

# Programowanie obiektowe - Java

dla studentów II roku Wydziału Fizyki PW

**Sławomir Ertman, 2014**

Kontakt: e-mail: [ertman@if.pw.edu.pl](mailto:ertman@if.pw.edu.pl)

Konsultacje: pn. 12.15-13, czw. 14.15-15 pok. 8A GF

Materiały z wykładu i instrukcje do laboratorium:

<http://www.pojava.fizyka.pw.edu.pl>

Wykład – w programie 15 godzin, co dwa tygodnie

Laboratorium – co dwa tygodnie część A (7 ćwiczeń) i część B (projektowa)

Część A – dr inż. Daniel Budaszewski, mgr inż. Bartosz Maksiak, mgr inż. Daniel Wielanek, mgr.inż Krzysztof Dynowski

Część B – mgr inż. Jacek Bzdak, dr Sławomir Ertman

Podział na grupy i terminy Laboratorium na stronie:

<http://www.if.pw.edu.pl/~labkomp/>

# Zasady zaliczania przedmiotu „Programowanie Obiektowe -Java” w roku akademickim 2012/2013

## **Laboratorium** 7 ćwiczeń w środowisku Linux

Ocena z laboratorium jest średnią 7 ocen cząstkowych oraz z jednego kolokwium sprawdzającego wiedzę teoretyczną (na ostatnich zajęciach).

Nieobecność nieusprawiedliwiona lub nieoddanie programu daje ocenę cząstkową 0.

Nieobecności usprawiedliwione można odrobić w grupach równoległych.

## **Zadania w dwóch wersjach**

–**wersja podstawowa** (na ocenę max 3.5) powinna zrealizowana być w zasadzie w czasie zajęć laboratoryjnych .

–**wersja pełna** (na ocenę 5) realizowana jest samodzielnie i zaliczana na początku następnych zajęć. (po dwóch tygodniach).

**Projekt** - zadanie realizowane zespołowo.

**Ocena końcowa liczona będzie jako średnia ważona z laboratorium i projektu .**

**Laboratorium - waga 2, projekt - waga 1**

## **Uwaga1**

Osoby, które potrafią programować w Java - po napisaniu programu sprawdzającego przedmiot mogą zaliczać na indywidualnych zasadach.

## **Uwaga2**

Osoby, dla których programowanie *"nie jest powołaniem życiowym"* mogą zaliczyć przedmiot pracując w zespole dwuosobowym (razem z kimś bardziej zaawansowanym w programowaniu) i pisząc dokładne sprawozdania do każdego z oddawanych programów. W takim trybie można otrzymać maksymalnie ocenę 3.0

# Literatura

- **„Java – przewodnik dla początkujących”, Herbert Schildt**
- **„Thinking in Java” - Bruce Eckel**
- **„Java Receptury” - Ian F. Darwin**
- **„Java ćwiczenia praktyczne” - Marcin Lis**
- **„Java w zadaniach” - Steve Potts**
- **„Java po C++” - Jan Bielecki**
- **„Java 4 Swing” - Jan Bielecki**
- **Dokumentacja JAVA**

# Dokumentacja on-line (javadoc): <http://docs.oracle.com/javase/>

The screenshot shows the Oracle Java SE Technical Documentation website. The browser's address bar displays `docs.oracle.com/javase/`. The page features the Oracle logo and navigation links. The main heading is "Java SE Technical Documentation". Below it, the section "Java Platform, Standard Edition (Java SE)" is highlighted. Under this section, there are links for "Java SE 7" (described as the current release), "API Documentation", "Java Language and Virtual Machine Specifications", "Developer Guides", "JDK / JRE Installation Instructions", "JDK Release Notes", and "JDK Troubleshooting Guide". There is also a link for "Java Tutorials" described as a practical guide. A "Previous Releases" link is located at the bottom left. On the right side, there is a search bar, a link to "Advanced Search", and a section titled "Other Java Documents" containing links for Java EE, Java ME, Java DB, and JavaFX. Below that is a "Contact Us" section with a link for Java SE. The Windows taskbar at the bottom shows various application icons and the system clock indicating 22:10 on 2013-02-25.

Opera Java SE Technical Docu... +

Web docs.oracle.com/javase/ Szukaj w Google

**ORACLE**

Oracle Technology Network > Java SE > Java SE Documentation

## Java SE Technical Documentation

### Java Platform, Standard Edition (Java SE)

**Java SE 7**  
Java SE 7 is the current release of the Java SE platform.

- [API Documentation](#)
- [Java Language and Virtual Machine Specifications](#)
- [Developer Guides](#)
- [JDK / JRE Installation Instructions](#)
- [JDK Release Notes](#)
- [JDK Troubleshooting Guide](#)

**Java Tutorials**  
A practical guide to the Java language containing hundreds of complete working examples.

**Previous Releases**

### Java SE for Embedded

Java SE for Embedded is based on Java SE and provides specific features and support for embedded systems. View [technical documentation](#).

Search

[Advanced Search](#)

#### Other Java Documents

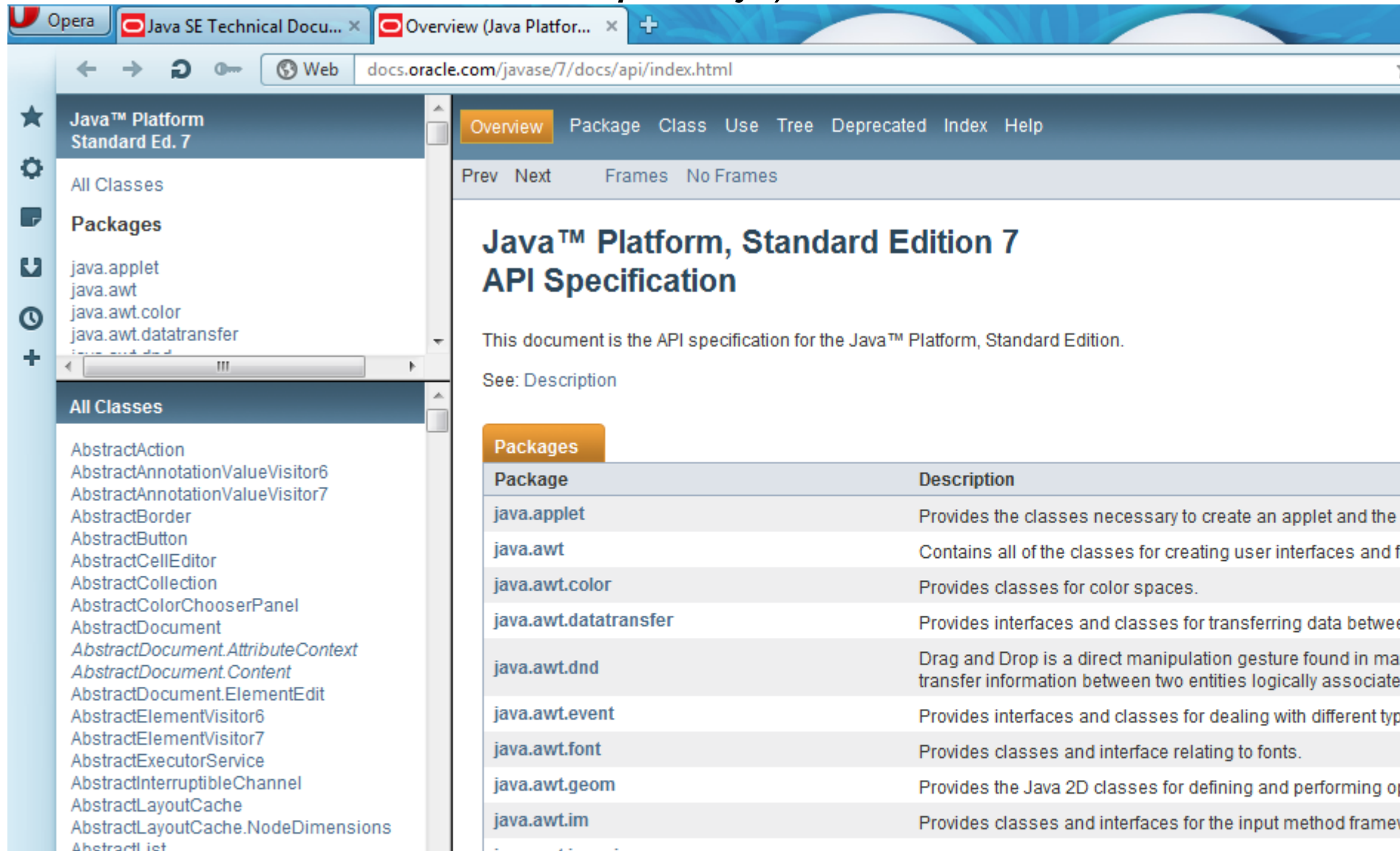
- Java EE
- Java ME
- Java DB
- JavaFX

#### Contact Us

- Java SE

# API Documentation

(*Application Programming Interface - interfejs programowania aplikacji*)



Opera Java SE Technical Docu... Overview (Java Platfor... +

Web docs.oracle.com/javase/7/docs/api/index.html

Java™ Platform Standard Ed. 7

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

## Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

### Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the
java.awt	Contains all of the classes for creating user interfaces and f
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in ma transfer information between two entities logically associate
java.awt.event	Provides interfaces and classes for dealing with different typ
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing op
java.awt.im	Provides classes and interfaces for the input method framev

All Classes

- AbstractAction
- AbstractAnnotationValueVisitor6
- AbstractAnnotationValueVisitor7
- AbstractBorder
- AbstractButton
- AbstractCellEditor
- AbstractCollection
- AbstractColorChooserPanel
- AbstractDocument
- AbstractDocument.AttributeContext
- AbstractDocument.Content
- AbstractDocument.ElementEdit
- AbstractElementVisitor6
- AbstractElementVisitor7
- AbstractExecutorService
- AbstractInterruptibleChannel
- AbstractLayoutCache
- AbstractLayoutCache.NodeDimensions
- AbstractList

# Wieloplatformowość

Kod źródłowy  
pliki \*.java

kompilacja

Kod bajtowy (B-kod)  
pliki \*.class

ładowanie

Wirtualna maszyna Java (**JVM**)

wykonywanie w środowisku Win/Unix/...

# **Java jako uniwersalny język programowania**

- **składniowe podobieństwo do C/C++**
- **automatyczne odśmiecanie** (*ang. Garbage collector*)
- **brak arytmetyki wskaźnikowej.**
- **zamiast wskaźników referencje**
- **ściśła kontrola typów**
- **obsługa wyjątków**
- **wbudowane elementy współbieżności**  
(tworzenie i synchronizacja wątków)
- **obiektość**
- **brak przeciążania operatorów**
- **przenośność “Write Once, Run Anywhere”** (złośliwi mówią “Write Once, Debug Everywhere”)



# Edycje Java

- **Java Standard Edition (Java SE)**

platforma przeznaczona dla zwykłych stacji roboczych, które w większości przypadków pracują pod kontrolą systemów: Linux, Solaris lub Windows. Jest to specyfikacja pozwalająca na tworzenie aplikacji klienckich.

- **Java Enterprise Edition (Java EE)**

platforma przeznaczona m.in. dla systemów rozproszonych, tj. świadczących usługi dla wielu użytkowników. J2EE dostarcza narzędzi programistycznych do programowania po stronie serwera (np. serwery aplikacji).

- **Java Micro Edition (Java ME)**

jest to zbiór technologii i specyfikacji wykorzystywanych przez małe urządzenia, takie jak: pagery, telefony komórkowe i inne “małe” urządzenia. J2ME wykorzystuje niektóre komponenty J2SE, takie jak mniejsza maszyna wirtualna i odchudzone API.

# Edycje Java

- **JavaFX** - rodzina technologii i produktów firmy Sun Microsystems, przeznaczonych głównie do tworzenia Rich Internet Application. W chwili obecnej w skład JavaFX wchodzi język skryptowy JavaFX Script oraz system dla urządzeń mobilnych Java ME. Język JavaFX Script ma w założeniu stać się konkurentem dla Adobe Flash i Flex, technologii AJAX oraz Microsoft Silverlight.
- **inne...**

## **Trzy najpopularniejsze rodzaje programów:**

- **aplikacja konsolowa – samodzielny program** pracujący w trybie konsolowym/tekstowym systemu operacyjnego,
- **aplikacja graficzna – samodzielny program** pracujący w trybie graficznym (okienkowym)
- **aplet – najczęściej nieduży program napisany w** język Java i umieszczony na stronie HTML i uruchamiany wraz z nią przez przeglądarkę internetową, obsługującą język Java.

# Java - uniwersalne środowisko programowania w sieci (klient-serwer)

Java zawiera standardowe środki do tworzenia:

- **apletów** - *programy wykonywalne w środowisku przeglądarki umożliwiającym:*
  - interakcję z użytkownikiem w rozbudowanym GUI
  - transakcje klient-serwer, w tym poprzez JDBC
- **serwletów** - *obsługa transakcji po stronie serwera, Java Servlet Api*

# Java - uniwersalne środowisko programowania

- **Łatwość tworzenia GUI (Graphical User Interface):**  
AWT (ang. *Abstract Windowing Toolkit*), Swing, SWT,
- **Duża liczba bibliotek** np. JDBC API (ang. *Java Database Connectivity*), JSP, JMX, JNDI, JMF, JNI, J3D i wiele innych
- **uniwersalne środowisko programowania multimedialnych**

# Java Development Kit (JDK)

## JDK – narzędzia podstawowe

- java - interpreter
- javac - kompilator
- apt – annotation processing tool
- javadoc
- appletviewer
- jar – zarządca Java Archives (jars)
- jdb – debugger
- javah – narzędzie do tworzenia metod natywnych

# Java Development Kit (JDK)

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

The screenshot shows the Oracle Java SE Downloads page in a web browser. The browser's address bar displays the URL [www.oracle.com/technetwork/java/javase/downloads/index.html](http://www.oracle.com/technetwork/java/javase/downloads/index.html). The page features a red header with the Oracle logo and navigation links. A sidebar on the left lists various Java products and services. The main content area is titled "Java SE Downloads" and includes tabs for Overview, Downloads, Documentation, Community, Technologies, and Training. Below the tabs, there are four categories: Latest Release, Next Release (Early Access), Embedded Use, and Previous Releases. Each category has a corresponding download button. The right sidebar lists Java SDKs and Tools, including Java SE, Java EE and Glassfish, Java ME, JavaFX, Java Card, and NetBeans IDE. At the bottom right, there is a small box with the text "UPC0055204 Dostęp do Internetu".

**ORACLE** Sign In/Register for Account Help Select Country/Region Communities I am a... I want to... Search


PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT Oracle Technology Network

Oracle Technology Network > Java > Java SE > Downloads


Overview Downloads Documentation Community Technologies Training

## Java SE Downloads


Latest Release Next Release (Early Access) Embedded Use Previous Releases

 **DOWNLOAD**

**Java Platform (JDK) 7u15**

 **DOWNLOAD**

**JavaFX 2.2.7**

 **DOWNLOAD**

**JDK 7 + NetBeans**

### Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [JavaFX](#)
- [Java Card](#)
- [NetBeans IDE](#)

### Java Resources

- [New to Java?](#)
- [APIs](#)
- [Code Samples & Apps](#)
- [Developer Training](#)
- [Documentation](#)
- [Java.com](#)
- [Java.net](#)

UPC0055204 Dostęp do Internetu

# Najprostsza aplikacja

```
public class Hello
{
    public static void main(String[ ] args)
    {
        System.out.println("Hello World!");
    }
}
```

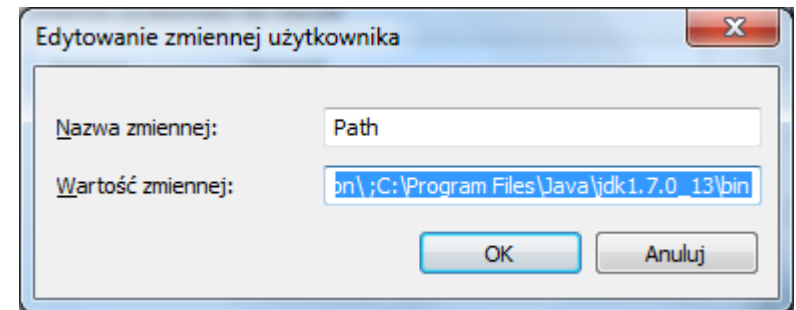
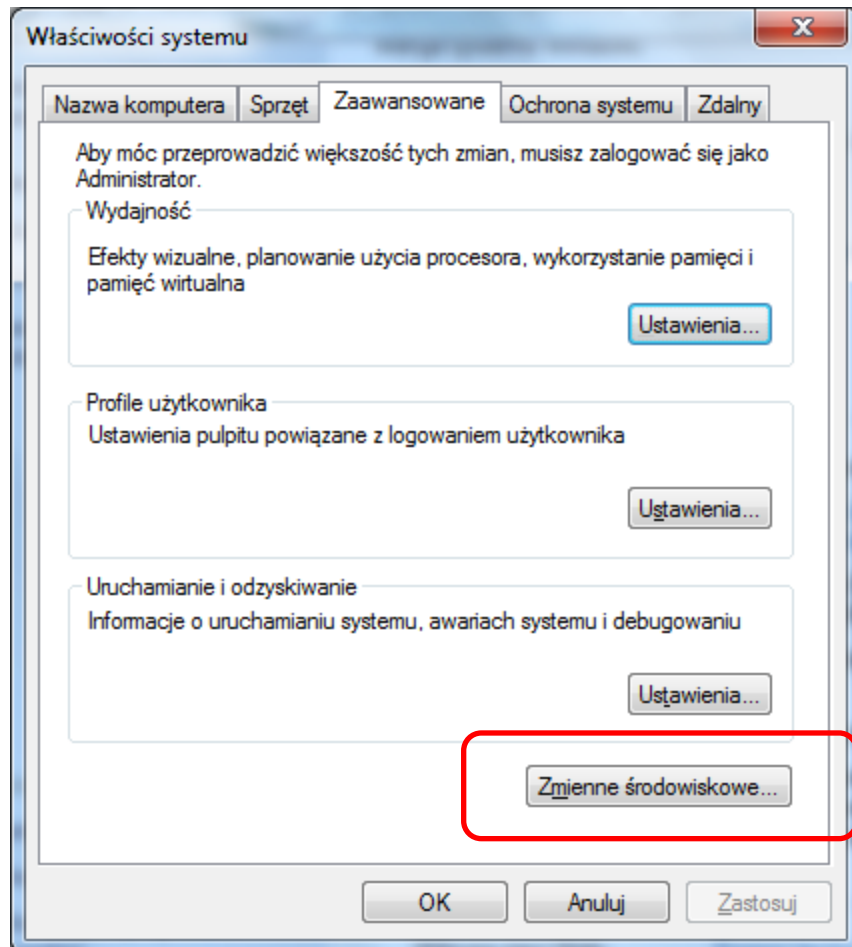
Klasa publiczna – plik powinien nazywać się tak samo jak klasa z rozszerzeniem .java (Hello.java)

Kompilacja z linii poleceń: javac Hello.java

Uruchamianie: java Hello



Uwaga: aby pod Windows możliwe było stosowanie komend JDK trzeba do zmiennej systemowej PATH dodać katalog \bin z JDK (np. C:\Program Files\Java\jdk1.7.0\_13\bin )



# IDE - Integrated development environment (zintegrowane środowisko programowania)

IDE - aplikacja (lub zespół aplikacji - środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania.

Aplikacje będące zintegrowanymi środowiskami programistycznymi charakteryzują się tym, że udostępniają złożoną, wieloraką funkcjonalność obejmującą edycję kodu źródłowego, kompilowanie kodu źródłowego, tworzenie zasobów programu (tzn. formatek / ekranów / okien dialogowych, menu, raportów, elementów graficznych takich jak ikony, obrazy itp.), tworzenie baz danych, komponentów i innych

# IDE dla Javy

- Eclipse (<http://www.eclipse.org>)
- Netbeans (<http://netbeans.org/>),
- IntelliJ IDEA  
(<http://www.jetbrains.com/idea/>),
- Jcreator (<http://www.jcreator.com/> )
- ...

**Na laboratoriach używany będzie Eclipse**

Eclipse – <http://www.eclipse.org/downloads>

The screenshot shows the Eclipse Downloads page in the Opera browser. The address bar displays [www.eclipse.org/downloads/](http://www.eclipse.org/downloads/). The page features a header with the EclipseCon Boston 2013 logo and navigation links. A sidebar on the left contains social media links and a search bar. The main content area lists various Eclipse packages for download, including Eclipse IDE for Java EE Developers, Eclipse Classic 4.2.1, Eclipse IDE for Java Developers, Spring Tool Suite, Eclipse IDE for C/C++ Developers, Eclipse for Mobile Developers, and Eclipse IDE for Java and DSL Developers. Each package entry includes a download icon, the package name, size, and download count. A right sidebar provides links for installing Eclipse and a promotional banner for the Eclipse Plug-in for Spring.

Opera Eclipse Downloads x +

Web [www.eclipse.org/downloads/](http://www.eclipse.org/downloads/) Szukaj w Google

Visit other Eclipse Sites

Home Downloads Users Members Committers Resources Projects About Us Google Custom Search Search

# Eclipse Downloads

Packages Developer Builds Projects

Eclipse Juno (4.2) SR1 Packages for Windows

	<b>Eclipse IDE for Java EE Developers</b> , 221 MB Downloaded 4,040,722 Times <a href="#">Details</a>	Windows 32 Bit Windows 64 Bit
	<b>Eclipse Classic 4.2.1</b> , 183 MB Downloaded 2,662,356 Times <a href="#">Details</a> <a href="#">Other Downloads</a>	Windows 32 Bit Windows 64 Bit
	<b>Eclipse IDE for Java Developers</b> , 150 MB Downloaded 1,951,372 Times <a href="#">Details</a>	Windows 32 Bit Windows 64 Bit
	<b>Spring Tool Suite</b> <span>Promoted Download</span> Complete IDE for enterprise Java, Spring, Groovy, Grails and the Cloud.	Download
	<b>Eclipse IDE for C/C++ Developers</b> , 129 MB Downloaded 941,673 Times <a href="#">Details</a>	Windows 32 Bit Windows 64 Bit
	<b>Eclipse for Mobile Developers</b> , 144 MB Downloaded 638,448 Times <a href="#">Details</a>	Windows 32 Bit Windows 64 Bit
	<b>Eclipse IDE for Java and DSL Developers</b> , 260 MB Downloaded 15,120,000 Times <a href="#">Details</a>	Windows 32 Bit Windows 64 Bit

## Installing Eclipse

- [Install Guide](#)
- [Compare/Combine Packages](#)
- [Known Issues](#)
- [Updating Eclipse](#)

FREE

Eclipse Plug-in for Spring

DOWNLOAD NOW

springsource by vmware

## Related Links

- [Documentation](#)

UPC0055204 Dostęp do Internetu

# Struktura programu:

```
package ... //deklaracja pakietu, opcjonalna ale zalecana
import ... // deklaracje importu
import ...
/** Komentarz dokumentacyjny */
// To jest klasa A
public class A
{
    ...
}
/* To jest
komentarz wielowierszowy
*/
class B
{
    ...
} //Koniec klasy B
```

# Struktura programu:

```
package pl.mojastrona.mojpakiet;

import javax.swing.*;
import java.awt.Container;

class MojaKlasa extends JFrame
{
    public MojaKlasa()
    {
        ...
    }
}

class KlasaStartowa
{
    MojaKlasa ob1;
    MojaKlasa ob2 = new MojaKlasa();

    public static void main(String[] args)
    {
        ...
    }
}
```

Określenie pakietu, do którego należą klasy zdefiniowane w tym pliku (*opcjonalne...*).

Zewnętrzne pakiety (lub pojedyncze klasy, nterfejsy), z których korzystamy w naszym programie.  
„odpowiednik” dyrektywy `#include` w C/C++.

Deklaracja klasy rozszerzającej inną klasę (dziedziczenie)

Konstruktor – taka sama nazwa jak klasa, może być kilka definicji konstruktorów dla jednej klasy, np.  
`public MojaKlasa( int parametrPoczątkowy)`  
{  
}

Druga klasa, w której deklarowane są referencje do obiektów innej klasy, oraz tworzony jest nowy obiekt operator „new” + wywołanie konstruktora

Metoda *main* klasy startowej – od niej rozpoczyna się uruchamianie programu

# Tworzenie obiektów

```
NazwaKlasy zmienna; //deklaracja zmiennej/referencji  
zmienna = new NazwaKlasy(argumenty_konstruktor);  
//tworzenie nowego obiektu przypisanego do referencji
```

Przykład: (tworzenie obiektu klasy Rectangle):

```
Rectangle prostokat;  
prostokat = new Rectangle(10,10,100,200);
```

Powyższy kod można uprościć, umieszczając deklarację zmiennej i tworzenie obiektu w jednej linii kodu:

```
NazwaKlasy zmienna = new NazwaKlasy(argumenty_konstruktor);
```

Przykład:

```
Rectangle prostokat = new Rectangle(10,10,100,200);
```

Jeśli obiekt jest potrzebny tylko po to, aby przekazać go do jakiejś metody, nie musimy tworzyć do niego referencji. W takich sytuacjach możliwe jest tworzenie nowego obiektu w trakcie wywoływania metody:

```
nazwaMetody( new NazwaKlasy(argumenty_konstruktor) );
```

# ZAPAMIĘTAJ!

Obiekty w Javie zawsze tworzone są przy pomocy operatora **new** po którym następuje konstruktor (czyli nazwa klasy, której obiekt jest tworzony oraz lista argumentów w nawiasach).

Operator **new** rezerwuje pamięć dla nowego obiektu oraz wywołuje odpowiedni konstruktor (klasa może mieć więcej niż jeden konstruktor, o tym który zostanie wywołany decyduje zgodność listy argumentów).

Operator **new** zwraca referencję do nowo utworzonego obiektu.



# Struktura klasy:

```
class NazwaKlasy {  
    //deklaracje pól  
    Typ pole1;  
    ...  
    Typ poleN;  
  
    //deklaracje metod i konstruktora/ów  
    Typ1 metoda1(lista-parametrów) {  
        //treść/zawartość metody1  
        return obiektTyp1;  
    }  
    ...  
    void metodaM(lista-parametrów) {  
        //treść/zawartość metodyM  
    }  
  
    NazwaKlasy(lista parametrów) {  
        //treść/zawartość konstruktora  
    }  
}
```

# Konwencja kodu

- Nazwa klasy powinna „coś znaczyć”. Pierwsza litera w nazwie klasy pisana jest wielka litera. Jeśli nazwa klasy składa się z kilku słów to pisze się je łącznie, każde słowo rozpoczynając z wielkiej litery:

```
class MojaKlasa
{
    ...
}
```

# Konwencja kodu

Metody i pola oraz nazwy referencji pisze się analogicznie, poza tym że pierwsza litera jest mała (należy pamiętać, że język Java rozróżnia małe i wielkie litery):

```
class MojaKlasa{  
    int mojaZmienna;  
    void pobierzZmienna(int numerZmiennej) {  
        ...  
    }  
}
```

# Konwencja kodu

Instrukcja złożona to ciąg instrukcji pomiędzy nawiasami { ... }, nazywana blokiem instrukcji np.:

```
instrukcja_grupujaca //klasa, metoda, itp.  
{  
... //blok kodu  
} //koniec instrukcja_grupujaca
```

Instrukcja grupująca (np. klasa, metoda, instrukcja sterująca, np. pętla) rozpoczyna blok kodu. Jeśli zadeklarujemy w bloku zmienna to będzie ona widoczna tylko w tym bloku. **Po klamrze zamykającej nie trzeba stawiać średnika.**

Istnieją kilka wariantów stawiania nawiasów {}:

```
while (i) {  
    }
```

```
while (i)  
{  
}
```

```
while (i)  
    {  
    }
```

Wszystkie te są poprawne. Warto jednak wybrać jeden i konsekwentnie się go trzymać. Warto również po otwarciu klamry od razu postawić klamrę zamykającą. Pozwoli to uniknąć wielu niepotrzebnych błędów.

# Konwencja kodu

Warto stosować nadmiarowości i dodatkowe separatory celem poprawienia czytelności kodu np.:

```
int a = (c * 2) + (b / 3);
```

zamiast:

```
int a=c*2+b/3;
```

# Konwencja kodu

```
instrukcja_grupujaca  
{  
    instrukcje bloku...  
} //koniec instrukcja_grupujaca
```

Kod po klamrze otwierającej umieszcza się w nowej linii. Każda linia bloku jest przesunięta (wcięta) względem pierwszego znaku `instrukcja_grupujaca` o stałą liczbę znaków. Blok kończy się klamrą zamykającą w nowej linii na wysokości pierwszego znaku instrukcji grupującej. Blok warto kończyć komentarzem umożliwiającym identyfikację bloku zamykanego przez daną klamrę.

# Konwencja kodu

Ważnym elementem czytelnej konstrukcji kodu jest stosowanie komentarzy liniowych:

```
int i = 5; //komentarz pojedynczej linii
```

lub blokowych:

```
/*  
blok komentarza:  
druga linia komentarza  
...  
*/
```

lub blokowych z dodatkowymi „gwiazdkami” \*:

```
/*  
* blok komentarza:  
* druga linia komentarza  
*/
```



# Typy proste JAVA

Nazwa typu	Liczba bajtów	Dopuszczalne wartości	Znaczenie	Przykłady literałów
byte	1	-128 / +127	l. całkowita	1, 01, 0x01
short	2	-32768 / +32767	l. całkowita	128, 0xFF
int	4	-2147483648 / +2147483647	l. całkowita	32768, 0x1000
long	8	-9223372036854775808 / +9223372036854775807	l. całkowita	3l(L), 21474836
float	4	-3.xE-38 / (+3.xE+38)-1	l. rzeczywiste	3f, 3F, 3e(E)+10
double	8	-1.xE-30 / (+1.xE+30) -1	l. rzeczywiste	0.3, 0.3d(D) ...
char	2	0...65556	znaki Unicode	'a', \u0013
boolean	1	true / false	wartości logiczne	true, false

# Operatory

Operator	Znaczenie
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
%	Reszta z dzielenia
\	Wynik dzielenia z resztą

Operator	Znaczenie
	Operacja $\vee$ - LUB
&&	Operacja $\wedge$ - I
!	Operacja $\sim$ - negacja
>	większy niż
<	mniej niż
>=	większy równy
<=	mniej równy
!=	różny od

Operator	Znaczenie
++	Inkrementacja - zwiększenie o 1
--	Dekrementacja - zmniejszenie o 1

# Instrukcje sterujące są analogiczne do C:

```
if(warunek_logiczny){  
    //instrukcje wykonane jeżeli warunek jest PRAWDZIWY  
}
```

```
if(warunek_logiczny){  
    //instrukcje wykonane jeżeli warunek jest PRAWDZIWY  
}  
else{  
    //instrukcje wykonane jeżeli warunek jest FAŁSZYWY  
}
```

Instrukcje „if” można zagłębiać lub dokonywać wielokrotnego wyboru:

```
if(warunek_logiczny1){  
    if(warunek_logiczny2){  
        //instrukcje wykonane jeżeli warunek2 jest PRAWDZIWY  
    }  
}
```

```
if(warunek_logiczny1){  
    //instrukcje wykonane jeżeli warunek1 jest PRAWDZIWY  
}  
else if(warunek_logiczny2){  
    //instrukcje wykonane jeżeli warunek2 jest PRAWDZIWY  
}  
else{  
    //instrukcje wykonane jeżeli warunek1 i warunek 2 są FAŁSZYWE  
}
```

# Zamiast bloku „if-else” można użyć operatora trójargumentowego ?

```
zmienna = warunek ? wartosc_jak_prawda : wartosc_jak_falsz;
```

## Co jest równoważne:

```
if(warunek){  
    zmienna = wartosc_jak_prawda;  
}  
else{  
    zmienna = wartosc_jak_falsz;  
}
```

# switch-case

```
switch ( key ) {  
    case value1:  
        // instrukcje dla key równego value1  
        break;  
    case value2:  
        // instrukcje dla key równego value2  
        break;  
    default:  
        break;  
}
```

# switch-case

```
int i = 0;
switch ( i ) {
    case 0:
        System.out.println(0);
    case 1:
        System.out.println(1);
        break;
    default:
        System.out.println("default");
        break;
}
```

Brak „break” przy pierwszym warunku –  
wynikiem będzie wypisanie „0” i „1”

# Pętla „for”

```
for(int i = 0; warunek; krok){  
    //instrukcja  
}
```

- Zmienną *i* nazywamy Indeks Pętli
- Indeks może być dowolnym typem prostym poza boolean.
- Warunek może być dowolnym zdaniem logicznym, należy jednak zwrócić uwagę by nie była to tautologia. Otrzymamy wtedy pętlę nieskończoną
- Krok pętli może być dowolny jednak tak samo jak w przypadku warunku trzeba uważać na zapętlenie się programu.

```
int[] a = new int[] { 1, 2, 3 };  
for (int i = 0; i < a.length; i++) {  
    System.out.println(a[i]);  
}
```



# do/while

Obie te konstrukcje są bardzo podobne do siebie. Główna różnica polega na tym iż w pętli `while` warunek jest sprawdzany przed wykonaniem instrukcji, a w pętli `do while` po wykonaniu instrukcji. Oznacza to, że pętla `do while` wykona się co najmniej jeden raz niezależnie od warunku..

```
while(warunekLogiczny){  
    //instrukcja  
}
```

```
do{  
    //instrukcja  
}while(warunekLogiczny)
```

# break/continue

Jeżeli chcemy przerwać wykonanie pętli jeżeli spełniony jest jakiś warunek to musimy użyć słowa `break`, a jeśli chcemy pominąć jakiś krok w pętli to musimy użyć słowa `continue`:

```
int i = 0;
while(true){
    if(i==5)
        break;
    System.out.println(i);
    i++;
}
```

```
for( int i = 0; i < 10; i++){
    if( i == 5 )
        continue;
    System.out.println(i);
}
```

# typ łańcuchowy (String)

- zaimplementowano jako obiekt
- automatyczne konwersje z innych typów i łączenie
- znaki łańcuchów są indeksowane od 0
- znak nieistniejący to -1

**operatory:** **+**, **=**, **+=** mają specjalne znaczenie dla obiektów klasy String

```
String s1;
```

```
String s2='Ala ';
```

```
s1=s2+"ma Asa" // s1 będzie zawierać napis  
„Ala ma Asa”
```

Z obiektami typu String można konkatelować liczby  
towarzyszy temu niejawne wywołanie metody **toString()**

```
int k=1;
```

```
s1 = k+k+s2; // s1 będzie zawierać 2Ala
```

```
s1=s2+k+k; // s1 będzie zawierać Ala11
```

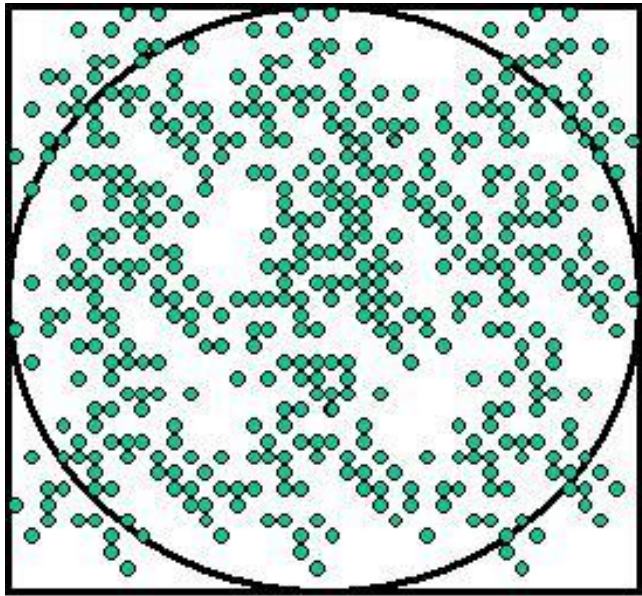
```
s1=s2+(k+k); // s1 będzie zawierać Ala2
```

# typ łańcuchowy (String)

//Przykład użycia obiektów String

```
public class StringPrzyklad
{
    public static void main(String args[])
    {
        String s1 = "Hello";
        String s2;
        s2 = s1;           //pełna kopia obiektu
        s1 = "World";
        System.out.println( "s1=" + s1 ); //s1=World
        System.out.println( "s2=" + s2 ); //s2=Hello
        s1 = 20+20+"";
        s2 = (20==30)+"";
        System.out.println( "s1=" + s1 ); //s1=40
        System.out.println( "s2=" + s2 ); //s2=false
        System.out.println( s1.charAt(1) ); //0
        System.out.println( s1.charAt(2) );
        // java.lang.StringIndexOutOfBoundsException:
        // String index out of range: 2
    }
}
```

# Przykład – wykorzystanie metody Monte-Carlo do wyznaczenia liczby $\pi$



Pole kwadratu:  $l^2$

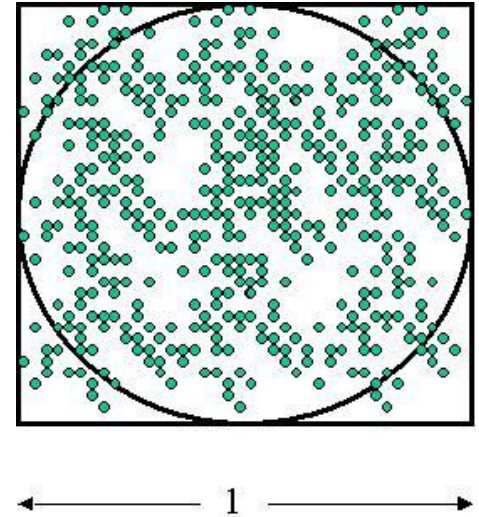
Pole koła:  $\pi \left(\frac{l}{2}\right)^2$

Losowanie  $N$  punktów, z czego  $M$  leży wewnątrz koła.

$$\frac{M}{N} = \frac{\pi}{4}$$

# Przykład – wykorzystanie metody Monte-Carlo do wyznaczenia liczby $\pi$

```
class Monte_Carlo {  
    public static void main(String[] args) {  
        double r1, r2;  
        int i, m=0, n=10000;  
        for (i=1 ; i<=n ; i++){  
            r1=Math.random()-0.5;  
            r2=Math.random()-0.5;  
            if (r1 *r1 + r2 *r2 <0.25) m++;  
        }  
        System.out.println("PI oszacowane = " + 4.* (double) m/n);  
        System.out.println("PI z Math.PI = " + Math.PI);  
    }  
}
```



# typ tablicowy

- Tablice w Javie są jednowymiarowe, ale mogą zawierać inne tablice.
- Przy definicji typu nie podaje się wielkości tablicy
- Każda tablica jest obiektem i ma zdefiniowane pole `length` o wartości równej liczbie elementów w tablicy
- Zadeklarowanie tablicy nie przydziela jej pamięci! Aby przydzielić pamięć należy użyć `new`
- Tablice są indeksowane wartościami typu `int`, stąd największa tablica ma 2147483647 elementów.

//przykłady deklaracji i tworzenia tablic

```
int arr[];
```

```
int [] arr;           //to samo co wyżej
```

```
arr = new int[3];
```

```
int arr[] = { 0, 0, 0 }; //równoważne
```

```
int arr[][];
```

```
arr[][] = new int[3][]; //powstaje tablica wektorów
```

```
arr[0] = new int[5];    //pierwszy wektor
```

```
arr[1] = new int[5];
```

```
arr[2] = new int[5];    //ostatni wektor
```

```
arr[][] = new int[10][];
```

```
for (int i=0; i<10; i++)
```

```
    arr[i] = new int[i+1]; //tablica trójkątna
```

```
arr[][] = { {1}, {0, 1}, {0, 0, 1} };
```

```
arr[0] = null;
```

```
arr[1] = new int[10];
```

```
arr[2] = {10, 10};
```



# Pakiety i JAR

- **Pakiet** – zbiór powiązanych klas i interfejsów, który pozwala na kontrole dostępu i zapewnia hierarchiczny system nazewnictwa. Przykładami są: `javax.swing`, `java.applet`, `java.io`. ...
- **JAR (Java Archive)** – archiwum Java, czyli plik zawierający skompresowane (ZIP) pliki klas i zasobów. Tworzy się je za pomocą narzędzia **jar** wchodzącego w skład JDK. Z klas znajdujących się w pliku `.jar` można korzystać, w tym uruchamiać aplikacje i aplety.

# Pakiety – pełna lista standardowych pakietów w dokumentacji

## Packages

Package	Description
<code>java.applet</code>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<code>java.awt</code>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<code>java.awt.color</code>	Provides classes for color spaces.
<code>java.awt.datatransfer</code>	Provides interfaces and classes for transferring data between and within applications.
<code>java.awt.dnd</code>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<code>java.awt.event</code>	Provides interfaces and classes for dealing with different types of events fired by AWT components.
<code>java.awt.font</code>	Provides classes and interface relating to fonts.
<code>java.awt.geom</code>	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
<code>java.awt.im</code>	Provides classes and interfaces for the input method framework.
<code>java.awt.im.spi</code>	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.

# Pakiety

- programy w Javie są złożone z zestawów pakietów, zawierających definicje klas i interfejsów
- typ zadeklarowany w pakiecie jest dostępny na zewnątrz pakietu tylko gdy został zadeklarowany jako publiczny (public)
- nazwa pakietu powinna być niepowtarzalna i jednocześnie określać jego charakter.
- zaleca się by określić kraj pochodzenia pakietu przez standardowe, dwuliterowe kody ISO, np..

`pl.edu.pw.fizyka.pojava.wyklad`

`com.sun.examples`

- pakiet składa się z kompilowalnych plików, które automatycznie mają dostęp do wszystkich typów deklarowanych w pakiecie
- jeśli plik nie zawiera deklaracji pakietu to domyślnie należy do pakietu unnamed/default