Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk

WYDZIAŁ INFORMATYKI

Kierunek INFORMATYKA

Studia I stopnia (dyplom inżyniera)

Język Java – laboratorium 1

dr inż. Łukasz Sosnowski
lukasz.sosnowski@wit.edu.pl
sosnowsl@ibspan.waw.pl
l.sosnowski@dituel.pl

www.lsosnowski.pl

Część 1 – administracyjna

Laboratorium z języka JAVA

Cel główny: Zdobycie podstawowych praktycznych umiejętności wykorzystywanych do tworzenia programów w JAVA

Cele szczegółowe: środowisko pracy JAVA i ECLIPSE, testy jednostkowe, projektowanie i implementowanie klas, realizacja hermetyzacji, praktyczne aspekty dziedziczenia, implementacja polimorfizmu, obsługa typów prostych, typów obiektowych, obsługa wątków, podstawy biblioteki swing i inne.

Liczba zajęć:
8 laboratoriów,

Zasady współpracy w ramach laboratoriów

1) Prezentacja tematów uzupełniających względem wykładu
2) Znajomość materiału z wykładów obowiązkowa
3) Obecność na laboratorium obowiązkowa
4) Wykonywanie ćwiczeń z praktycznych przewodników (tutorial)
5) Praca z kodem na zajęciach – rozwiązywanie problemów Programistycznych
6) Obowiązkowe prace domowe na każdych zajęciach
7) Po 2 referaty z tematów rozszerzających wiedzę o języku i narzędziach

Zaliczenie:	**Punktacja**:
2 kolokwia (4 i 7 laboratorium – zaoczne)	20 pkt (x2)
• 1 projekt do samodzielnego wykonania (tematy rozdane na zajęciach 5)	10 pkt
• Praca domowa (referaty, przewodniki)	5 pkt
• Praca na zajęciach	5 pkt

Minimum dopuszczające do egzaminu: 11 pkt

Część 2 – instalacja i konfiguracja środowiska JAVA i ECLIPSE

Przewodnik 1

instalacja i konfiguracja środowiska JAVA i ECLIPSE do pracy na zajęciach z przedmiotu Język JAVA

Plik Przewodnik1.pdf

Instalacja JDK

(Java Development Kit)

środowisko niezbędne do programowania w języku Java. Produkt dostępny jest dla wielu systemów operacyjnych, np.. Windows, Linux, Solaris

Instalacja APACHE MAVEN

narzędzie automatyzujące budowanie projektu dla platformy Java. Poszczególne funkcjonalności Mavena realizowane są poprzez wtyczki, które są automatycznie pobierane przy ich pierwszym wykorzystaniu. Plik określający sposób budowy aplikacji nosi nazwę POM.XML (ang. Project Object Model).

APACHE MAVEN (https://maven.apache.org/)

- Maven korzysta z tzw. cykli budowania projektu
- Wyróżniamy 3 podstawowe cykle: default, site, clean

clean – czyści strukturę katalogową projektu po poprzednim budowaniu

site – generuje dokumentację projektową

default – domyślny tryb budowania projektu w podziale na fazy budowania

APACHE MAVEN – wybrane podstawowe fazy budowania

- validate - weryfikuje strukturę projektu
- compile - kompiluje kod źródłowy projektu
- test - wykonuje testy jednostkowe zdefiniowane w projekcie
- package – tworzy wynikowy plik w katalogu „target” (pom.xml)
- integration-test – odpala testy integracyjne projektu
- verify - uruchamia walidację wyniku fazy package
- install - instaluje paczkę (JAR/WAR) w lokalnym repozytorium
- deploy - kopiuje paczkę do publicznego repozytorium
- Przykładowe użycie: *mvn verify , mvn clean install, mvn clean deploy*

APACHE MAVEN - Project Object Model

- Struktura pliku:
  - groupId - tag definiujący pakiet projektu (np.. pl.wit.jj)
  - artifactId – tag definiujący nazwę projektu
  - packaging – tag definiujący formę wyjściową (jar, war, etc).
  - version – tag definiujący wersję naszego projektu
  - name – tag opcjonalny odpowiedzialny za wyświetlaną nazwę
  - url – opcjonalny tag z linkiem do strony projektu
  - dependencies - opcjonalny tag definiujący zależności od innych modułów

Instalacja ECLIPSE

(Integrated Development Environment)

Środowisko okienkowe ułatwiające tworzenie kodu programu, kompilacji oraz zarządzanie różnymi procesami podczas tworzenia oprogramowania. Umożliwia również tzw. „odpluskwianie” kodu (debug) w postaci wizualnego podglądu wartości obiektów i zmiennych.

Część 3 – tworzenie projektu, prace przygotowawcze

Język Java laboratorium – dr inż. Łukasz Sosnowski

Przewodnik 2

generowanie projektu z pakietu MAVEN, import do Eclipse, tworzenie klasy testu jednostkowego

Plik Przewodnik2.pdf

Generowanie szablonu projektu z obsługą testów jednostkowych

Utworzenie struktury projektu z testowymi klasami oraz przykładowym testem jednostkowym. Generowanie odbywa się z poziomu konsoli (CMD), natomiast robi się to relatywnie rzadko, na początku projektu.

Importowanie projektu MAVEN'owego do ECLIPSE

Utworzony projekt należy zaimportować, wskazując ECLIPSE'owi plik pom.xml projektu, z którego zostaną odczytane zależności bibliotek, nazwa, pakietowanie, itp.

W różnych przypadkach poza importem projektu konieczne może być przebudowanie poprzez wtyczkę MAVEN'a w ECLIPSE. Aby tego dokonać należy zaznaczyć projekt, kliknąć prawy przycisk myszy i wybrać Maven → Update Project

Tworzenie testu jednostkowego

Procedura tworzenia klasy testu jednostkowego jest relatywnie prosta. Wymaga wybrania odpowiedniej opcji w menu File →New → JUnit Test Case

Jednakże stworzenie poprawnego i przydatnego testu jednostkowego wymaga znajomości problemu testowanego oraz ogólnej wiedzy o testach jednostkowych

Część 4 – testy jednostkowe

Testy jednostkowe

Test jednostkowy (ang. unit test) do pewna konwencja testowania oprogramowania poprzez pisanie programów, które testują inny fragment oprogramowania. Konwencja ta polega na wydzielaniu mniejszych części funkcjonalnych (jednostkowych) i testowaniu poszczególnych takich składowych oddzielnie. W praktyce będzie to pojedyncza klasa lub metoda, która podlega testowaniu. W celu ułatwienia pisania testów używamy pakietu jUnit, który umożliwia nam również prostą integrację testów jednostkowych w proces budowania projektu (maven).

Testy jednostkowe

Klasa testu jednostkowego pakietu jUnit posiada kilka użytecznych metod:

```
public static void setUpBeforeClass() throws Exception
public static void tearDownAfterClass() throws Exception
public void setUp() throws Exception
public void tearDown() throws Exception
```

- **setUpBeforeClass** – metoda służąca do globalnej inicjalizacji testu, uruchamiana na początku procesu uruchomienia testów
- **tearDownAfterClass** – metoda służąca do „posprzątania” po wszystkich testach, uruchamiana na sam koniec procesu odpalania testów
- **setUp** – metoda służąca do inicjalizacji wartości potrzebnych do realizacji testów przed każdym testem jednostkowym
- **tearDown** – metoda służąca do „posprzątania” po pojedynczym teście

Testy jednostkowe

- Pojedynczy test jednostkowy to metoda w klasie testu opatrzona adnotacją „@Test”
- W celu napisania testów używamy tzw. asercji (ang. assert), czyli metody spodziewającej się konkretnego wyniku
- Przykłady dostępnych asercji:
  - **assertTrue**: sprawdza czy wartością przekazanego argumentu jest PRAWDA,
  - **assertFalse**: sprawdza czy wartością przekazanego argumentu jest FAŁSZ,
  - **assertNull**: sprawdza czy przekazany argument jest null,
  - **assertNotNull**: sprawdza czy przekazany argument nie jest null,
  - **assertEquals**: sprawdza czy wartość spodziewana i wartość rzeczywista przekazane w argumentach są identyczne, jeśli są różne rzuca wyjątek,
  - **assertNotEquals** sprawdza czy przekazane w argumentach referencje wskazują na ten sam obiekt
  - **fail** – kończy test niepowodzeniem z z opcjonalnym przekazaniem komunikatu

Część 5 – ćwiczenia praktyczne

Ćwiczenia praktyczne

- Klasa Point do reprezentacji punktu na płaszczyźnie
- Dodanie konstruktora parametrycznego 2 argumentowego
- Metoda w klasie Point realizująca przesunięcie dla osi X
- Metoda w klasie Point realizująca przesunięcie dla osi Y
- Metoda w klasie Point realizująca dodawanie punktów
- Metoda w klasie Point realizująca odejmowanie punktów
- Metoda w klasie Point zwracająca referencję do nowego obiektu po wykonaniu operacji dodania punktów
- Testy jednostkowe sprawdzający klasę i jej metody
- Zbudowanie projektu poprzez maven

Podsumowanie

- Instalacja środowiska JDK, MAVEN, Eclipse
- Tworzenie szablonów projektów z użyciem MAVEN
- Tworzenie testów jednostkowych z użyciem jUnit
- Przykłady implementacji klas, metod
- Przykłady testów jednostkowych dla klas
- Realizacja hermetyzacji na przykładzie klasy Punkt

Praca domowa:

- Obowiązkowe wykonanie zadania domowego: Zadanie1.pdf
- Referat na temat: „Maven – konfiguracja i użycie”
- Referat na temat „jUnit 4.0 – pisanie testów jednostkowych w JAVA”

Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk

WYDZIAŁ INFORMATYKI

Kierunek INFORMATYKA

Studia I stopnia (dyplom inżyniera)

Dziękuję za uwagę!