

Wykład I

Programowanie zorientowane obiektowo

Literatura:

- Cay S. Horstmann, Gary Cornell - Java. Podstawy. Wydanie IX,
- Marcin Lis - Java. Ćwiczenia praktyczne. Wydanie IV
- Bruce Eckel - Thinking in Java. Edycja polska. Wydanie IV
- Dr Alex Blewitt - Eclipse 4. Programowanie wtyczek na przykładach
- Krzysztof Barteczko - Java Programowanie praktyczne od podstaw

Eclipse

<https://eclipse.org/downloads/> - Eclipse IDE for Java EE Developers

Środowisko programowania w języku Java

Java - Java Platform Standard Edition - (JDK - Java SE Development Kit) 8

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Programowanie w języku Java wymaga środowiska programowania JDK

Pakiet ten zawiera JRE - Java Runtime Environment - Środowisko w którym zostanie uruchomiony program napisany w języku Java.

Środowisko **uruchomieniowe** programów napisanych w języku Java - JRE składa się z maszyny wirtualnej Javy - JVM oraz biblioteki zawierającej klasy podstawowe i dodatkowych plików odpowiedzialnych za uruchomienie programu.

Wirtualna maszyna Javy (Java Virtual Machine) jest środowiskiem, które na podstawie kodu bajtowego Javy w danym środowisku charakteryzowanym przez system operacyjny i architekturę sprzętową jest w stanie wytworzyć kod binarny realizowany przez procesor.

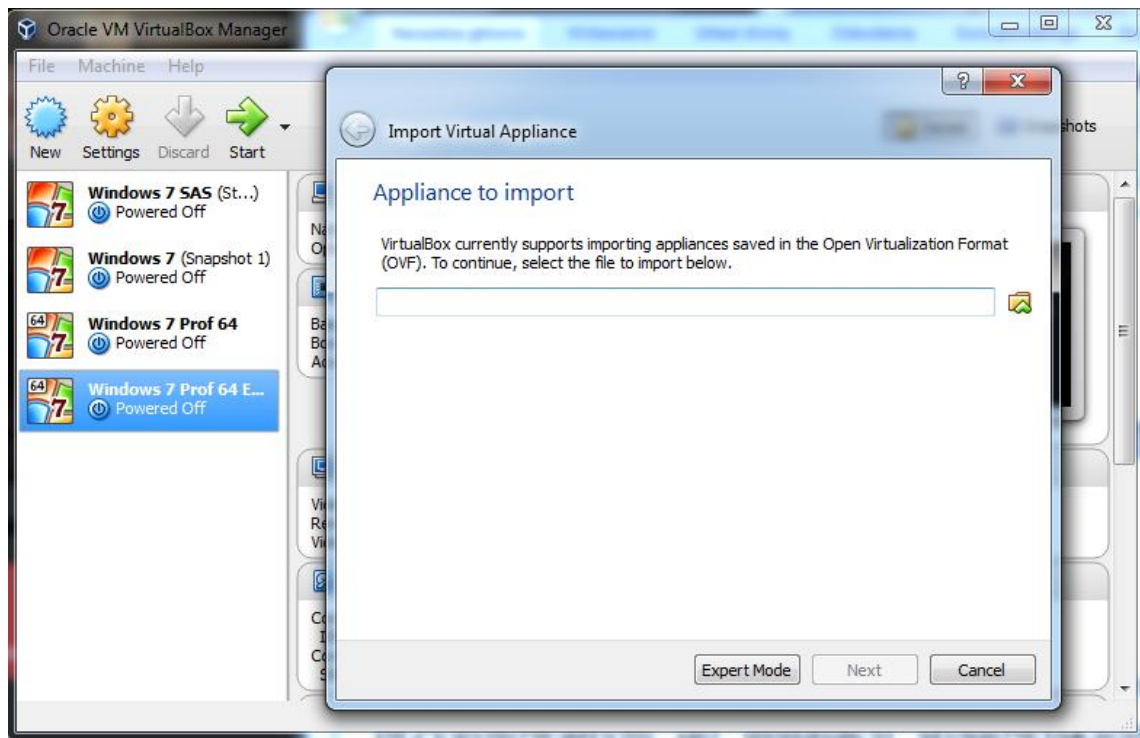
Podstawową zaletą tego rozwiązania jest wieloplatformowość - praktycznie dla większości systemów operacyjnych i architektur sprzętowych dostępna jest wirtualna maszyna Javy - JVM

Środowisko tworzenia aplikacji napisanych w języku Java.

Przygotowane zostało środowisko w postaci maszyny wirtualnej VirtualBox do zaimportowania w otwartym standardzie *ova* . Maszyna wirtualna jest dostępna w udziale sieciowym `\\perseus\pub\pzo\maszyna wirtualna`

Maszyna wirtualna zawiera system Windows 7 z zainstalowanym środowiskiem JDK i

przygotowanym do instalacji środowiskiem Eclipse. Maszynę w standardzie OVA należy zaimportować do środowiska VirtualBox



Licencja systemu Windows jest pobrana z systemu MSDN AA.

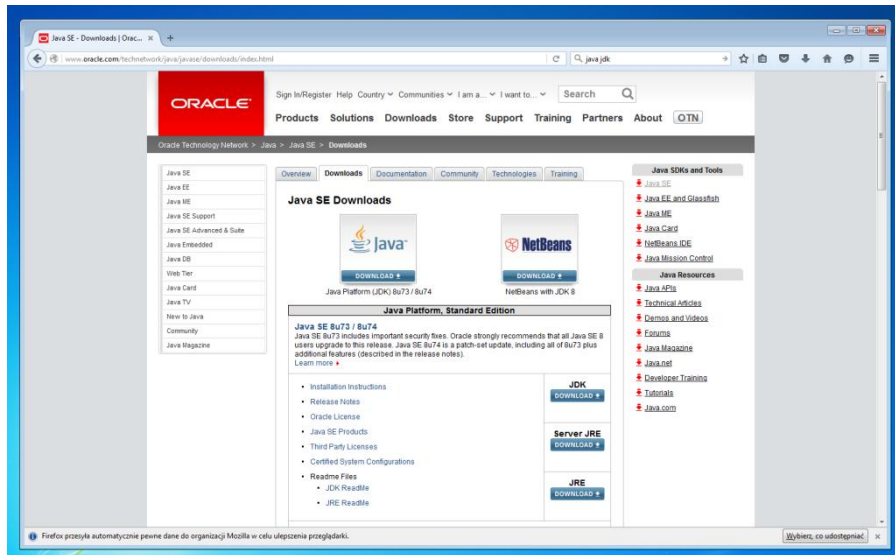
Hasło dla użytkownika student to: Zaq12wsx

+

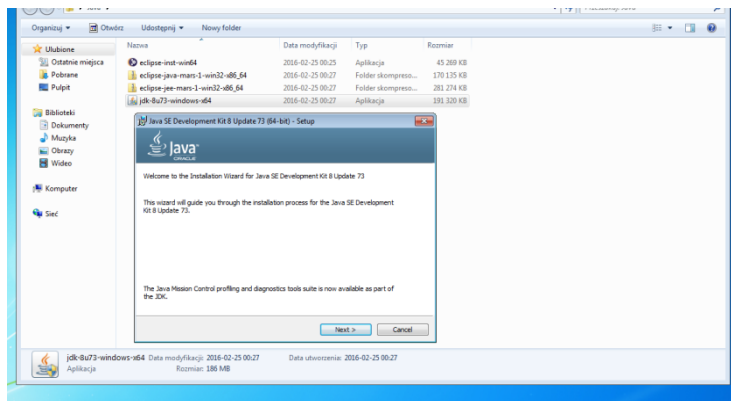
Instalacja środowiska JDK

Ze strony <http://www.oracle.com/technetwork/java/javase/downloads/index.html> należy pobrać wersję dostosowaną do systemu operacyjnego.

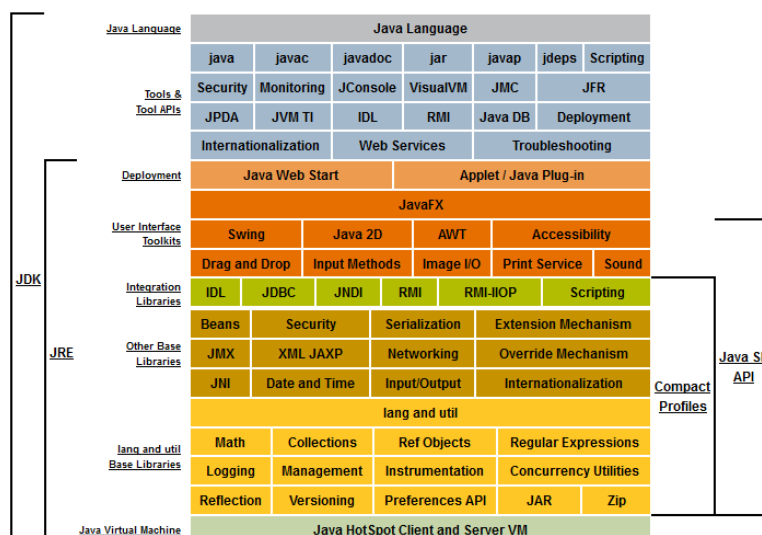
Dostępne są wersje dla systemu Linux z procesorami ARM, x86, x64, Mac OS X, Solaris z procesorami SPARC, x64 oraz platforma Windows x86 i x64



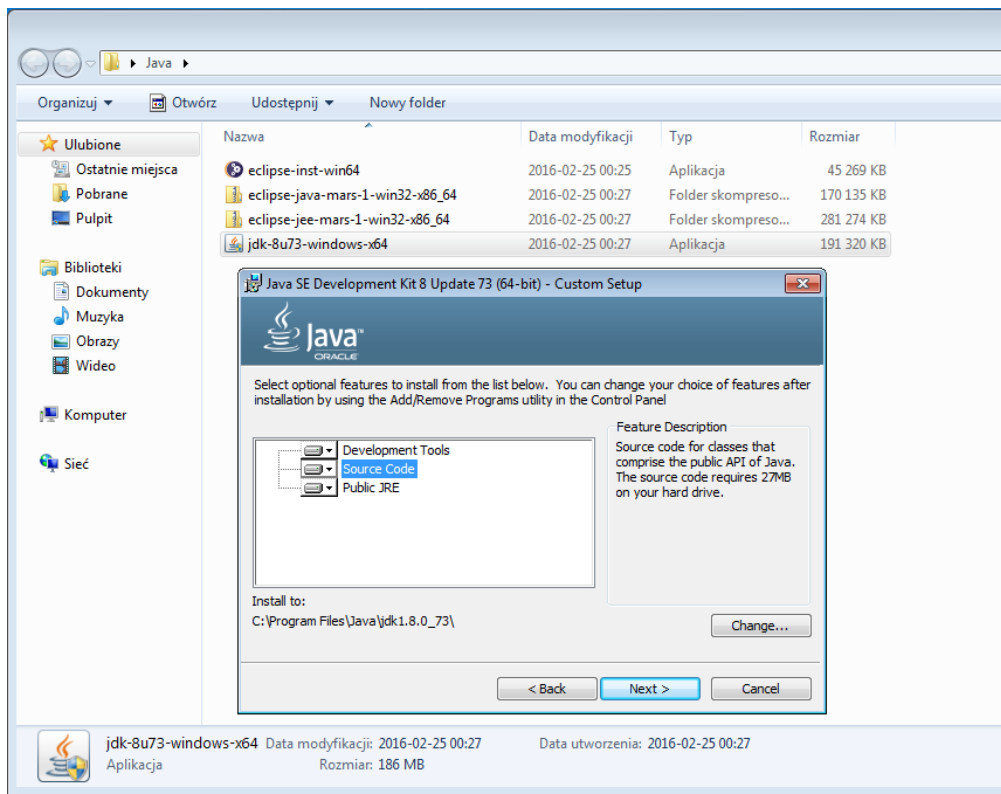
Instalacja wersji dla Windows x64



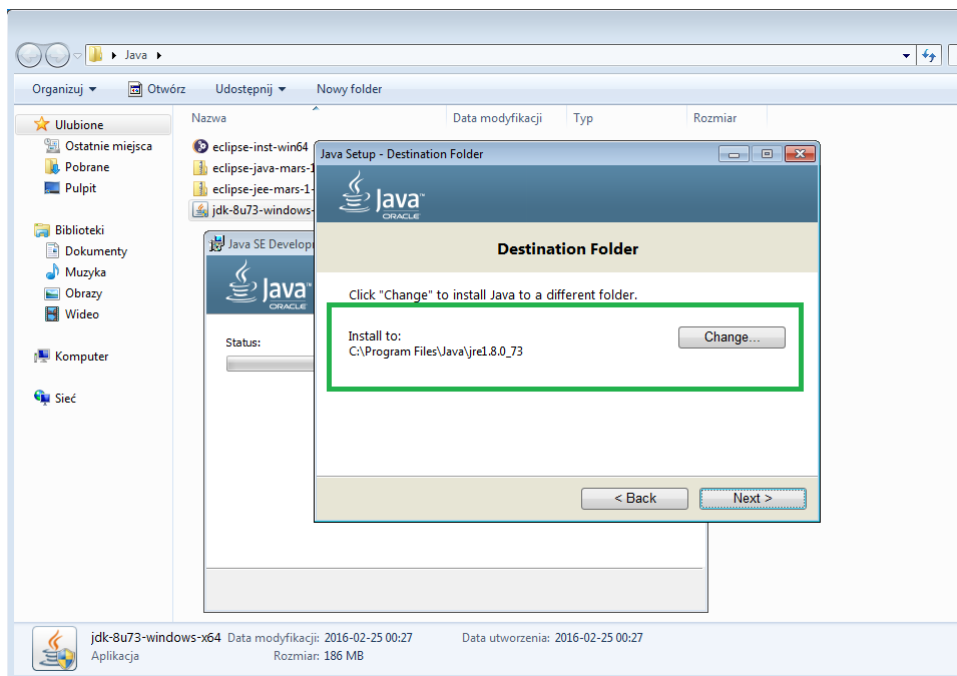
Platforma JDK i jej elementy



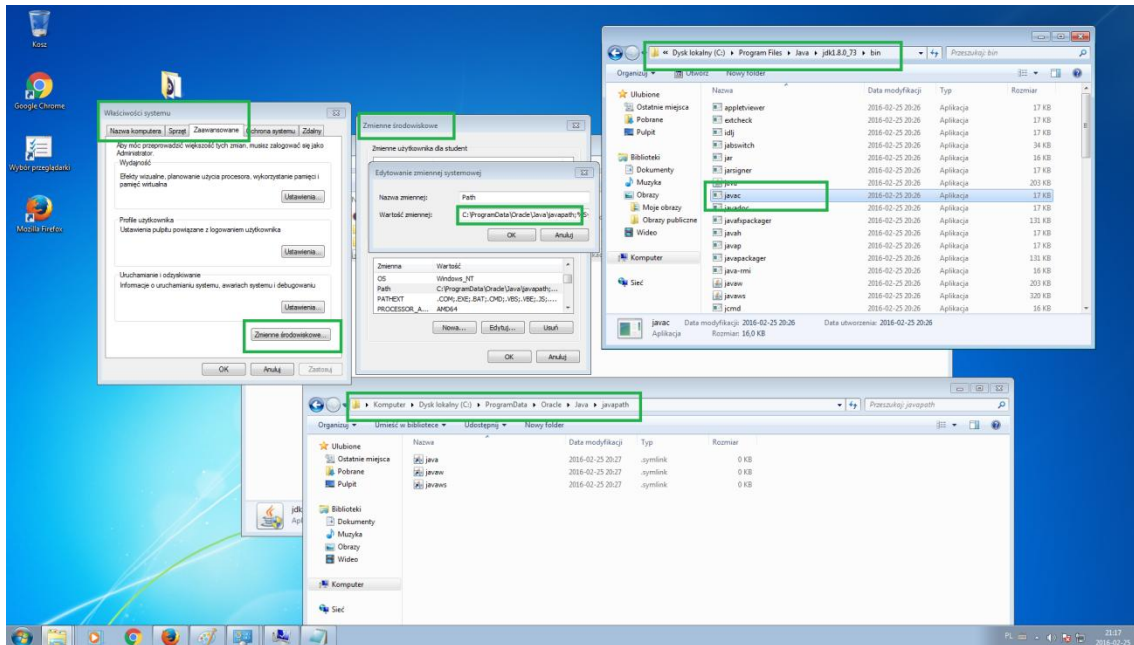
Instalacja elementów środowiska



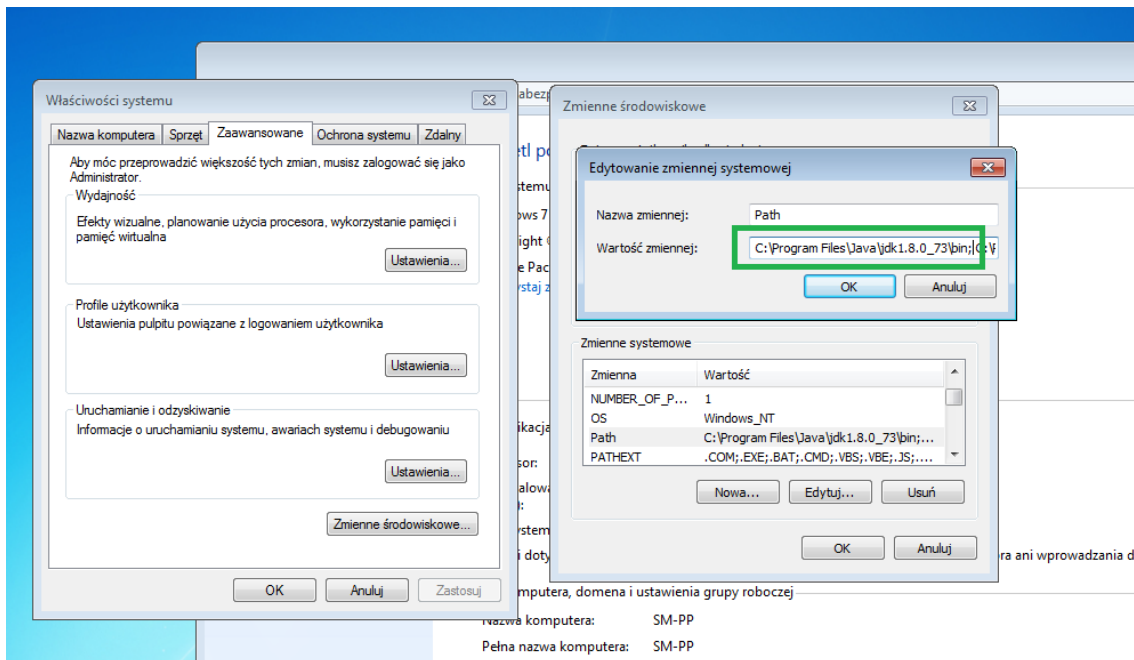
Domyślny katalog instalacji JDK



Zmienne środowiskowe po instalacji JDK w wersji 8



Dodanie ścieżki do katalogu zawierającego plik javac.exe



Zalety programowania w środowisku Java:

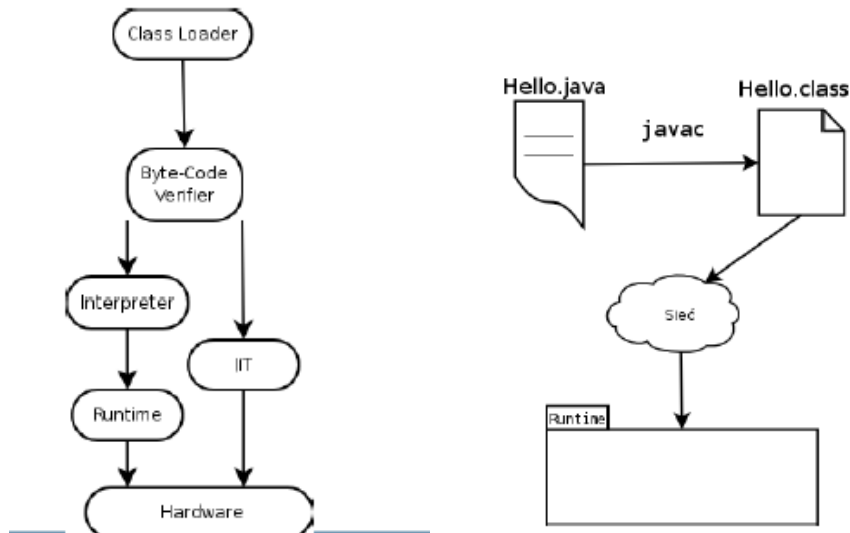
- Przenośność - raz napisany kod można przenieść do dowolnej JVM w dowolnym środowisku
- Zorientowanie Obiektowe - Java powstała jako język obiektowy
- Bardzo obszerne biblioteki standardowe, tworzenie aplikacji z gotowych komponentów
- Uniwersalne środowisko dostępu do GUI i multimediów
- Uniwersalne środowisko dostępu do baz danych
- Uniwersalne środowisko programowania aplikacji sieciowych
- Wielowątkowość realizowana na poziomie języka
- Programowanie rozproszone realizowane na poziomie języka
- Bezpieczne i kontrolowane środowisko wykonania programu - JVM.

Program w języku Java

- Program w języku Java jest zestawem klas. Podstawową jednostką enkapsulacji jest klasa. Kod języka można tylko wprowadzać w obrębie klasy.
- Kod źródłowy programu zapisywany jest w plikach o rozszerzeniu .java
- Nazwa pliku powinna być zgodna z nazwą klasy. W pliku może występować wiele klas, wtedy zgodność nazwy dotyczy klasy głównej.
- Tworzone programy w języku Java to:
 - aplikacje (standalone programs)
 - aplety (applets)
- Aplikacje - działają w trybie tekstowym lub w trybie graficznym - do uruchomienia wymagana jest maszyna wirtualna Javy w danym systemie operacyjnym - JVM.
- Aplety działają wyłącznie w trybie graficznym w środowisku przeglądarki internetowej oczywiście jeżeli ta ma zaimplementowaną zintegrowaną wirtualną maszynę Javy

Konstruowanie programów w języku Java

- Pliki kodu źródłowego *.java zawierają kod źródłowy. Nazwa pliku to nazwa klasy publicznej zapisanej w pliku.
- Kompilacja - Pliki źródłowe są kompilowane do kodu bajtowego (Java byte code). Kompilacja jest wykonywana za pomocą kompilatora *javac*.
 - Polecenie: *javac nazwa_klasy_publicznej.java*
 - Plik wynikowy zawierający J-code będzie miał nazwę *nazwa_klasy_publicznej.class*
- Plik wynikowy z rozszerzeniem class może zostać wykonany przez JVM, czyli przekształcony na binarne rozkazy dla procesora realizowane na określonej platformie sprzętowej i w określonym systemie operacyjnym. Uruchomienie maszyny wirtualnej i wykonanie programu umożliwia polecenie.
 - Polecenie uruchamiające program z kodu bajtowego do binarnego programu wykonywalnego to:
java nazwa_klasy_publicznej.class argument1 argument2
 - Interpreter na podstawie kodu bajtowego tworzy kod wykonywalny
 - Możliwe jest przyspieszenie działania aplikacji w języku Java dzięki metodzie JiT - (Just-In-Time compilation) poprzez utworzenie kodu wykonywalnego bezpośrednio z kodu źródłowego przy wywołaniu klasy.



Tworzenie aplikacji w języku Java

Tworzenie Aplikacji (application) - program uruchamiany w systemie operacyjnym w maszynie wirtualnej JVM.

W kodzie programu musi znajdować się klasa publiczna zawierająca metodę publiczną

- `public static void main(String args[])`

Kod źródłowy pierwszej aplikacji w języku Java

```
public class Program
{ // Wyświetla komunikat
public static void main ( String[] args)
{
System.out.println ("Pierwszy program ");
}
}
```

Powyższy kod należy zapisać w pliku Program.java

Kompilacja

Dokonać kompilacji do kodu bajtowego

```
javac Program.java
```

Zostanie wygenerowany plik Program.class do wykonania przez maszynę wirtualną Java - JVM

Interpretacja

Uruchomienie maszyny wirtualnej i przekazanie kodu bajtowego realizowane jest poleceniem:

```
java Program
```

Kolejność wykonywanych czynności Interpreter java

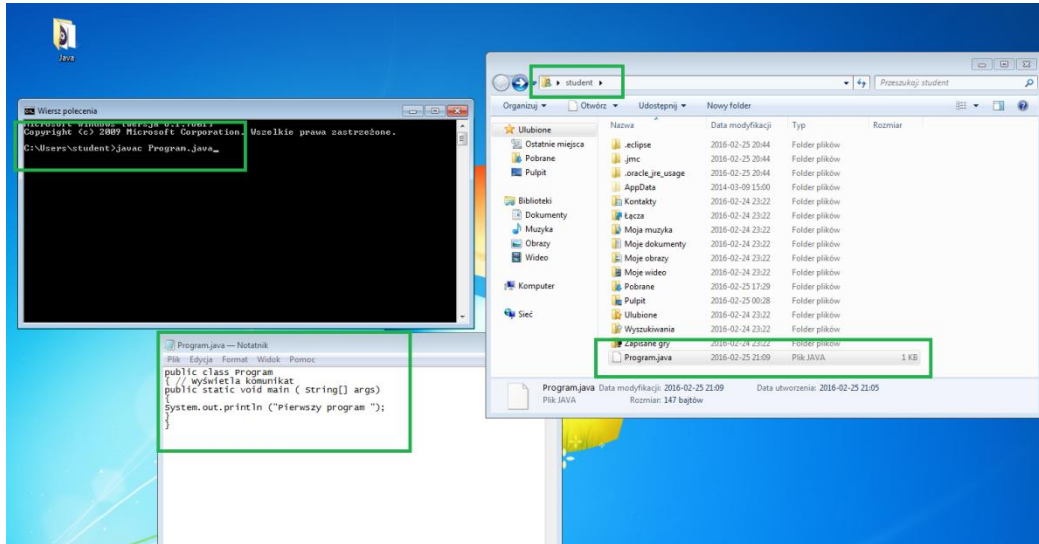
Zakładamy że zmienne środowiskowe path są tak ustawione że nie potrzebujemy wskazywać katalogów zawierających programy javac i java

- wyszuka plik o nazwie Program.class w katalogu bieżącym
- sprawdzi, czy klasa Program posiada publiczną metodę statyczną main
- wykona instrukcje zawarte w bloku metody main, czyli wyświetli na ekranie napis

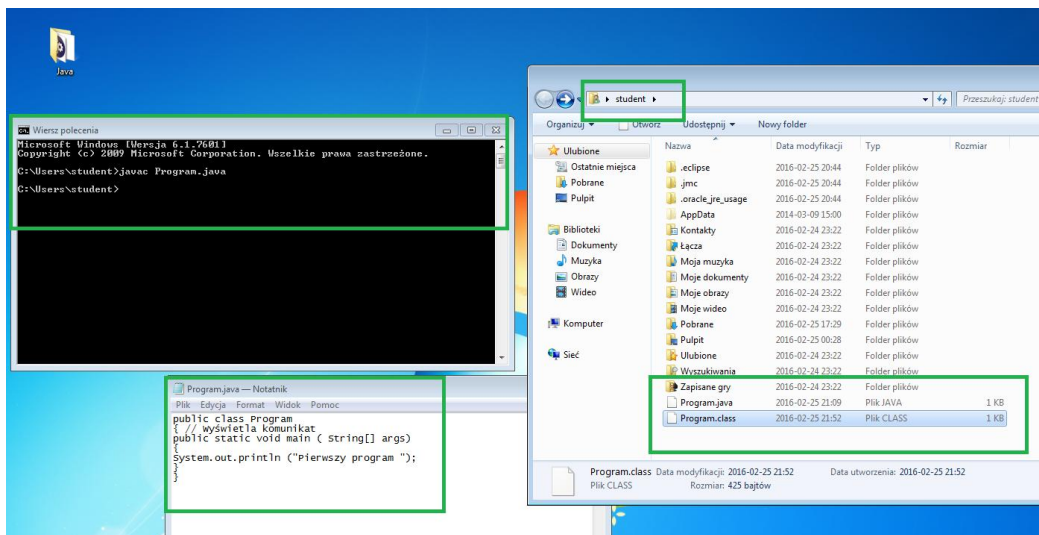
Pierwszy program

- do metody main z wiersza poleceń jest przekazywana tablica argumentów args obiektów (łańcuchów) klasy String, jako parametr wywołania interpretera - w naszym programie nie występuje.
- każda instrukcja kończy się średnikiem.

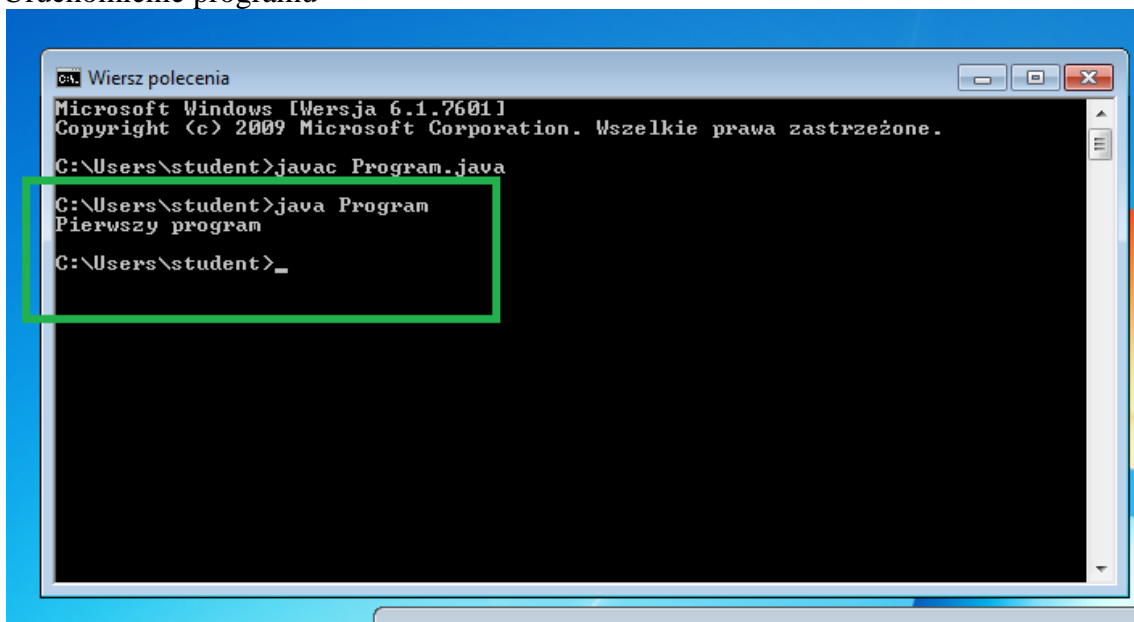
Kompilacja programu



Powstanie pliku Program.class



Uruchomienie programu



Pakiety Javy

- Środowisko Java oferuje i umożliwia wykorzystanie pakietów które są bibliotekami klas, każda klasa w Javie należy do określonego pakietu.
- Utworzona w naszym programie klasa Program należy do domyślnego pakietu definiowanego przez środowisko.
- Pakiety grupują i porządkują klasy uniemożliwiają pojawienie się kolizji nazw klas. Jednym ze standardowych pakietów, nie wymagających deklaracji, jest pakiet `java.lang`, zawierający główne klasy języka Java.

Klasa `System` pakietu `java.lang` wykorzystano pole `out` związane ze standardowym wyjściem na konsolę

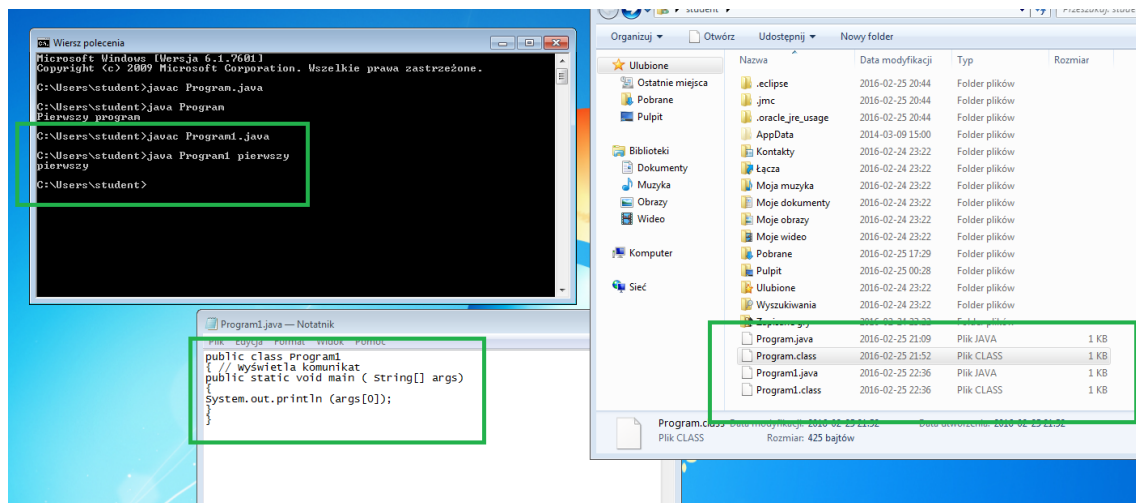
Metoda `println()` wyświetla łańcuch znaków podany jako argument.

Przekazywanie parametrów do aplikacji w języku Java

```
public class Program1
{ // Wyświetla komunikat
public static void main ( String[] args)
{
System.out.println (args[0] );
}
}
```

W linii poleceń zostanie po wywołaniu programu zostaną podane argumenty, a właściwie ich macierz.

java Program1 pierwszy



Pobieranie wartości zmiennych od użytkownika w języku Java

Do pobierania danych podczas wykonywania programu możemy wykorzystać bibliotekę klas System.in. W ramach tej biblioteki do pobierania danych z konsoli służy obiekt Scanner. Klasa Scanner musi być zadeklarowana i zainicjowana

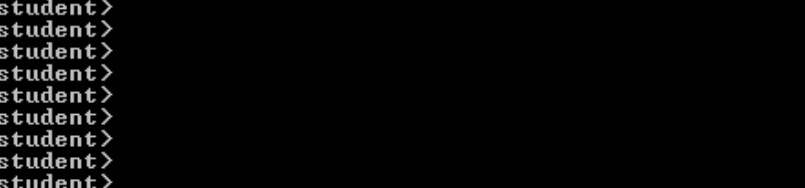
Przykład:

```
import java.util.Scanner;
public class Program2 {

    public static void main(String[] args) {
        //deklaracja zmiennej w której przechowywane będą dane pobrane od użytkownika.
        String tekst_1;
        //deklaracja obiektu pobrane_dane który będzie pobierał dane z konsoli
        Scanner pobrane_dane;
        //inicjalizacja obiektu pobrane_dane
        pobrane_dane = new Scanner(System.in);
        //pobranie danych tekstowych ze standardowego wejścia
        tekst_1 = pobrane_dane.nextLine();
        //wyswietlenie pobranych danych
        System.out.println (tekst_1 );
    }
}
```

Kompilacja programu `javac Program2.java`

Uruchomienie programu **java Program2**



```
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>
C:\Users\student>javac Program2.java
C:\Users\student>java Program2
Wpisz tekst
Wpisany tekst
Wpisany tekst
C:\Users\student>
```

Tworzenie apletu w języku Java

W przypadku apletu musi zostać użyte środowisko graficzne, aplet musi zostać skompilowany do postaci pliku `.class` lub `.jar`.

Plik apletu jest wykonywany przez wirtualną maszynę Javy zainstalowaną wewnątrz przeglądarki. Odpowiednie środowisko dla przeglądarek jest instalowane ze strony <https://www.java.com/pl/>



Kod źródłowy apletu nie musi zawierać metody `public static void main (String[] args)` i w najprostszym przypadku będzie następujący:

```
import java.awt.*;
import java.applet.*;
public class Programg extends Applet
{ // Wyświetla komunikat
public void paint (Graphics g)
{
g.drawString("Pierwszy program ", 10, 20);
}
}
```

Aplety mogą być uruchamiane z wykorzystaniem dwóch metod:

- za pomocą znaczników `<applet>` w kodzie HTML
- Za pomocą JNLP (Java Network Launch Protocol)

Użyjemy prostszej metody osadzenia w kodzie HTML

Najprostszy wymagany kod zawiera podane niżej elementy

```
<HTML>
```

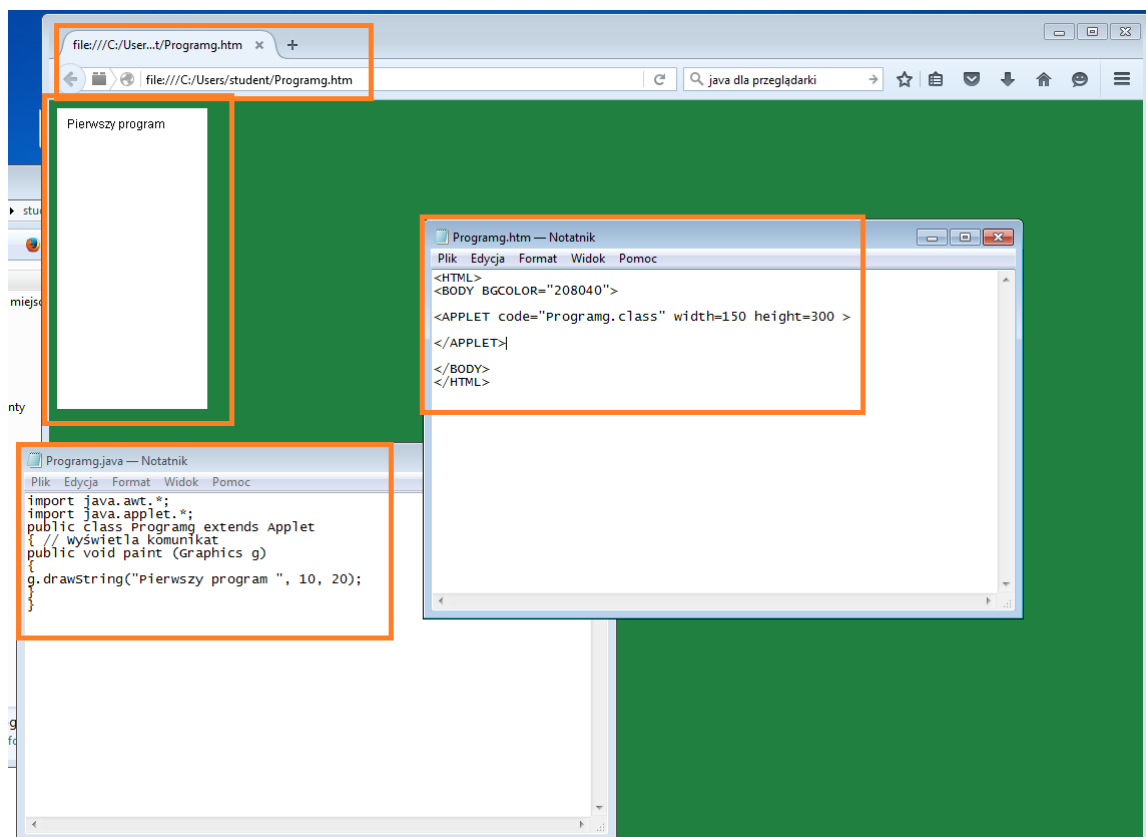
```
<BODY BGCOLOR="208040">
```

```
<APPLET code="Programg.class" width=150 height=300 >
```

```
</APPLET>
```

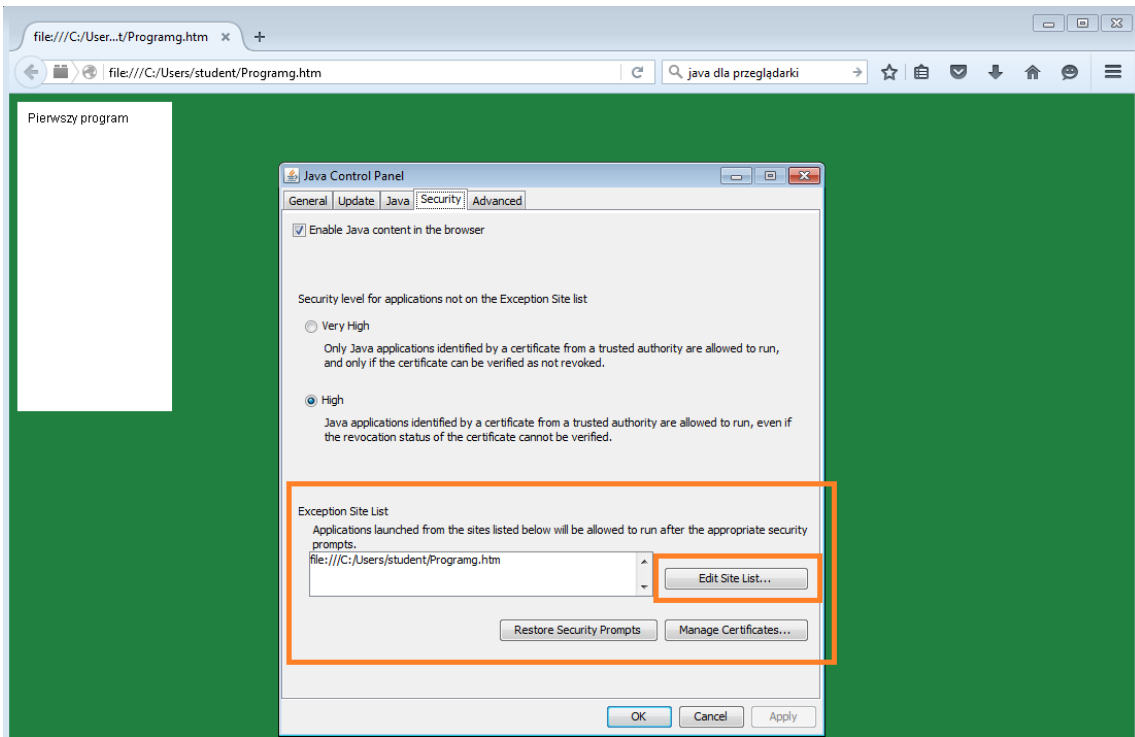
```
</BODY>
```

```
</HTML>
```



Ze względów bezpieczeństwa wyświetlanie apletu z nie zaufanych stron innych niż HTTPS jest ograniczane. Należy dodać ścieżkę do naszego pliku zawierającego dokument HTML do wyjątków bezpieczeństwa.

W Panelu sterowania należy wyszukać program do zarządzania środowiskiem Java - Java Control Panel w zakładce Security dodać URL do naszego pliku do listy wyjątków.



Komentarze i tworzenie dokumentacji

//Komentarz w jednym wierszu

/*Komentarz blokowy. Służy do obszerniejszego opisu programu komunikaty – kilka bądź kilkanaście wierszy.*/

/*

* A to komentarz dokumentacyjny.

* W taki sposób umieszczamy notatki,

* wykorzystywane później do łatwego opracowania

*dokumentacji projektowej.

*/

Tworzenie dokumentacji

Do opisu kodu źródłowego stosowane są komentarze.

Należy opisywać klasy, metody, interfejsy i moduły, oraz zmienne

Komentarz powinien poprzedzać komentowany kod źródłowy.

W środowisku Java można tworzyć dokumentację w formacie HTML za pomocą programu *javadoc*.

W programie *javadoc* rozpoznawane są komentarze, jeżeli zostały one umieszczone pomiędzy sekwencjami znaków */*** i **/*. Znaki *** w kolejnych wierszach są pomijane.

Podział dokumentu na paragrafy uzyskujemy umieszczając znak *@* na początku wiersza z jednym ze zdefiniowanych znaczników dokumentacyjnych.

Polecenie wygenerowania dokumentacji ma postać:

javadoc nazwa_pliku.java

Jego wynikiem jest zbiór plików z opisem w formacie HTML

Znaczniki dokumentacyjne javadoc

@author – informacje o autorze programu,

@version – informacje o wersji programu,

@return – opis wyniku zwracanego przez metodę,

@serial – opis typu danych i możliwych wartości przyjmowanych przez zmienną,

@see – tworzy łącze do innego tematu,

@since – opis wersji, od której zaistniał określony fragment kodu,

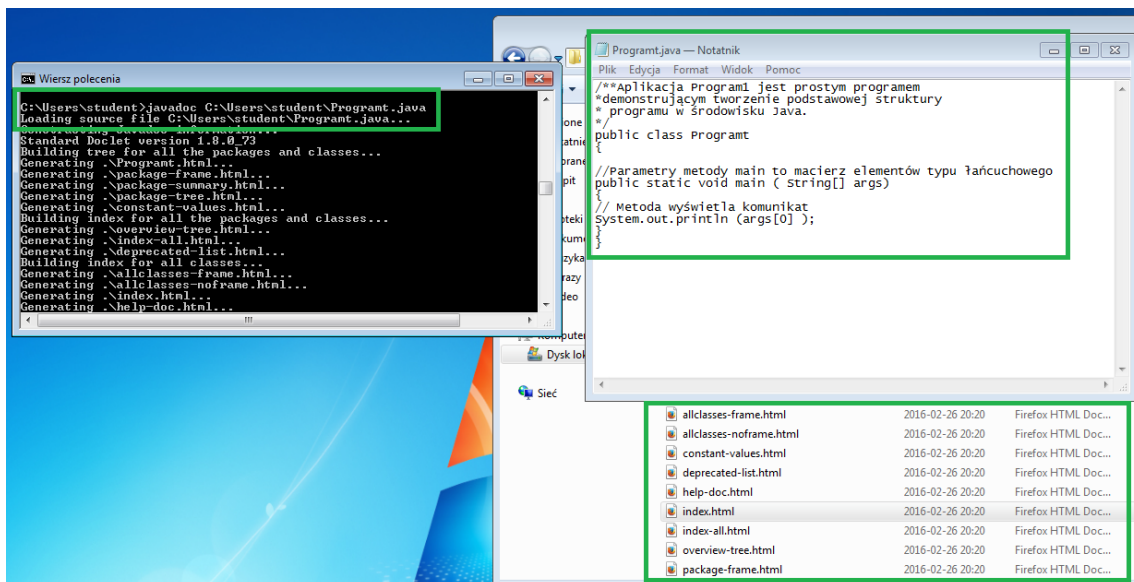
@deprecated – informacje o elementach zdeprecjonowanych (które nie są zalecane),

@param – opis parametru wywołania metody,

@exception – identyfikator wyjątku

Przykład tworzenia dokumentacji zawartość pliku Programt.java

```
/** Aplikacja Programt jest prostym programem
 *demonstrującym tworzenie podstawowej struktury
 * programu w środowisku Java. */
public class Programt {
    public static void main ( String[] args)
    { // Metoda wyświetla komunikat
      System.out.println (args[0] );
    }
}
```



Wygenerowana dokumentacja w języku HTML

