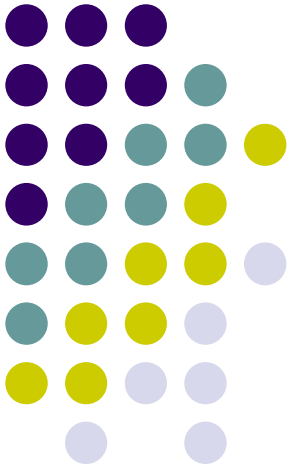
Javadoc

Piotr Dąbrowiecki
Sławomir Pawlewicz
Alan Pilawa
Joanna Sobczyk
Alina Strachocka



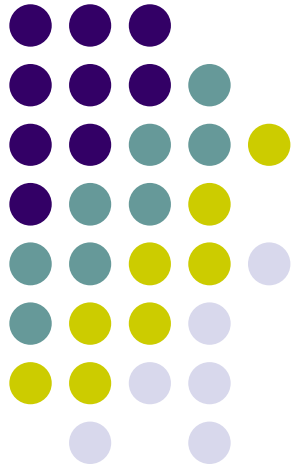
Wprowadzenie do Javadoc

Treść prezentacji:

- http://students.mimuw.edu.pl/~as219669/Javadoc.pdf

Zadania:

- http://students.mimuw.edu.pl/~as219669/zadanie.rar



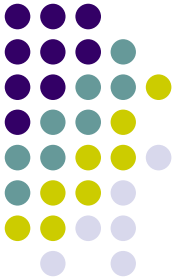
Wprowadzenie do Javadoc

- standardowe narzędzie do tworzenia dokumentacji API dla Javy
- powstał w firmie Sun Microsystems jako część pakietu do tworzenia aplikacji Java 2 SDK
- służy tylko do dokumentowania programów napisanych w Javie
- generuje dokumentację z kodu źródłowego do formatu html

Co wygeneruje Javadoc?

- listę wszystkich pakietów w projekcie wraz z listą klas w każdym pakiecie
- hierarchię klas oraz zależności między klasami
- opis klasy, pakietu, projektu
- informacje o przestarzałych metodach (*deprecated*)
- alfabetyczny indeks klas, interfejsów, konstruktorów, pól i metod

Przykład wygenerowanej dokumentacji 1



Java™ Platform Standard Ed. 6

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event

All Classes

AbstractAction
AbstractAnnotationValueVisi
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeC
AbstractDocument.Content
AbstractDocument.ElementE
AbstractElementVisitor6
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeD
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutab
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronize
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchr
AbstractQueuedSynchronize
AbstractScriptEngine
AbstractSelectableChannel
AbstractSelectionKey
AbstractSelector

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Java™ Platform Standard Ed. 6

Java™ Platform, Standard Edition 6 API Specification

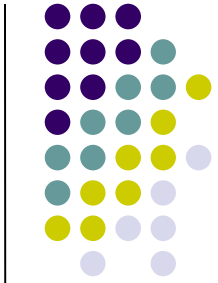
This document is the API specification for version 6 of the Java™ Platform, Standard Edition.

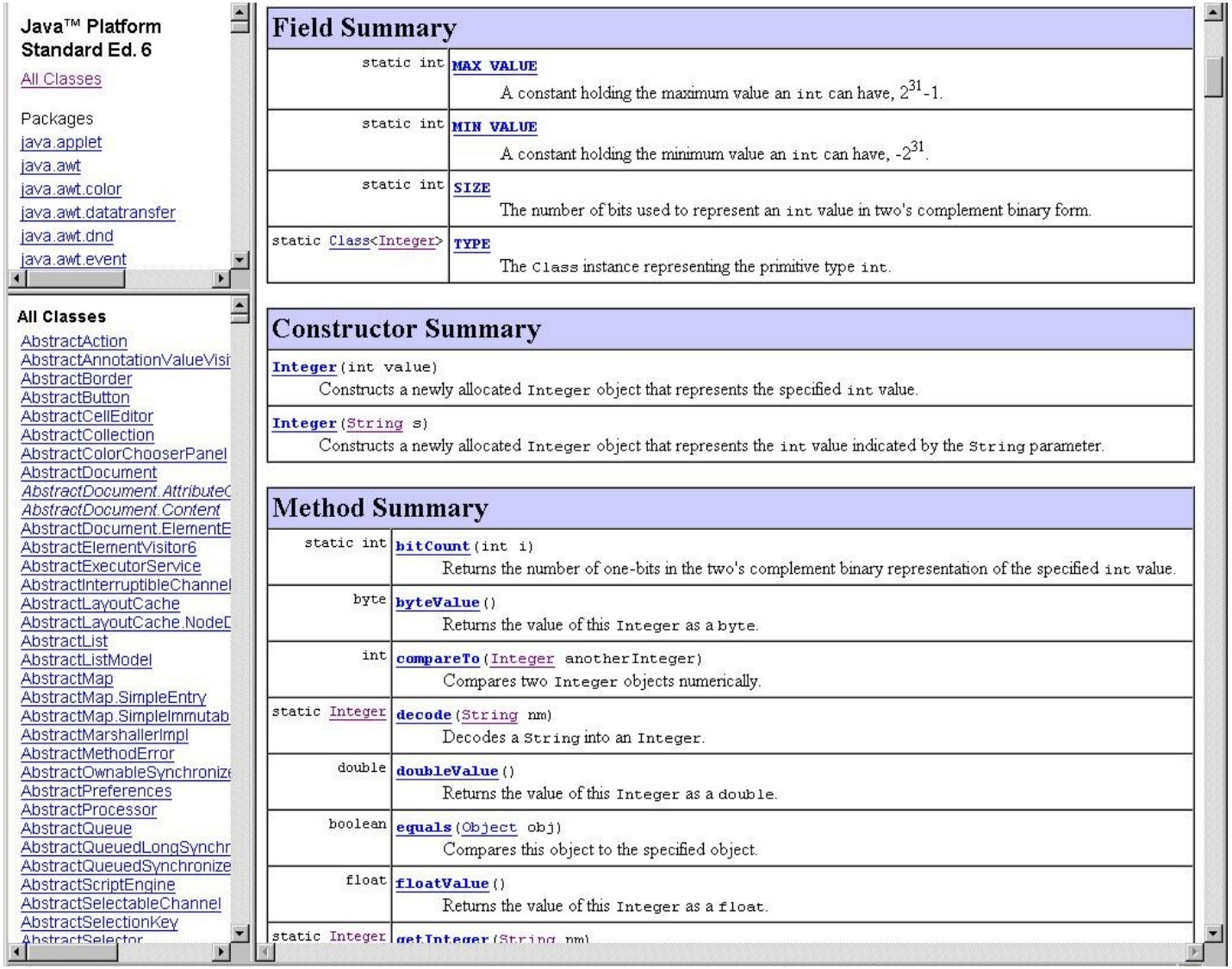
See:

Description

Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	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.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent

Przykład wygenerowanej dokumentacji 2





Java™ Platform Standard Ed. 6

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event

All Classes

AbstractAction
AbstractAnnotationValueVisi
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeC
AbstractDocument.Content
AbstractDocument.ElementE
AbstractElementVisitor6
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeD
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutab
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronize
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchr
AbstractQueuedSynchronize
AbstractScriptEngine
AbstractSelectableChannel
AbstractSelectionKey
AbstractSelector

Field Summary

static int	**MAX_VALUE** A constant holding the maximum value an int can have, $2^{31}-1$.
static int	**MIN_VALUE** A constant holding the minimum value an int can have, -2^{31}.
static int	**SIZE** The number of bits used to represent an int value in two's complement binary form.
static Class<Integer>	**TYPE** The Class instance representing the primitive type int.

Constructor Summary

Integer(int value)
Constructs a newly allocated Integer object that represents the specified int value.

Integer(String s)
Constructs a newly allocated Integer object that represents the int value indicated by the String parameter.

Method Summary

static int	**bitCount**(int i) Returns the number of one-bits in the two's complement binary representation of the specified int value.
byte	**byteValue**() Returns the value of this Integer as a byte.
int	**compareTo**(Integer anotherInteger) Compares two Integer objects numerically.
static Integer	**decode**(String nm) Decodes a String into an Integer.
double	**doubleValue**() Returns the value of this Integer as a double.
boolean	**equals**(Object obj) Compares this object to the specified object.
float	**floatValue**() Returns the value of this Integer as a float.
static Integer	**getInteger**(String nm)

Zasady działania Javadoc

- bazuje na kompilatorze Javy
- używa specjalnych znaczników umieszczonych w komentarzach do generowania dokumentacji
- używa tzw. *docletów* do generowania plików wyjściowych (do określenia formatu wyjściowego)

Zalety Javadoc

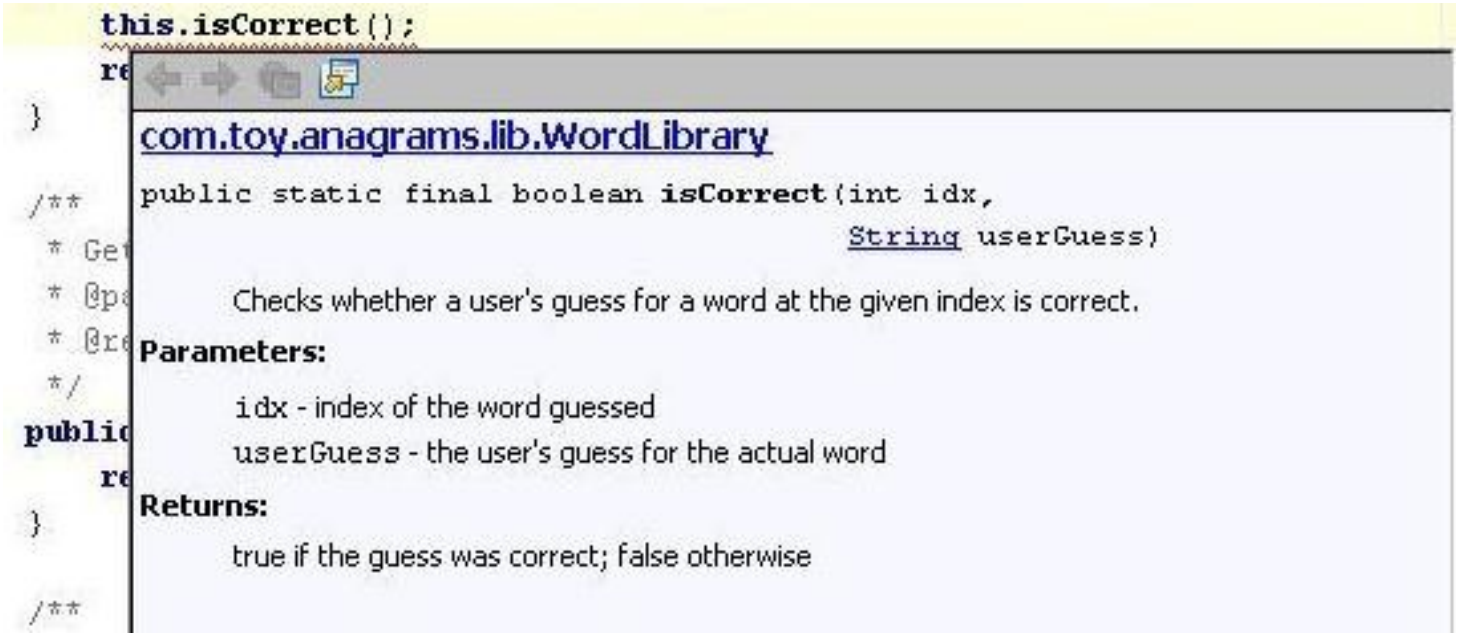
- dokumentacja zintegrowana z kodem źródłowym

```
/*
 * Main.java
 *
 * Created on 13 luty 2007, 14:28
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package helloworldapp;
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 *
 * @author Ala
 */
public class HelloWorldApp {
    /** Creates a new instance of HelloWorldApp */
    public HelloWorldApp() {
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello world!");//Display the string
    }
}
```

Zalety Javadoc

- NetBeans i Eclipse wykorzystują Javadoc

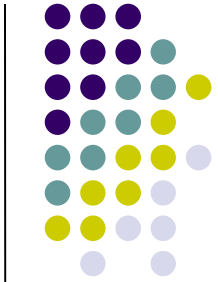
```
/**
 * Checks whether a user's guess for a word at the given index is correct.
 * @param idx index of the word guessed
 * @param userGuess the user's guess for the actual word
 * @return true if the guess was correct; false otherwise
 */
public static boolean isCorrect(int idx, String userGuess) {
    return userGuess.equals(getWord(idx));
}
```



Doclety

- programy napisane w Javie z własnym API
- określają zawartość i format pliku wyjściowego, wygenerowanego przez Javadoc
  - np. można napisać doclet do generowania dowolnego pliku tekstowego
  - firma Sun dostarcza standardowy doclet do generowania dokumentacji w formacie html oraz eksperymentalny – w formatach: MIF, PDF, PS, RTF, FrameMaker i innych.
- aby użyć niestandardowego docletu:
  - *javadoc* ***-doclet JP.co.esm.caddies.doclets.UMLDoclet*** *-private *.java*
  - (UMLDoclet, generuje strony HTML zawierające diagramy klas w standardzie UML)

Doclety



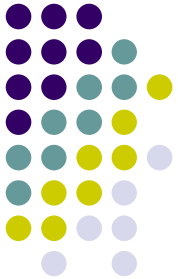
- szkielet docletu:

```
import com.sun.javadoc.*;

public class A {
    ...
    public static boolean start(RootDoc root, ...) {
        ...
    }
}
...
```

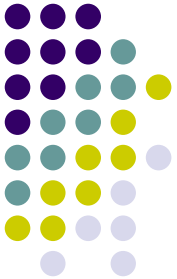
- klasa RootDoc przechowuje informacje z jednego uruchomienia Javadoc (pakiety, klasy i opcje określone przez użytkownika)
- pliki z klasami API docletu są w pliku *lib/tools.jar*. Po kompilacji *tools.jar* musi być w ścieżce klasowej.
  - można do tego celu użyć opcji *-classpath* z Javadoc.

Wady Javadoc



- Javadoc nie wspiera innych języków niż Java
- nie umożliwia dokumentowania treści metod
- generuje dokumentację tylko w formacie html

Jak korzystać z Javadoc?

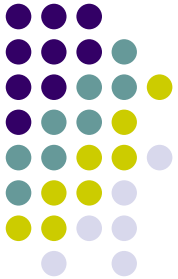


```java
/**
 * Komentarz klasy
 * ...
 * @author Autor
 * @version beta
 */
public class Klasa {
    /** komentarz do zmiennej */
    int zmienna = 42;

    /** Komentarz konstruktora */
    public Klasa() {}

    /**
     * Komentarz metody
     * @deprecated Uzyj metody metoda2
     * @see InnaKlasa#innaMetoda()
     */
    public int metoda() {
        return 42;
    }
    /** metoda  */
    public int metoda2() {
        return zmienna;
    }
}
```

Jak korzystać z Javadoc?

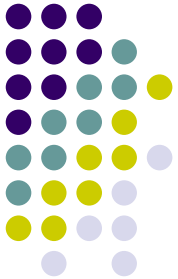


```java
/**
 * Komentarz klasy
 * ...
 * @author Autor
 * @version beta
 */
public class Klasa {
    /** komentarz do zmiennej */
    int zmienna = 42;

    /** Komentarz konstruktora */
    public Klasa() {}

    /**
     * Komentarz metody
     * @deprecated Uzyj metody metoda2
     * @see InnaKlasa#innaMetoda()
     */
    public int metoda() {
        return 42;
    }
    /** metoda  */
    public int metoda2() {
        return zmienna;
    }
}
```

Jak korzystać z Javadoc? Dokumentowanie klasy

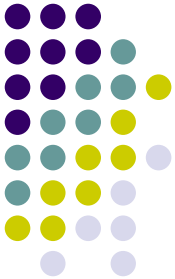


```java
/**
 * Komentarz klasy
 * ...
 * @author Autor
 * @version beta
 */
public class Klasa {
    /** komentarz do zmiennej */
    int zmienna = 42;

    /** Komentarz konstruktora */
    public Klasa() {}

    /**
     * Komentarz metody
     * @deprecated Uzyj metody metoda2
     * @see InnaKlasa#innaMetoda()
     */
    public int metoda() {
        return 42;
    }
    /** metoda  */
    public int metoda2() {
        return zmienna;
    }
}
```

Jak korzystać z Javadoc? Dokumentowanie pola

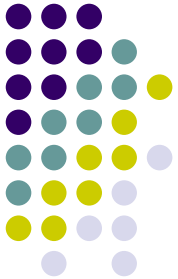


```java
/**
 * Komentarz klasy
 * ...
 * @author Autor
 * @version beta
 */
public class Klasa {
    /** komentarz do zmiennej */
    int zmienna = 42;

    /** Komentarz konstruktora */
    public Klasa() {}

    /**
     * Komentarz metody
     * @deprecated Uzyj metody metoda2
     * @see InnaKlasa#innaMetoda()
     */
    public int metoda() {
        return 42;
    }
    /** metoda  */
    public int metoda2() {
        return zmienna;
    }
}
```

Jak korzystać z Javadoc? Dokumentowanie metody

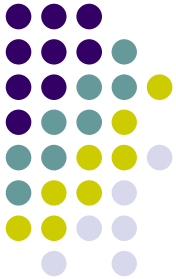


```java
/**
 * Komentarz klasy
 * ...
 * @author Autor
 * @version beta
 */
public class Klasa {
    /** komentarz do zmiennej */
    int zmienna = 42;

    /** Komentarz konstruktora */
    public Klasa() {}

    /**
     * Komentarz metody
     * @deprecated Uzyj metody metoda2
     * @see InnaKlasa#innaMetoda()
     */
    public int metoda() {
        return 42;
    }
    /** metoda  */
    public int metoda2() {
        return zmienna;
    }
}
```

Znaczniki Javadoc



- **@param nazwaParametru opis**
  znacznik opisujący parametr klasy lub metody. W opisie powinien znaleźć się typ parametru.
- **@return opis**
  znacznik opisujący wartość zwracaną przez metodę. Opis powinien zawierać typ zwracanego wyniku oraz zakres zwracanych wartości.
- **@throws** (lub **@exception**) **wyjątek opis**
  znacznik opsujący wyjątek, jaki może rzucić metoda.

Znaczniki Javadoc

- **@author nazwa**
  Informuje kto jest autorem kodu. Powinno być stosowane jedynie w opisie klas i interfejsów.
- **@version textWersji**
  znacznik informujący, do której wersji oprogramowania należy ta części kodu.
- **@since textWersji**
  znacznik informujący, w której wersji oprogramowania została dodana opisywana część kodu.

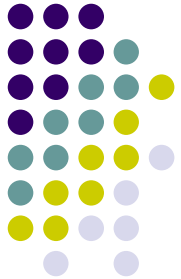
Znaczniki Javadoc

- **@see odnosnik**
  Odnośnik do innych materiałów.
  Dopuszczalne są trzy formy:
  @see “napis”
  nie tworzy linku. Wyświetla tylko napis
  @see etykieta
  tworzy link do strony podanej jako URL
  @see pakiet.klasa#element etykieta
  najbardziej popularna forma. Tworzy odnośnik wewnątrz projektu do elementu klasy (metoda, pole).

Znaczniki Javadoc

- **@deprecated opis**
  znacznik informujący, że dana część kodu jest przestarzała i nie powinna być wykorzystywana. W opisie powinna wystąpić informacja od której wersji kod jest przestarzały oraz czym należy go zastąpić.
- **{@link pakiet.klasa#element etykieta}**
  znacznik, który może wystąpić wewnątrz innego. Tworzy on odnośnik do elementu klasy i jest opatrzony etykietą.

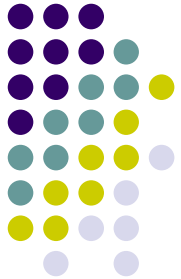
Znaczniki Javadoc
Gdzie jaki można stosować?



Znaczniki dokumentacji pakietu:

- @see
- @since
- @serial
- @author
- @version
- {@link}
- {@linkplain}
- {@docRoot}

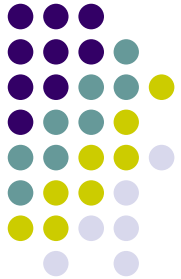
Znaczniki Javadoc
Gdzie jaki można stosować?



Znaczniki dokumentacji klasy i interfejsu:

- @see
- @since
- @deprecated
- @serial
- @author
- @version
- {@link}
- {@linkplain}
- {@docRoot}

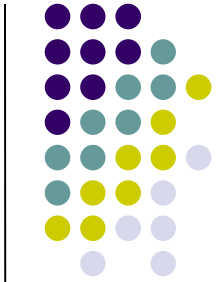
Znaczniki Javadoc
Gdzie jaki można stosować?



Znaczniki dokumentacji pola:

- @see
- @since
- @deprecated
- @serial
- @serialField
- {@link}
- {@linkplain}
- {@docRoot}
- {@value}

Znaczniki Javadoc
Gdzie jaki można stosować?



Znaczniki dokumentacji metod i konstruktorów

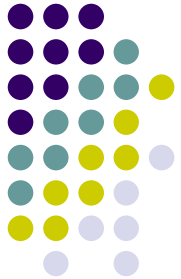
- @see
- @since
- @deprecated
- @param
- @return
- @throws i @exception
- @serialData
- {@link}
- {@linkplain}
- {@inheritDoc}
- {@docRoot}

Javadoc w Eclipse

- wpisanie w edytorze ciągu znaków: „/**” powoduje wygenerowanie szkieletu komentarza
- to samo można osiągnąć:
  - umieszczając kursor w nagłówku metody, klasy lub zmiennej, którą chcemy skomentować i z menu prawego przycisku myszki wybrać

    Source -> Add comment
  - Naciskając kombinację klawiszy Shift + Alt + J
- standardowo dodawane są parametry metody, autor klasy

Javadoc w Eclipse

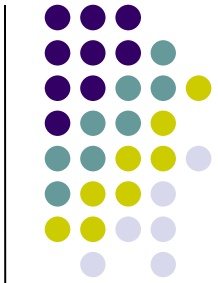


- szkielet komentarza można modyfikować:

  Window->Preferences->Java->Code Style->Code Templates.
- Eclipse umożliwia w łatwy sposób generowanie dokumentacji:

  menu Project -> Generate Javadoc

Uruchamianie Javadoc



- wyświetlenie instrukcji obsługi javadoc:
  - *javadoc* lub
    javadoc -help
- generowanie dokumentacji:
  - *javadoc [opcje] nazwa_pakietu* lub
    javadoc [opcje] nazwa_pliku.java

Uruchamianie Javadoc

- niektóre opcje:
  - **-public** – dokumentowane są tylko klasy public i ich elementy składowe
  - **-protected** – public + protected (tak jak domyślnie)
  - **-package** – protected + klasy i składowe o zasięgu widoczności pakietowym
  - **-private** – dokumentowane są wszystkie klasy i ich elementy składowe
  - **-exclude <lista pakietów>** - lista pakietów które zostaną wykluczone przy generowaniu dokumentacji
  - **-subpackages <lista pakietów>** - lista pakietów które zostaną przetworzone rekurencyjnie (wraz z podpakietami)

Uruchamianie Javadoc

- niektóre opcje cd.:
  - **-doclet nazwa_pliku** – określa plik klasowy z docletem
  - **-docletpath <ścieżka>** - katalog, w którym znajduje się plik .class z docletem
  - **-sourcepath <ścieżka>** – katalog zawierający dokumentowane pliki źródłowe (gdy używamy nazw pakietów)
  - **-classpath <ścieżka>** – określa katalogi, w których znajdują się pliki .class dla klas, do których są odwołania w dokumentacji.

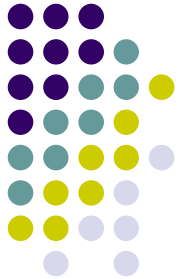
Uruchamianie Javadoc

- opcje standardowego docletu:
  - **-d katalog** – określa katalog docelowy plików HTML
  - **-version** – uwzględnia znacznik @version
  - **-author** – uwzględnia znacznik @author
  - **-splitindex** – dzieli plik z indeksem na 26 plików, po jednym dla każdej litery
  - **-doencoding <nazwa>** – określa nazwę kodowania wynikowego pliku HTML

Uruchamianie Javadoc

- opcje standardowego docletu cd.:
  - **-use** – generuje stronę z informacją gdzie i przez kogo dana klasa lub pakiet są używane
  - **-nodeprecated** – ukrywa informację pochodzącą z akapitów ze znacznikiem @deprecated
  - **-noindex** – nie generuje indeksu
  - **-notree** – nie generuje hierarchii klas/interfejsów
  - **-nohelp** – nie generuje strony pomocy

Ćwiczenia



- Zad1. Rozpakować załączony projekt i umieścić go w workspace Eclipsa. Zaimportować Projekt. Uruchomić Javadoca (z Eclipse lub konsoli), porównać wygenerowaną stronę z komentarzami w kodzie.
  http://students.mimuw.edu.pl/~as219669/zadanie.zip
  http://students.mimuw.edu.pl/~as219669/zadanie.rar

- Zad2. Wykonać polecenia z komentarzy w kodzie.

Przydatne linki

- http://java.sun.com/j2se/javadoc/
- http://java.sun.com/j2se/javadoc/writingdoccomments/
- dla zainteresowanych:

  http://java.sun.com/j2se/1.5.0/docs/guide/javadc