Technologia Programowania 2017/18 - Lista 1 (lab)
Termin: do 20 października

0 Java JDK

Zakładam, że masz pobrane Java Development Kit i pytania takie jak to są już za Tobą.

1 Zintegrowane środowisko programistyczne

Zadanie 1 — Pobierz aktualną wersję Eclipse IDE for Java Developers. Aby zapoznać się ze środowiskiem przejrzyj następujące rozdział pomocy: *Help* > *Help Contents*:

- > *Workbench User Guide* > *Eclipse platform overview*,
- > *Workbench User Guide* > *Getting Started* > *Basic tutorial*,
- > *Java development user guide* > *Getting Started* > *Basic tutorial*.

W ostatnim punkcie zwróć szczególną uwagę na sekcje: *Debugging your programs* oraz *Writing and running JUnit tests*. Odpowiedz na pytania prowadzącego. **(10 p.)**

2 Testy jednostkowe

Testy jednostkowe testują oprogramowanie na poziomie działania pojedynczych metod. Ich główną zaletą jest możliwość wykonywania na bieżąco w pełni zautomatyzowanych testów. Dzięki temu, błędy w modyfikowanym kodzie wychwytywane są od razu gdy się pojawią. Testy jednostkowe najczęściej są pisane nie przez testerów, a przez samych programistów i w aktualnie obowiązujących metodykach tworzenia oprogramowania odgrywają bardzo istotną rolę. Mogą być też formą specyfikacji wymagań (sprawdź: *test-driven development*).

JUnit jest jednym z najpopularniejszych frameworków (szkieletów ułatwiających budowę aplikacji) umożliwiającym łatwe pisanie oraz wykonywanie testów jednostkowych dla oprogramowania tworzonego w języku Java (dla C# mamy np. NUnit). Testowanie w JUnit polega na sprawdzaniu *asercji*, czyli na weryfikacji czy metody testowanej klasy zachowują się zgodnie z oczekiwaniami. W JUnit 3 testowanej klasie `KlasaX` odpowiada klasa `KlasaXTest` dziedzicząca po *junit.framework.TestCase* i zawierająca testy dla metod klasy `KlasaX`. JUnit 4 wprowadza usprawnienia oparte głównie na mechanizmie adnotacji (ang. annotations). Większość nowoczesnych frameworków wykorzystuje adnotacje, dzięki nim rozwiązania są czytelniejsze. 10 września 2017 ogłoszono wydanie JUnit 5: `http://junit.org/junit5/`.

Zadanie 2 — Pobierz *kod źródłowy JUnit 3.8*. Zapoznaj się z metodami klasy `Assert` z pakietu *junit.framework* oraz z przykładami testów zawartymi w pakiecie *junit.samples*. Możesz też przejrzeć krótkie szkolenie do JUnit (np. [1], [2]). Dla czytelności kodu w praktyce bardzo ważne jest konsekwentne stosowanie standardowych konwencji nazewniczych, zobacz *popularne konwencje*. Wybierz jedną z konwencji i staraj się jej przestrzegać.

Zadanie 3 — Zdefiniuj klasę `Deck` przechowującą tablicę obiektów typu `Card` oraz zawierającą metody do ich tasowania, sortowania, a także metodę zwracającą kartę z góry stosu. Konstruktor klasy powinien umożliwić zdefiniowanie, czy talia ma 52 karty czy 32 karty (od siódemek) czy 24 karty (od dziewiątek). Wykorzystując JUnit 3 utwórz klasę do testowania metod klasy `Deck`. Wykorzystaj co najmniej 5 różnych metod klasy `Assert`. **(10 p.)**

Zadanie 4 — Wykorzystując JUnit 4 utwórz klasę do testowania metod klasy `Deck`. Wykorzystaj adnotacje: *@Test*, *@Test(expected = ...)*, *@Test(timeout=...)*, *@Before*, *@After*, *@Ignore*. **(5 p.)**

3 System kontroli wersji

Pliki wchodzące w skład projektu są modyfikowane przez wielu członków zespołu. System kontroli wersji to narzędzie służące do śledzenia zmian oraz pomocy w łączeniu zmian wprowadzonych przez różne osoby. Systemy kontroli wersji ze względu na architekturę można podzielić na scentralizowane, oparte na architekturze klient–serwer (np. CVS, SVN) oraz rozproszone, oparte na architekturze *peer-to-peer* (np. Git). Rozwiązania scentralizowane wykorzystują jedno główne repozytorium, z którym członkowie zespołu synchronizują swoje zmiany. Rozwiązania rozproszone umożliwiają prowadzenie wielu równorzędnych i niezależnych gałęzi, które mogą być ze sobą dowolnie synchronizowane. Poniżej znajduję się zadanie dotyczące SVN. O systemie Git będziemy mówić wkrótce.

Zadanie 5 — Znajdź w Internecie serwer umożliwiający nieodpłatne prowadzenie repozytorium SVN (np. `www.riouxsvn.com`). Zarejestruj się i utwórz nowe repozytorium. Zainstaluj w Eclipse plug-in do obsługi repozytorium SVN (np. *Help→Eclispe Marketplace→Find:* ***Subclipse***) i zapoznaj się z jego instrukcją obsługi (np. tu). Połącz się przez Eclipse ze swoim repozytorium i umieść w nim klasy napisane w ramach Zadania 3 i 4. Nadaj koledze lub koleżance uprawnienia do wprowadzania zmian w repozytorium i poproś o pobranie plików, dopisanie w Eclipse testu dla klasy `Deck` i umieszczenie zmian w repozytorium. Wraz z partnerem spróbuj wywołać konflikt a następnie go rozwiąż (patrz: tu). Przedstaw historię zmian w repozytorium prowadzącemu. **(10 p.)**

Powodzenia, J.L.