeniem adekwatnych testów jednostkowych (zgodnie z elementami metodyki TDD).

2 Przygotowanie do zajeć

2.1 Wymagania

Na zajeciach wykorzystamy nastepujace narzedzia:

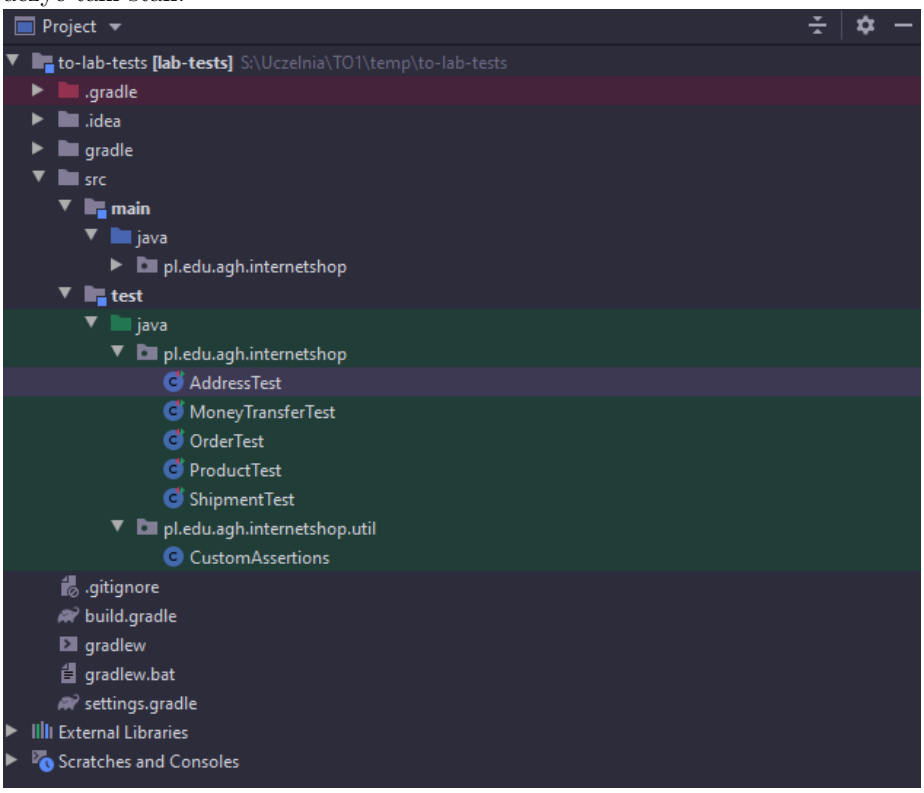
1. IntelliJ + Gradle
2. JDK 8
3. Biblioteki JUnit5 oraz Mockito

2.2 Importowanie projektu

Nie ma potrzeby instalowania bibliotek recznie, wystarczy poprawnie zaimportować projekt Gradle, a ten sam dociagnie wszystkie potrzebne zależności:

1. Wypakowujemy załaczona paczke z kodem.
2. Otwieramy IntelliJ, wybieramy *File* → *Open...* i wskazujemy wypakowany katalog.
3. Jeśli IntelliJ spyta nas o rodzaj importowanego projektu, wybieramy *Gradle project*.
4. Czekamy, aż IntelliJ pobierze zależności.

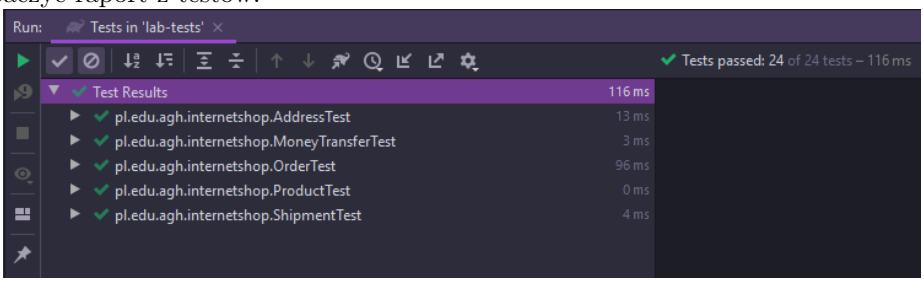
Jeśli wszystko poszło ok, po rozwinieciu struktury projektu powinniśmy zobaczyć taki stan:



Uwaga. *Kluczowe by katalogi z kodem zostały rozpoznane jako źródłowe – jeśli tak sie stało, katalog java w module main bedzie niebieski, a katalog java w module test zielony.*

2.3 Uruchamianie testów

Po zaimportowaniu warto uruchomić wszystkie testy w aplikacji, klikajac PPM na korzeniu projektu i wybierajac *Run 'Tests in 'lab-tests"*. Powinniśmy wówczas zobaczyć raport z testów:



Uwaga. *jeśli podczas uruchamiania pojawiły sie błedy kompilacji może to oznaczać, że projekt Gradle nie został prawidłowo zaimportowany lub IntelliJ podpiał inne SDK niż JDK8. Pierwszy problem można spróbować rozwiazać klikajac Reimport All Gradle Projects w widoku Gradle. Drugi problem wymaga recznego ustawienia SDK w File → ProjectStructure... → Project.*

3 Problem?

Twoim klientem jest rozwijajacy sie sklep internetowy, który obsługuje obecnie jeden typ płatności (płatność przelewem) i jeden typ wysyłki (przesyłka pocztowa). Właściciel zwrócił sie do Ciebie z prośba o dodanie nowych funkcji.

4 Realizacja

Należy zadbać o to, aby utrzymać obecne testy. W trakcie pisania rozszerzeń funkcjonalności należy:

1. Przemyśleć i wprowadzić ew. zmiany na poziomie struktury kodu: nowy interfejs, nowa metoda w interfejsie lub klasie (bez implementacji logiki działania), które beda realizować nowe funkcjonalności.
2. Napisać test sprawdzajacy realizacje pojedynczego przypadku dla nowej funkcjonalności.
3. Przystapić do implementacji funkcjonalności, która pozwoli na poprawne uruchomienie testu.
4. Przeprowadzić refaktoryzacje tak, aby kod był lepszy: bardziej czytelny, spójny, bez powtórzeń itd.
5. Powrót do punktu 2.

5 Zadania

1. Zmienić wartość procentowa naliczanego podatku z 22% na 23%. Należy zweryfikować przypadki brzegowe przy zaokragleniach.
2. Rozszerzyć funkcjonalność systemu, tak aby zamówienie mogło obejmować wiecej niż jeden produkt na raz.
3. Dodać możliwość naliczania rabatu do pojedynczego produktu i do całego zamówienia.
4. Umożliwić przechowywanie historii zamówień z wyszukiwaniem po: nazwie produktu, kwocie zamówienia, nazwisku zamawiajacego. Wyszukiwać można przy użyciu jednego lub wielu kryteriów.

6 Materiały pomocnicze

1. M. Fowler, GivenWhenThen
2. R. Leszko, Perfect Unit Test
3. S. Haines, JUnit 5 tutorial, part 1: Unit testing with JUnit 5, Mockito, and Hamcrest
4. JUnit5 User Guide oraz O różnicach miedzy JUnit4 a JUnit5