Technologia Programowania 2013/2014
Lista 1 (laboratorium)

1 Zintegrowane środowisko programistyczne

Zadanie 1 — Ze strony `www.eclipse.org` pobierz `Eclipse IDE for Java Developers`. Zapoznaj się z możliwościami tego środowiska. Przeczytaj następujący rozdział pomocy do programu: *Java development user guide* > *Getting Started* > *Basic tutorial*. Zwróć szczególną uwagę na sekcję *Writing and running JUnit tests*. Odpowiedz na pytania prowadzącego. **(3 p.)**

2 Testy jednostkowe

W aktualnie obowiązujących metodykach tworzenia oprogramowania testy jednostkowe odgrywają bardzo istotną rolę. Typowo testują one oprogramowanie na poziomie działania pojedynczych metod i są pisane nie przez testerów, a przez samych programistów. Główną zaletą testów jednostkowych jest łatwość wykonywania na bieżąco w pełni zautomatyzowanych testów na modyfikowanych elementach programu, co umożliwia wychwycenie błędu natychmiast po jego pojawieniu się. Testy jednostkowe mogą być również formą specyfikacji (*test-driven development*).

JUnit jest obecnie jednym z najpopularniejszych frameworków (szkieletów do budowy aplikacji), ułatwiającym tworzenie oraz wykonywanie testów jednostkowych dla oprogramowania tworzonego w języku Java. 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*, która zawiera testy dla metod klasy `KlasaX`. JUnit 4 wprowadza usprawnienia oparte głównie na mechanizmie adnotacji (ang. annotations).

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*. Przejrzyj krótkie szkolenia wprowadzające do JUnit 3 i JUnit 4 (np. [1], [2], [3]).

Zadanie 3 — Zdefiniuj klasę `Liczba`, która będzie przechowywała liczbę naturalna i zwracała jej zapis w dowolnym systemie o podstawie od 2 do 16. Wykorzystując JUnit 3 utwórz klasę do testowania metod klasy `Liczba`. Wykorzystaj co najmniej 5 różnych metod klasy `Assert`. **(3 p.)**

Zadanie 4 — Wykorzystując JUnit 4 utwórz klasę do testowania metod klasy `Liczba`. Wykorzystaj adnotacje: *@Test*, *@Test(expected = ...)*, *@Test(timeout=...)*, *@Before*, *@After* oraz *@Ignore*. **(2 p.)**

3 System kontroli wersji

Pliki źródłowe wchodzące w skład projektu programistycznego są na ogół modyfikowane przez wielu członków zespołu. System kontroli wersji to narzędzie służące do rejestrowania i śledzenia zmian oraz pomocy w łączeniu zmian dokonanych przez wiele osób w różnych momentach. Systemy kontroli wersji ze względu na architekturę można podzielić na scentralizowane, oparte na architekturze klient–serwer (np. CVS, Subversion) 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 (ang. *branch*), które mogą być ze sobą dowolnie synchronizowane.

Zadanie 5 — Znajdź w Internecie serwer umożliwiający nieodpłatne prowadzenie repozytorium SVN (np. `code.google.com`). Utwórz na nim nowe repozytorium. Zainstaluj w Eclipse plug-in do obsługi repozytorium SVN (np. *Help→Marketplace→**Subclipse***) i zapoznaj się z jego dokumentacją (np. tu). Umieść w stworzonym repozytorium klasy napisane w ramach Zadania 3 i 4. Nadaj koledze lub koleżance uprawnienia do wprowadzania zmian w repozytorium i poproś o dopisanie testów dla klasy `Liczba`. Przedstaw historię zmian w repozytorium prowadzącemu i odpowiedz na jego pytania. **(3 p.)**

J.L.