Test jednostkowy (ang. *unit test*) to sposób testowania programu, w którym wydzielamy mniejszą jego część, jednostkę i testujemy ją w odosobnieniu. Taką jednostką do testowania może być pojedyncza klasa czy metoda. Testy jednostkowe można pisać bez bibliotek zewnętrznych jednak jest to uciążliwe. Dodatkowo warto używać istniejących bibliotek ponieważ IDE dobrze się z nimi integrują. Test jednostkowy to metoda testująca naszą jednostkę, metodę w innej klasie z dodaną adnotacją @Test. Wynik tej metody jest przekazywany do metody Assert.assertTrue(), jest to tak zwana asercja.

Asercje

Asercje to metody dostarczone przez bibliotekę JUnit, które pomagają przy testowaniu. Jeżeli metoda zwróci false, asercja assertTrue rzuci wyjątek, który przez IDE zostanie zinterpretowany jak test jednostkowy, który pokazuje błąd działania testowanego kodu. Mówimy wówczas, że „test nie przeszedł". Testy jednostkowe łączymy w klasy z testami, bardzo często nazywamy je tak samo jak klasy, które testujemy dodając do nich Test na końcu. Asercje tworzą komunikaty błędów (w trakcie testów jednostkowych), które ułatwiają znalezienie błędu. Komunikaty te są bardziej czytelne niż standardowy wyjątek AssertionError. Asercje w bibliotece JUnit to nic innego jak metody statyczne w klasie Assert. Poniżej znajduje się kilka najczęściej stosowanych asercji.

- assertTrue sprawdza czy przekazany argument to true,
- assertFalse sprawdza czy przekazany argument to false,
- assertNull sprawdza czy przekazany argument to null,
- assertNotNull sprawdza czy przekazany argument nie jest nullem,
- assertEquals przyjmuje dwa parametry wartość oczekiwaną i wartość rzeczywistą, jeśli są różne rzuca wyjątek,
- assertNotEquals przyjmuje dwa parametry wartość oczekiwaną i wartość rzeczywistą, rzuci wyjątek jeśli są równe.

Importy statyczne

W języku Java trzeba importować klasy z innych pakietów, które chcemy użyć w definicji naszej klasy. Poza standardową konstrukcją ze słowem kluczowym import istnieją także tak zwane importy statyczne. Import statyczny pozwala na zaimportowanie metody/wszystkich metod statycznych znajdujących się w definicji jakiejś klasy.

Testowanie metod rzucających wyjątki

Czasami zdarza się przetestować pewną sytuację wyjątkową. Nie powinniśmy móc utworzyć instancji klasy z niepoprawnymi argumentami. Wywołanie konstruktora w teście z niepoprawnymi argumentami kończyłoby się od razu rzuceniem wyjątku, czyli testem jednostkowym, który nie przeszedł.

Przygotowanie testów i cykl życia testów

Testy jednostkowe wymagają pewnego „przygotowania". Na przykład trzeba utworzyć instancję, która będzie później testowana. Twórcy biblioteki JUnit stworzyli adnotację @Before, którą można dodać do metody w klasie z testami. Metoda ta zostanie uruchomiona przed każdym testem jednostkowym. Adnotacja @Before jest jedną z czterech adnotacji, które pozwalają na wykonanie fragmentów kodu przed/po testach. Pozostałe trzy to:

- @After– metoda z tą adnotacją uruchamiana po każdym teście jednostkowym, pozwala na „posprzątanie" po teście,
- @AfterClass– metoda statyczna z tą adnotacją uruchamiana jest raz po uruchomieniu wszystkich testów z danej klasy,
- @BeforeClass– metoda statyczna z tą adnotacją uruchamiana jest raz przed uruchomieniem pierwszego testu z danej klasy.

JUnit