



DOKUMENTOWANIE KODU ŹRÓDŁOWEGO - JAVADOC

GRUPA POMARAŃCZOWA

CO TO DOKUMENTACJA KODU

Dokumentacja kodu to zbiór opisów, instrukcji i komentarzy, które wyjaśniają działanie kodu źródłowego. Służy ona do ułatwienia zrozumienia kodu przez innych programistów, wspomaga utrzymanie oraz rozwijanie oprogramowania, a także umożliwia efektywniejsze wdrożenie nowych członków zespołu. Jest niezbędna dla zapewnienia ciągłości projektu, jako że umożliwia szybsze identyfikowanie i rozwiązywanie problemów oraz efektywne przekazywanie wiedzy w obrębie zespołu. W praktyce dobrze udokumentowany kod jest bardziej wartościowy, ponieważ jego dalsze użytkowanie i modyfikacja są znacznie łatwiejsze.

METODY DOKUMENTOWANIA KODU

A dark blue rounded rectangle containing the text `// Komentarz` in a light blue monospace font.

// Komentarz

Komentarze w kodzie: Najprostsza forma dokumentacji, polegająca na umieszczaniu krótkich i ogólnych opisów bezpośrednio w kodzie źródłowym, obok bloków kodu, funkcji czy klas. Pozwala to na szybkie zrozumienie, co dana część kodu robi.

A dark blue rounded rectangle with the text 'DOKUMENTACJA' in red and 'PROJEKT IT' in white below it.

DOKUMENTACJA
PROJEKT IT

Dokumentacja techniczna: Formalny sposób dokumentowania kodu, może obejmować obszerne podręczniki dla programistów z opisami funkcji, parametrów czy zwracanych wartości. Często zawiera diagramy UML.

A dark blue rounded rectangle with a blue GitHub Octocat logo on the left and the text 'README.MD' in white on the right.

README.MD

Dokumentacja README: W repozytoriach kodu (np.: na GitHub) służą jako wstępne wprowadzenie do projektu, wyjaśniają jak rozpocząć pracę, jakie są zależności oraz jak skompilować i uruchomić projekt.

A white rounded rectangle containing the 'doxygen' logo in blue, with a blue underline under the 'y' and 'g'.

Generatory dokumentacji: Narzędzia takie jak Doxygen, Javadoc, Sphinx automatyzują proces tworzenia dokumentacji, generując ją na podstawie komentarzy w kodzie zgodnie z ustalonymi standardami.

DOKUMENTOWANIE KODU W JAVIE



Podręcznik programisty

Zawiera opisy architektury systemu, bibliotek i API, uzupełnione o diagramy UML i schematy, co pomaga w orientacji w projekcie i jego efektywnym wykorzystaniu.

Komentarze w kodzie

Służą do wyjaśniania działania kodu, pomagając w zrozumieniu metod i algorytmów, co jest kluczowe dla utrzymania i rozwijania oprogramowania.

Javadoc

Narzędzie do generowania dokumentacji HTML z komentarzy w kodzie Java, standardowe w dokumentowaniu API, ułatwia dostęp do uporządkowanej dokumentacji dla programistów.

CZYM JEST JAVADOC?

Standardowy system dokumentacji kodu źródłowego w języku Java

Narzędzie dostarczane wraz z JDK (Java Development Kit)

Generuje dokumentację API w formacie HTML

Bazuje na specjalnych komentarzach w kodzie źródłowym

Dokumentacja generowana jest poza kodem aplikacji



HISTORIA I ZNACZENIE



Wprowadzony w 1995 roku wraz z JDK 1.0 przez Sun Microsystems (dziś Oracle).



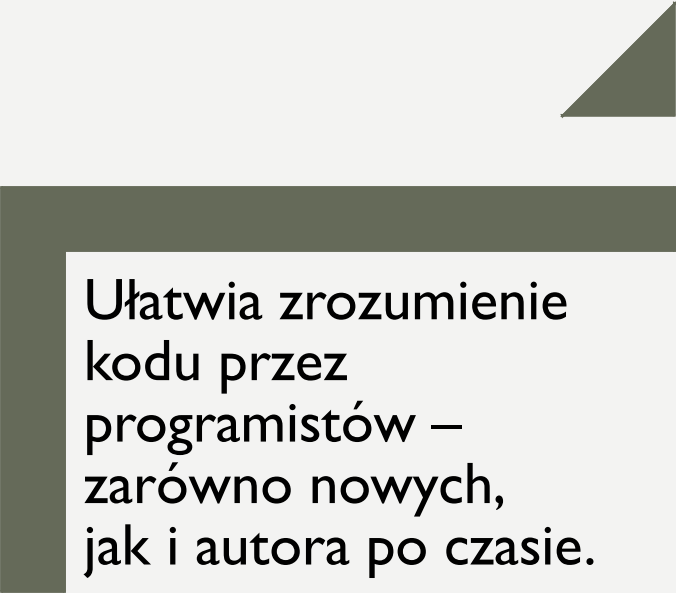
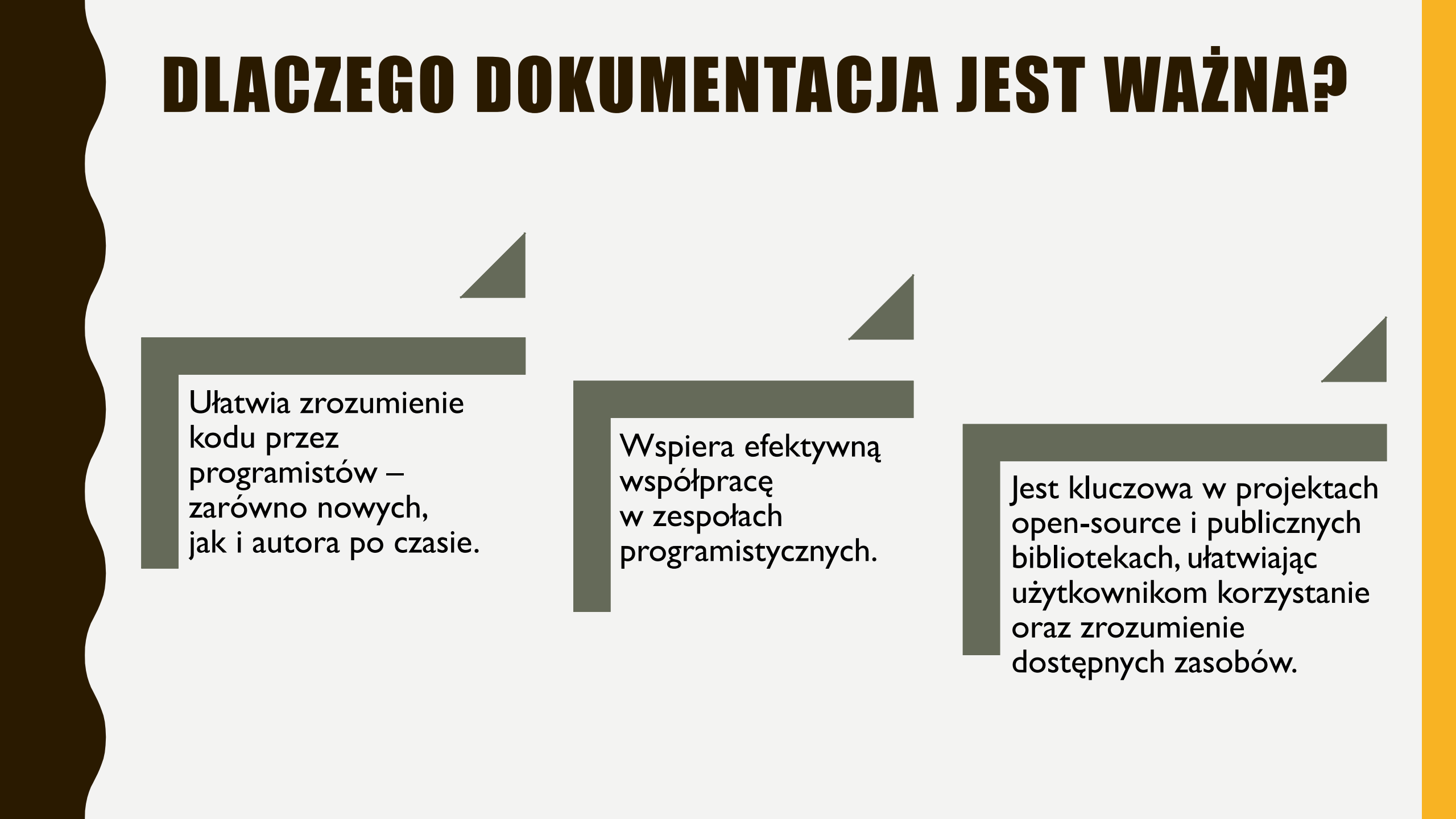
Od początku stał się standardem dokumentacji w ekosystemie Javy.



Po przejęciu Sun przez Oracle w 2010 roku, rozwój Javadoc kontynuowany jest pod nadzorem Oracle, jako integralna część JDK.



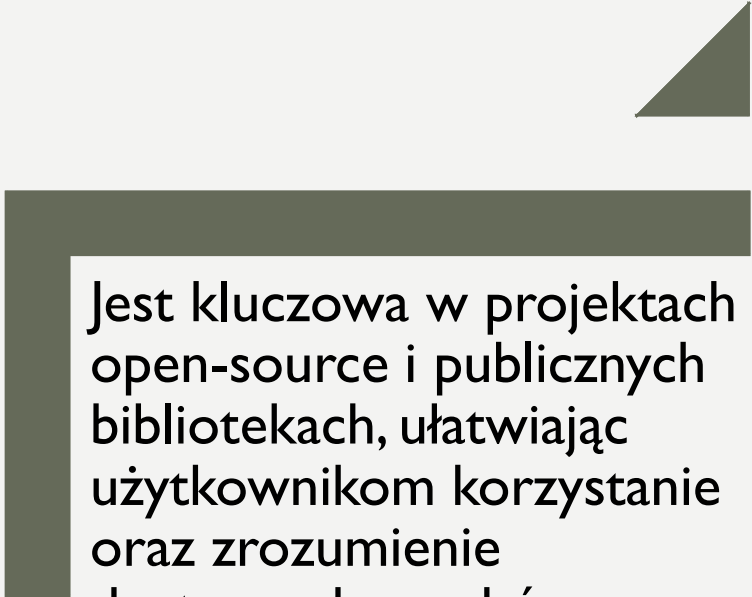
DLACZEGO DOKUMENTACJA JEST WAŻNA?



Ułatwia zrozumienie kodu przez programistów – zarówno nowych, jak i autora po czasie.




Wspiera efektywną współpracę w zespołach programistycznych.



Jest kluczowa w projektach open-source i publicznych bibliotekach, ułatwiając użytkownikom korzystanie oraz zrozumienie dostępnych zasobów.

STRUKTURA KOMENTARZY JAVADOC

Komentarze rozpoczynają się od `/**` i kończą się `*/`.
Komentarze `//` i `/* */` nie będą brane pod uwagę.



Każda linia komentarza Javadoc powinna zaczynać się od gwiazdki `*`.



Pierwsza linia to opis metody lub klasy, która zadeklarowana jest poniżej.

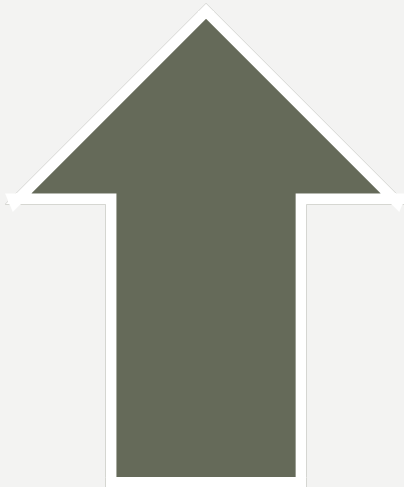


Kolejne akapity mogą zawierać szczegółowe informacje.



Specjalne tagi (rozpoczynające się od `@`) służą do opisu specyficznych elementów (parametry, wartość zwracana).

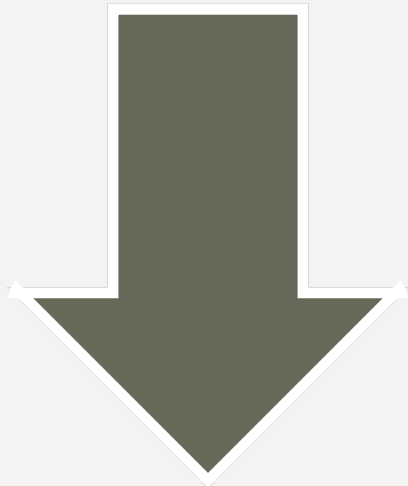
ZALETY I WADY



Automatyzacja: Javadoc automatycznie generuje dokumentację w formie HTML z komentarzy w kodzie, co znacząco oszczędza czas.

Standaryzacja: Umożliwia tworzenie jednolitych i spójnych dokumentacji dla całego projektu, co poprawia czytelność i łatwość użytkowania.

Integracja: Jest dobrze zintegrowany z większością środowisk programistycznych (IDE), co ułatwia tworzenie, zarządzanie i przeglądanie dokumentacji.



Zależność od formatowania: Wymaga przestrzegania określonego formatu komentarzy, co może być uciążliwe i ograniczać swobodę opisu.

Płytkie informacje: Skupia się głównie na dokumentowaniu interfejsów API, pomijając szerszy kontekst lub złożoność implementacji.

Konserwacja: Aktualizacja dokumentacji wymaga ręcznego aktualizowania komentarzy w kodzie, co w dużych projektach może być czasochłonne.

PODSTAWOWE TAGI JAVADOC

`@param <nazwa_parametru> <opis>` - Opisuje parametr metody.

`@return <opis>` - Opisuje wartość zwracaną przez metodę.

`@throws <nazwa_wyjątku> <opis>` - Opisuje wyjątek, który metoda może rzucić, i warunki, w jakich to się dzieje.

`@author <imię_nazwisko>` - Oznaczenie autora klasy/metody.

`@version <tekst_wersji>` - Opisuje wersje klasy/metody.

`@since <wersja>` - Wskazuje od jakiej wersji dostępna jest klasa/metoda.

PODSTAWOWE TAGI JAVADOC

`@serial <opis>` - Opisuje pola serializowane.

`@see <odnośnik>` - Odniesienie do innej dokumentacji.

`@exception <nazwa_wyjątku> <opis>` - Opisuje wyjątek, który metoda może rzucić, i warunki, w jakich to się dzieje. Starsza, mniej popularna alternatywa dla `@throws`.

`@serialData <opis_danych>` - Opisuje dane zapisywane przez metody `writeObject()` lub `writeExternal()`.

`@serialField <nazwa_pola> <typ_pola> <opis>` - Opisuje komponent typu `ObjectStreamField`.

`@deprecated <opis>` - Oznaczenie, że klasa/metoda jest przestarzała.

PODSTAWOWE TAGI JAVADOC

`{@code <tekst>}` - Wyświetla tekst w kodzie bez interpretacji jako HTML.

`{@docRoot}` - Reprezentuje ścieżkę do głównego katalogu dokumentacji.

`{@link pakiet.klasa#element etykieta}` - Wstawia odnośnik do dokumentacji innej klasy/metody.

`{@linkplain pakiet.klasa#element etykieta}` - Działa jak `{@link}`, ale wyświetlany jest jako zwykły tekst.

`{@value pakiet.klasa#pole}` - Wyświetla wartość stałej statycznej.

`{@inheritDoc}` - Dziedziczy dokumentację z najbliższej nadklasy lub interfejsu.

NAJLEPSZE PRAKTYKI

Dokumentuj każdą publiczną klasę, metodę i pole: • Opisz ich cel i sposób działania.

Używaj znaczników, np.: `@param`, `@return`, `@throws`: • Dla zachowania czytelności.

Pisz zwięźle i jasno, unikaj zbędnych informacji: • Pisz dla użytkownika.

Unikaj powielania kodu w opisie: • Zamiast tego, wyjaśnij jego logikę i zastosowanie.

Aktualizuj na bieżąco dokumentację: • Dopasowuj Javadoc do zmian w kodzie.

Dodawaj przykłady użycia, jeśli to możliwe: • Pokaż, jak korzystać z metody lub klasy.

CO SPRAWIA, ŻE KOMENTARZ JEST DOBRY?

Jasny opis klasy i metody:

- Klasa: Powinna zawierać krótkie wyjaśnienie, czym jest i do czego służy.
- Metoda: Powinna precyzyjnie opisywać, co robi oraz w jakim kontekście jest używana.

Szczegółowe parametry (@param):

- Każdy parametr metody powinien być dokładnie opisany, aby wyjaśnić jego przeznaczenie.
- Dla klas warto wskazać, jakie dane są przechowywane i do czego służą.

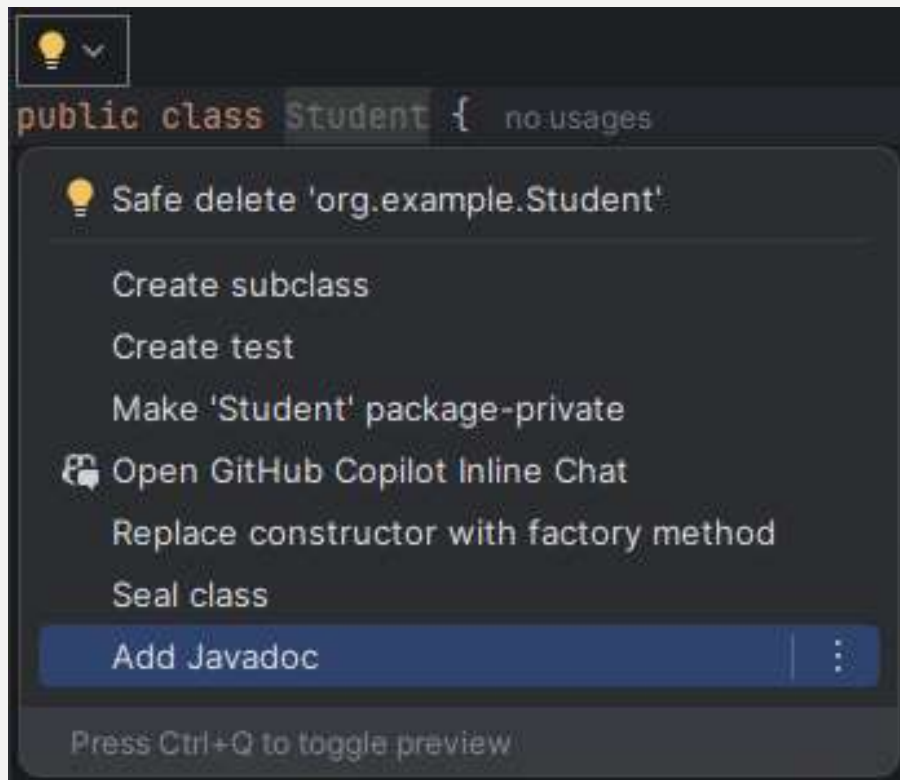
Dokładny opis zwracanej wartości (@return):

- Powinien jednoznacznie określać, co metoda zwraca i w jakiej sytuacji.

Uwzględnienie wyjątków (@throws):

- Metoda powinna informować o potencjalnych błędach, które może zgłosić.
- Dla klas można wskazać, jakie wyjątki mogą wystąpić podczas ich używania.

PRZYKŁAD DOKUMENTACJI KLASY



```
/**  
 *  
 */  
public class Student { no usages  
    private String imie; no usages  
    private String nazwisko; no usages  
    private int wiek; no usages  
    private int numerIndeksu; no usages  
}
```

PRZYKŁAD DOKUMENTACJI KLASY

```
/**
 * Klasa reprezentująca studenta
 * Przechowuje informacje o imieniu, nazwisku, wieku i numerze indeksu studenta
 *
 * @author Ja
 * @version 1.0
 * @since 1.0
 */
public class Student { no usages
    private String imie; no usages
    private String nazwisko; no usages
    private int wiek; no usages
    private int numerIndeksu; no usages

    // ...
}
```


PRZYKŁAD DOKUMENTACJI KLASY

```
/**  
 * Klasa reprezentująca studenta.  
 * Przechowuje informacje o imieniu, nazwisku, wieku i numerze indeksu studenta.  
 *  
 * @author Ja  
 * @version 1.0  
 * @since 1.0  
 */
```

```
|  
Klasa reprezentująca studenta. Przechowuje informacje o imieniu, nazwisku, wieku i numerze indeksu  
studenta.  
  
Since: 1.0  
  
Author: Ja  
  
Version: 1.0
```

PRZYKŁAD DOKUMENTACJI METODY

```
/**
 * Tworzy nowego studenta z podanymi danymi.
 *
 * @param imie      Imię studenta.
 * @param nazwisko   Nazwisko studenta.
 * @param wiek       Wiek studenta.
 * @param numerIndeksu Numer indeksu studenta.
 */
public Student(String imie, String nazwisko, int wiek, int numerIndeksu) {
    this.imie = imie;
    this.nazwisko = nazwisko;
    this.wiek = wiek;
    this.numerIndeksu = numerIndeksu;
}
```

PRZYKŁAD DOKUMENTACJI METODY

```
/**
 * Ustawia numer indeksu studenta.
 *
 * @param numerIndeksu Numer indeksu studenta.
 * @throws IllegalArgumentException Jeśli numer indeksu jest ujemny.
 */
public void setNumerIndeksu(int numerIndeksu) { no usages
    if (numerIndeksu < 0) {
        throw new IllegalArgumentException("Numer indeksu nie może być ujemny");
    }
    this.numerIndeksu = numerIndeksu;
}
```

GENEROWANIE DOKUMENTACJI

Użycie polecenia w terminalu:

```
javadoc -d doc *.java
```

Opcje generowania:

-d doc - Katalog docelowy dokumentacji.

-author - Informacje o autorach.

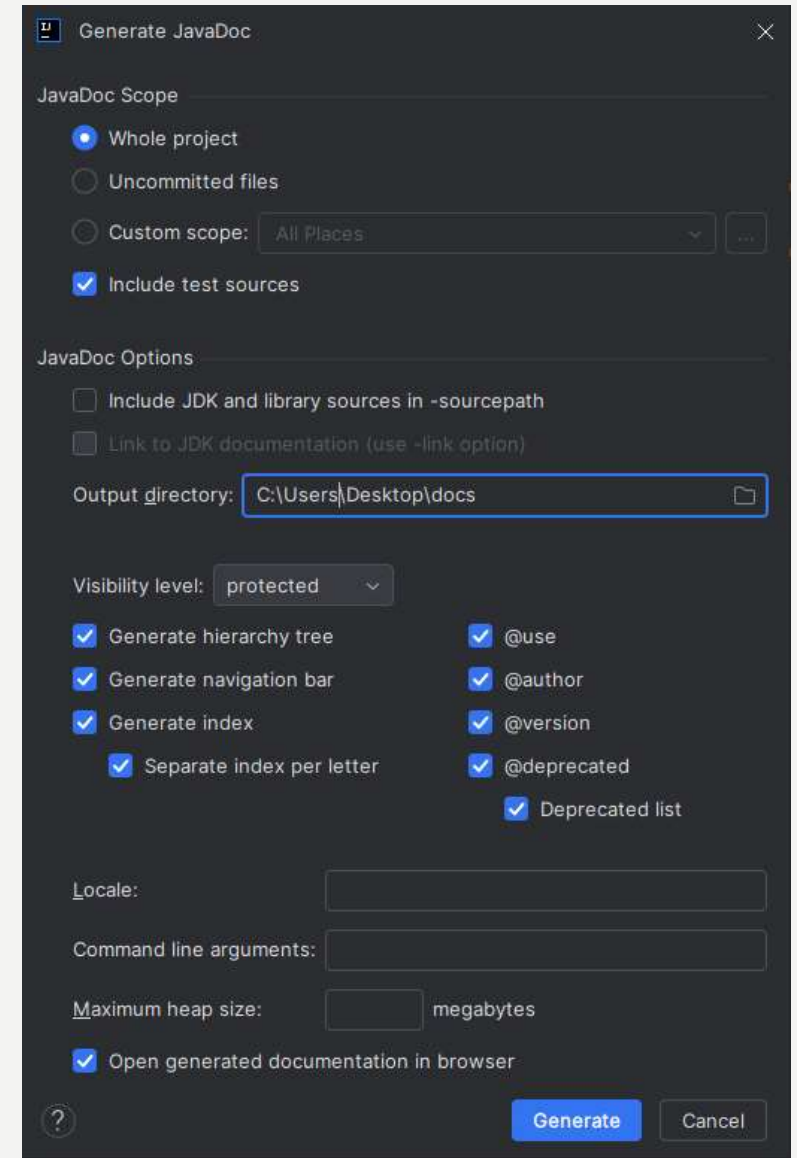
-version - Informacje o wersjach.

-private - Dokumentowanie prywatnych elementów.

-public - Dokumentowanie publicznych elementów.

GENEROWANIE JAVADOC W INTELLIJ IDEA

1. Otwórz projekt w IntelliJ IDEA.
2. Przejdź do „Tools” → „Generate JavaDoc”.
3. Wybierz odpowiednie opcje w oknie „Generate JavaDoc”.
4. Skonfiguruj „Output directory” wybierając miejsce gdzie zostanie wygenerowana dokumentacja oraz wciśnij „Generate”.
5. Dokumentację następnie możesz otworzyć w przeglądarce.



GENEROWANIE DOKUMENTACJI

```
PS C:\Users\krolb\OneDrive\Pulpit\JacaDocExample\src\main\java\org\example> javadoc -encoding UTF-8 -d doc Student.java
Loading source file Student.java...
Constructing Javadoc information...
Building index for all the packages and classes...
Standard Doclet version 17.0.9+11-LTS-201
Building tree for all the packages and classes...
Generating doc\org\example\Student.html...
Generating doc\org\example\package-summary.html...
Generating doc\org\example\package-tree.html...
Generating doc\overview-tree.html...
Building index for all classes...
Generating doc\allclasses-index.html...
Generating doc\allpackages-index.html...
Generating doc\index-all.html...
Generating doc\index.html...
Generating doc\help-doc.html...
PS C:\Users\krolb\OneDrive\Pulpit\JacaDocExample\src\main\java\org\example>
```

GENEROWANIE DOKUMENTACJI

legal	14.03.2025 15:14	Folder plików	
org	14.03.2025 15:14	Folder plików	
resources	14.03.2025 15:14	Folder plików	
script-dir	14.03.2025 15:14	Folder plików	
allclasses-index.html	14.03.2025 15:18	Brave HTML Document	3 KB
allpackages-index.html	14.03.2025 15:18	Brave HTML Document	3 KB
element-list	14.03.2025 15:18	Plik	1 KB
help-doc.html	14.03.2025 15:18	Brave HTML Document	9 KB
index.html	14.03.2025 15:18	Brave HTML Document	2 KB
index-all.html	14.03.2025 15:18	Brave HTML Document	4 KB
jquery-ui.overrides.css	14.03.2025 15:18	CSS Source File	1 KB
member-search-index.js	14.03.2025 15:18	JSFile	1 KB
module-search-index.js	14.03.2025 15:18	JSFile	1 KB
overview-tree.html	14.03.2025 15:18	Brave HTML Document	3 KB
package-search-index.js	14.03.2025 15:18	JSFile	1 KB
script.js	14.03.2025 15:18	JSFile	5 KB
search.js	14.03.2025 15:18	JSFile	13 KB
stylesheet.css	14.03.2025 15:18	CSS Source File	20 KB
tag-search-index.js	14.03.2025 15:18	JSFile	1 KB
type-search-index.js	14.03.2025 15:18	JSFile	1 KB

GENEROWANIE DOKUMENTACJI



The screenshot shows a web browser window with the address bar displaying the file path: C:/Users/krolb/OneDrive/Pulpit/lacDocExample/src/main/java/org/example/doc/org/example/package-summary.html. The browser's address bar also shows the text "Plik". The page has a navigation bar with links: PACKAGE, CLASS TREE, INDEX, and HELP. Below the navigation bar, there is a search bar with the text "SEARCH" and a search icon. The main content area displays the package name "Package org.example" and the package declaration "package org.example;". Below this, there is a section titled "Classes" with a table listing the classes in the package. The table has two columns: "Class" and "Description". The table contains one row with the class name "Student" and the description "Klasa reprezentująca studenta."

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

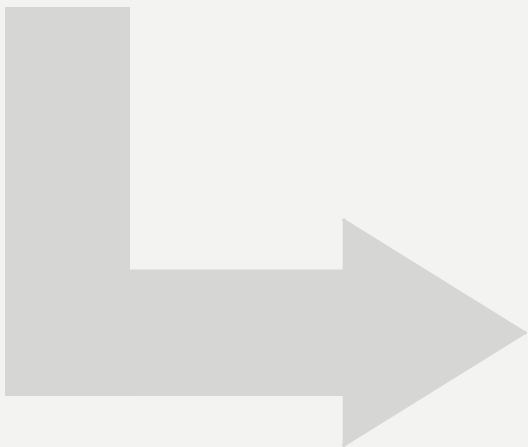
SEARCH

Package org.example

package org.example

Classes

Class	Description
Student	Klasa reprezentująca studenta.



Package org.example

package org.example

Classes

Class

Description

Student

Klasa reprezentująca studenta.

GENEROWANIE DOKUMENTACJI

< > ↺

Plik C:/Users/krolb/OneDrive/Pulpit/JacDocExample/src/main/java/org/example/doc/org/example/Student.html

PACKAGE **CLASS** TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package org.example

Class Student

java.lang.Object[Ⓢ]
org.example.Student

public class Student
extends Object[Ⓢ]

Klasa reprezentująca studenta. Przechowuje informacje o imieniu, nazwisku, wieku i numerze indeksu studenta.

Since:
1.0

Constructor Summary

Constructors

Constructor	Description
Student()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	setNumerIndeksu(int numerIndeksu)	Ustawia numer indeksu studenta.

Methods inherited from class java.lang.Object[Ⓢ]

clone[Ⓢ], equals[Ⓢ], finalize[Ⓢ], getClass[Ⓢ], hashCode[Ⓢ], notify[Ⓢ], notifyAll[Ⓢ], toString[Ⓢ], wait[Ⓢ], wait[Ⓢ], wait[Ⓢ]

PRZYKŁADOWY WYGLĄD JAVADOC

OVERVIEW PACKAGE CLASS TREE INDEX HELP	
SEARCH	
Packages	
Package	Description
calculator	
exceptions	
textutils	

OVERVIEW PACKAGE CLASS TREE INDEX HELP	
PACKAGE: DESCRIPTION RELATED PACKAGES CLASSES AND INTERFACES	
SEARCH	
Package calculator	
package calculator	
Classes	
Class	Description
AdvancedCalculator	Klasa AdvancedCalculator zapewnia bardziej zaawansowane operacje arytmetyczne.
BasicCalculator	Klasa BasicCalculator zapewnia podstawowe operacje arytmetyczne.
CalculatorUtils	Klasa pomocnicza zawierająca metody związane z kalkulatorem.

OVERVIEW PACKAGE CLASS TREE INDEX HELP	
PACKAGE: DESCRIPTION RELATED PACKAGES CLASSES AND INTERFACES	
SEARCH	
Package exceptions	
package exceptions	
All Classes and Interfaces Classes Exception Classes	
Class	Description
ExceptionHandler	Klasa zapewniająca metody do obsługi wyjątków.
ExceptionThrower	Klasa służąca do demonstracji rzucania różnych wyjątków.

PRZYKŁADOWY WYGLĄD JAVADOC

OVERVIEW

PACKAGE

CLASS

TREE

INDEX

HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

SEARCH

Package exceptions

Class CustomException

java.lang.Object

java.lang.Throwable

java.lang.Exception

exceptions.CustomException

All Implemented Interfaces:

Serializable

public class CustomException

extends Exception

CustomException to przykładowy wyjątek zdefiniowany przez użytkownika.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor	Description
CustomException()	Konstruktor domyślny bez komunikatu.
CustomException(String message)	Konstruktor przyjmujący komunikat o błędzie.

Method Summary

Methods inherited from class java.lang.Throwable

addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

PRZYKŁADOWY WYGLĄD JAVADOC

OVERVIEW PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Details

StringReverser

`public StringReverser()`

Method Details

reverse

`public String reverse(String input)`

Odwraca znaki w podanym ciągu.

Parameters:

input - ciąg znaków do odwrócenia

Returns:

nowy ciąg z odwróconymi znakami

reverseWords

`public String reverseWords(String input)`

Odwraca kolejność słów w podanym ciągu. Przyjmuje, że słowa są rozdzielone spacjami.

Parameters:

input - ciąg, którego słowa zostaną odwrócone

Returns:

nowy ciąg z odwróconą kolejnością słów

ANALIZA SKŁADOWYCH JAVADOC

Analiza składowych Javadoc to dokładne omówienie poszczególnych elementów dokumentacji generowanej za pomocą Javadoc. Obejmuje to:

- Komentarze ogólne (np.: opis klasy, pakietu).
- Komentarze dla metod (np.: opis działania, parametru, zwracanej wartości).
- Tagowanie (np.: zastosowanie `@param`, `@return`, `@throws`).
- Struktura wygenerowanej dokumentacji.

PODSUMOWANIE I WYZWANIA



Ułatwia dokumentowanie kodu i jego zrozumienie.



Pomaga w utrzymaniu wysokiej jakości oprogramowania.



Wymaga regularnej aktualizacji, co bywa wyzwaniem.



Powinien być stosowany w kluczowych elementach kodu.

**DZIĘKUJEMY ZA
UWAGĘ!**

GRUPA POMARAŃCZOWA