Secure OOP with Java

Lecture - Unit 02

Claudia Maderthaner <claudia.maderthaner@fh-hagenberg.at>

Data Types

- Java is a **strongly typed language**.
- Every variable must have a declared type.

Types of Data Types

- Primitive Data Types
- Reference Data Types
- Arrays

Primitive Data Types

- Integral Data Types
- Floating-Point Data Types
- The char Type
- The boolean Type

Type	Range		Length	Default	Wrapper
`byte`	$-2^7 \dots 2^7-1$	-128 … 127	1 Byte	`0`	`Byte`
`short`	$-2^{15} \dots 2^{15}-1$	- 32 768 … 32 767	2 Byte	`0`	`Short`
`int`	$-2^{31} \dots 2^{31}-1$	-2 147 483 648 … -2 147 483 647	4 Byte	`0`	`Integer`
`long`	$-2^{63} \dots 2^{63}-1$	-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807	8 Byte	`0L`	`Long`
`float`	$1.4 \times 10^{-45} \dots 3.4 \times 10^{38}$	7 significant digits (IEEE 754)	4 Byte	`0.0f`	`Float`
`double`	$4.9 \times 10^{-324} \dots 1.7 \times 10^{308}$	15 significant digits (IEEE 754)	8 Byte	`0.0d`	`Double`
`boolean`	`true`, `false`		1 Byte	`false`	`Boolean`
`char`	16-bit Unicode (UTF-16)	0x0000 … 0xFFFF	2 Byte	`\u0000`	`Character`

Primitive Literals

```
byte b = 100;
short s = 10000;
int i = 100000;
```

Primitive Literals

```
long longExample = 123L;
double doubleExample1 = 123.4;
double doubleExample2 = 1.234e2; // 123.4
float floatExample = 123.4f;
```

Primitive Literals

```
int decimalExample = 26;
int octalExample = 032;
int hexadecimalExample = 0x1a;
int binaryExample = 0b11010;
```

Primitive Literals

```
long groupedNumber = 123_456_783_456L;

float pi = 3.14_15f;
long hexWords = 0xcafe_cafe;
long someByte = 0b0010_0101;
```

The char Type

- Describe individual characters
- Some Unicode characters require **two** char values
- Literal values are enclosed in single quotes.

char Literals

```
char capitalC = 'C';
```

Escape Sequences

Escape Sequence	Name	Unicode Value
`\b`	backspace	`\u0008`
`\t`	tab	`\u0009`
`\n`	linefeed	`\u000a`
`\r`	carriage return	`\u000d`
`\"`	double quote	`\u0022`
`\'`	single quote	`\u0027`
`\\`	backslash	`\u005c`

The boolean Type

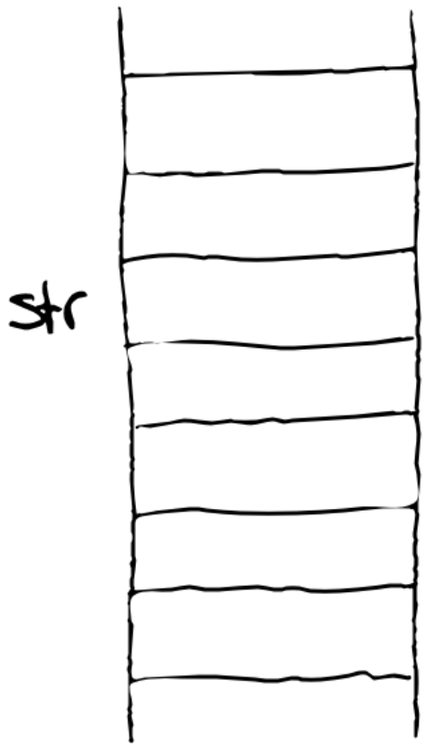
- Two values: `true` and `false`
- Used for evaluating logical conditions
- No conversion between integers and `boolean` values

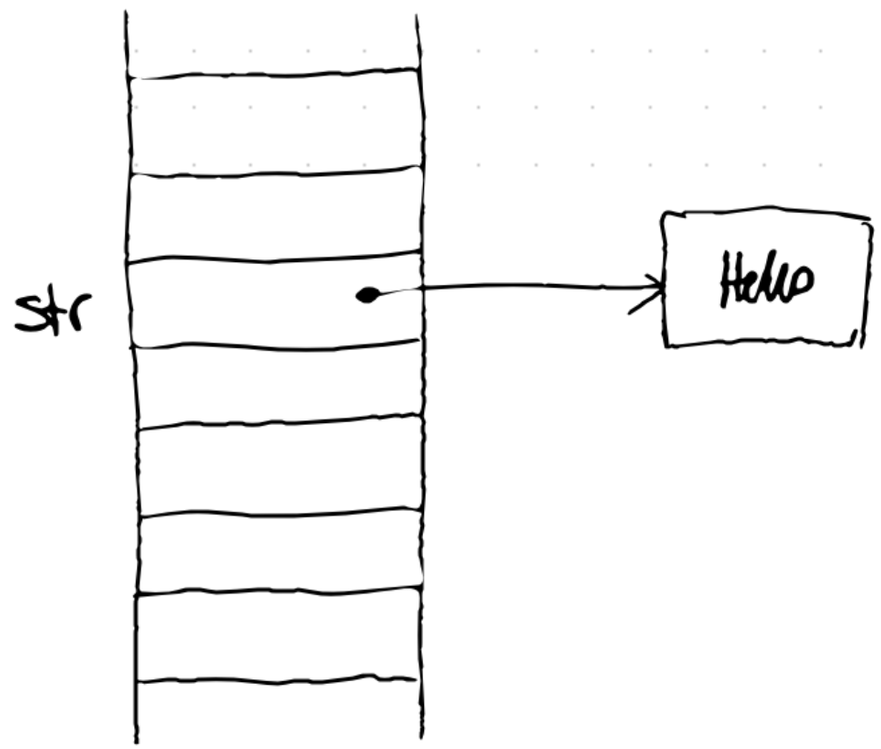
boolean Literals

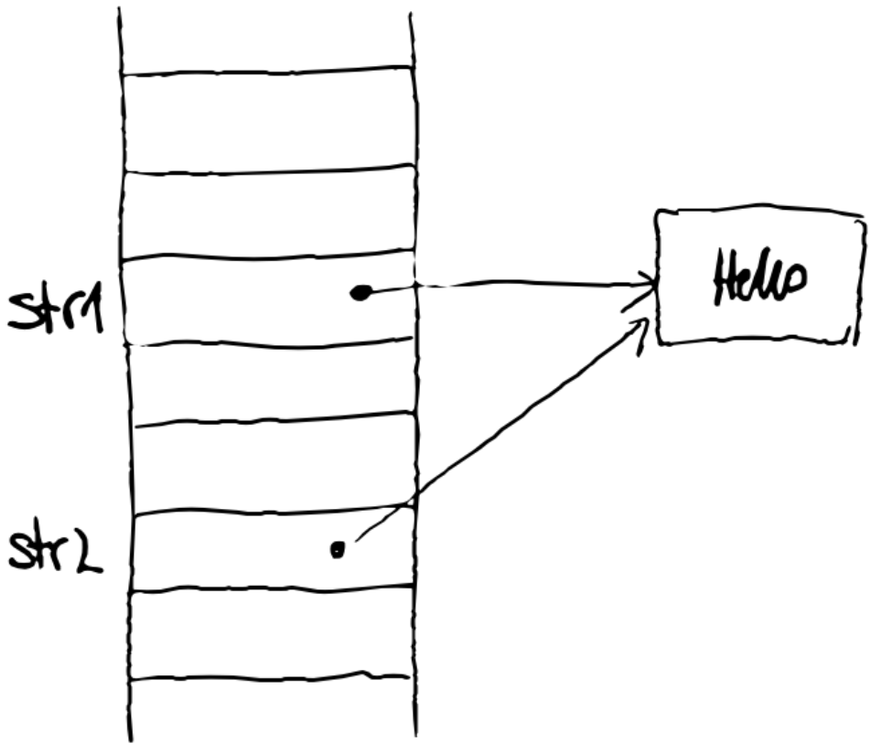
```
boolean success = true;
boolean failure = false;
```

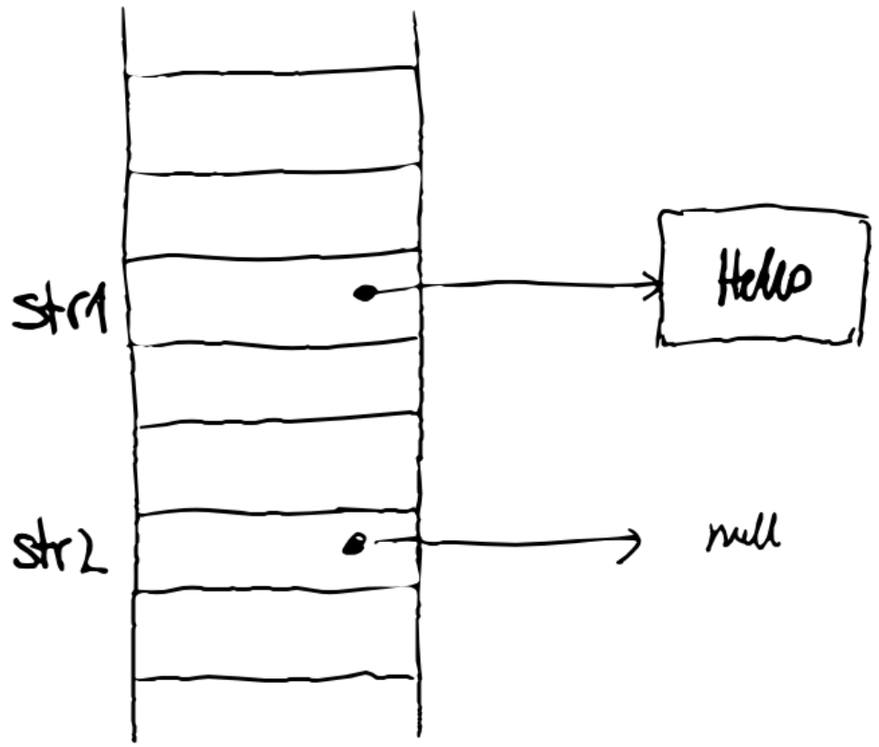
Reference Data Types

- Variable holds the reference to an object in memory.
- An object may be referenced from more than one variable.


str


str
Hello


str1
Hello
str2


str1
Hello
str2
null

String

- Predefined class (no primitive type)
- Sequences of Unicode characters

String Literals

```
String line = "Lorem ipsum dolor sit amet, consectetuer " +
    "adipiscing elit. Aenean commodo ligula eget dolor.";
```

String Literals

```
String block = """
    Lorem ipsum dolor sit amet, consectetuer
    adipiscing elit. Aenean commodo ligula eget dolor.""";
```

Arrays

- Ordered collection of elements
- The type of elements is called the **base type**
- The number of elements it holds is a fixed attribute called `length`

Accessing Array Elements

```
int[] a = new int[10] // => { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
for (int i = 0; i < a.length; i++) {
    a[i] = i;
}
```

```
a[10] = 10;
// -> Exception java.lang.ArrayIndexOutOfBoundsException
```

null

- `null` has no type
- Every variable of a reference type may have the value `null`

Typecasts

long ↔ int ↔ short ↔ byte

Implicit Typecasts

Widening Conversion

`long` ← `int` ← `short` ← `byte`

Explicit Typecasts

Narrowing Conversion

`long` → `int` → `short` → `byte`

```
int myInteger = 42;
byte myByte = myInteger; // does not compile

byte myOtherByte = (byte) myInteger;
```

Typecasts Overview

↓ from / → to	`boolean`	`byte`	`short`	`char`	`int`	`long`	`float`	`double`
`boolean`	-	✗	✗	✗	✗	✗	✗	✗
`byte`	✗	-	implicit	explicit	implicit	implicit	implicit	implicit
`short`	✗	explicit	-	explicit	implicit	implicit	implicit	implicit
`char`	✗	explicit	explicit	-	implicit	implicit	implicit	implicit
`int`	✗	explicit	explicit	explicit	-	implicit	implicit*	implicit
`long`	✗	explicit	explicit	explicit	explicit	-	implicit*	implicit*
`float`	✗	explicit	explicit	explicit	explicit	explicit	-	implicit
`double`	✗	explicit	explicit	explicit	explicit	explicit	explicit	-

* probable loss of significant digits

Syntax

Variables

Declaration

```
int a;
```

Initialization

```
a = 10;

int b = 20;
```

Declaring Arrays

```
int[] a; //preferred way of declaration

int b[];
```

Initializing Arrays

```
int[] a = {1, 2, 4, 5};

int[] b = new int[3]
b = new int[] {6, 7 ,8}
```

Identifiers

- Name of a language element
- Unlimited length (practically it needs to be less than 64k)
- Cannot begin with a digit
- Cannot be equal to a (reserved) keyword, `null` literal, or `boolean` literal
- Identifiers are case sensitive

Keywords

`abstract`	`assert`	`boolean`	`break`	`byte`
`case`	`catch`	`char`	`class`	`const` (reserved)
`continue`	`default`	`do`	`double`	`else`
`enum`	`exports` (restricted)	`extends`	`final`	`finally`
`float`	`for`	`if`	`goto` (reserved)	`implements`
`import`	`instanceof`	`int`	`interface`	`long`
`module` (restricted)	`native`	`new`	`non-sealed` (restricted)	`open` (restricted)
`opens` (restricted)	`package`	`permits` (restricted)	`private`	`protected`
`provides` (restricted)	`public`	`record` (restricted)	`return`	`sealed` (restricted)
`short`	`static`	`strictfp`	`super`	`switch`
`synchronized`	`this`	`throw`	`throws`	`to` (restricted)
`transient`	`transitive` (restricted)	`try`	`uses` (restricted)	`var` (restricted)
`void`	`volatile`	`while`	`with` (restricted)	`yield` (restricted)

_ (underscore)

Restricted keywords are only keywords in a specific context.

Comments

Explanatory text ignored by the compiler

Single Line Comments

```
public final double E = 2.718281828459045235536; // Euler's number
```

Block Comments

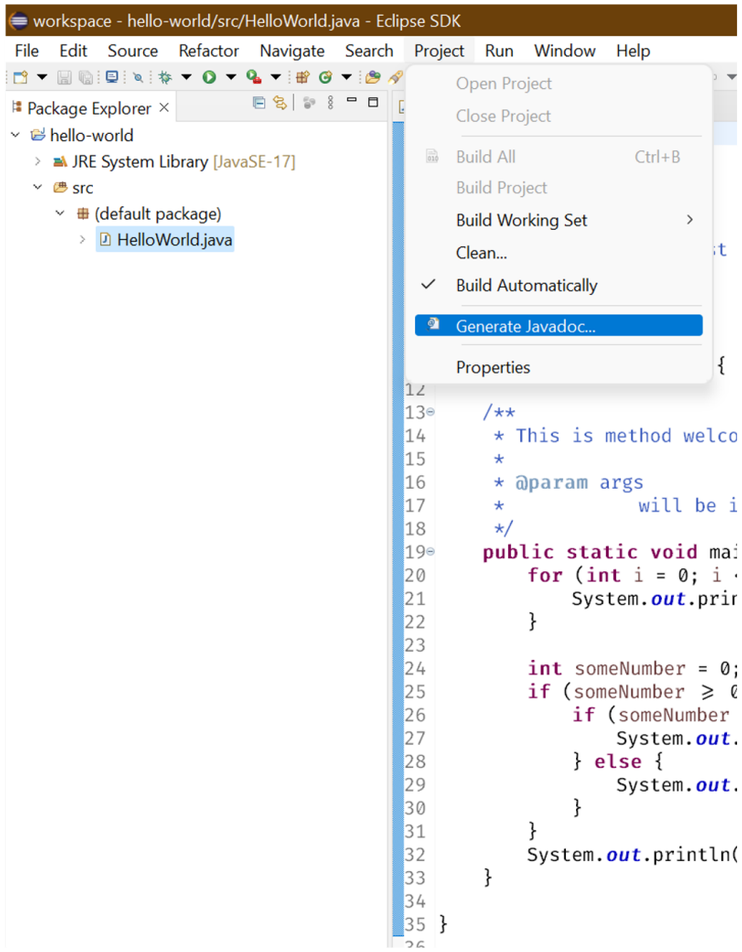
```
/*
 * Return a greeting derived from the given name.
 */
return String.format("Hello %s!", name);
```

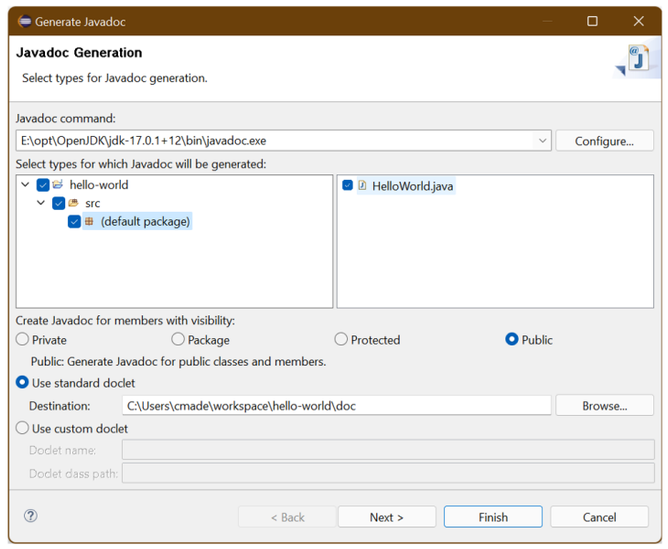
JavaDoc

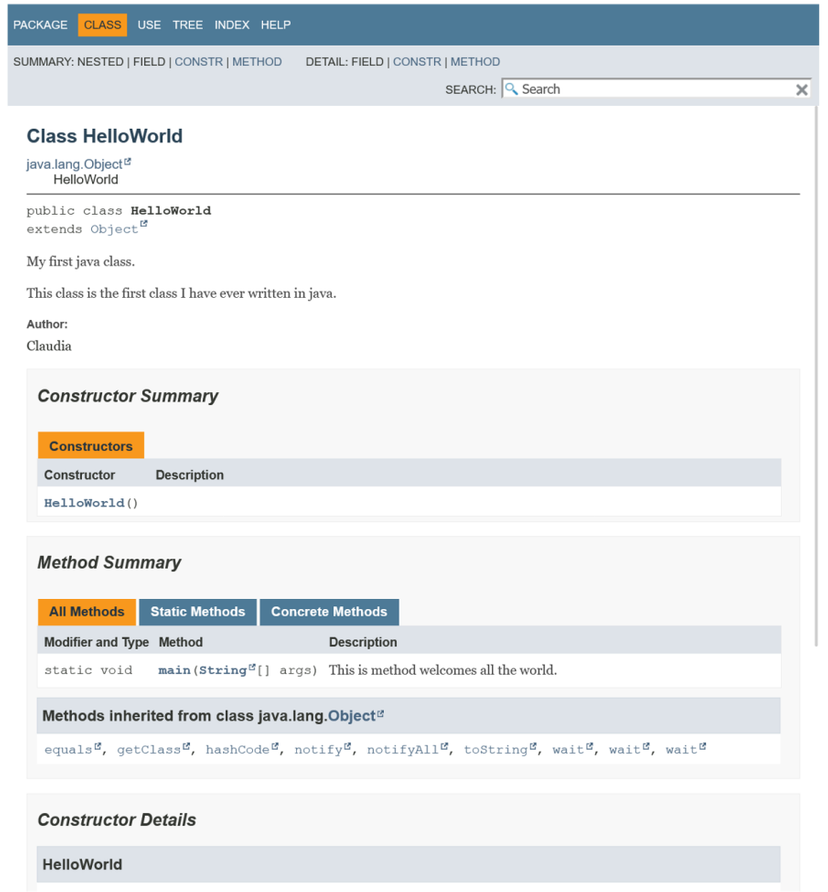
```java
/**
 * Create a greeting message.
 *
 * @param name the name to be greeted
 * @return a greeting derived from the given name
 */
public String getGreeting(String name) {
    return String.format("Hello %s!", name);
}
```

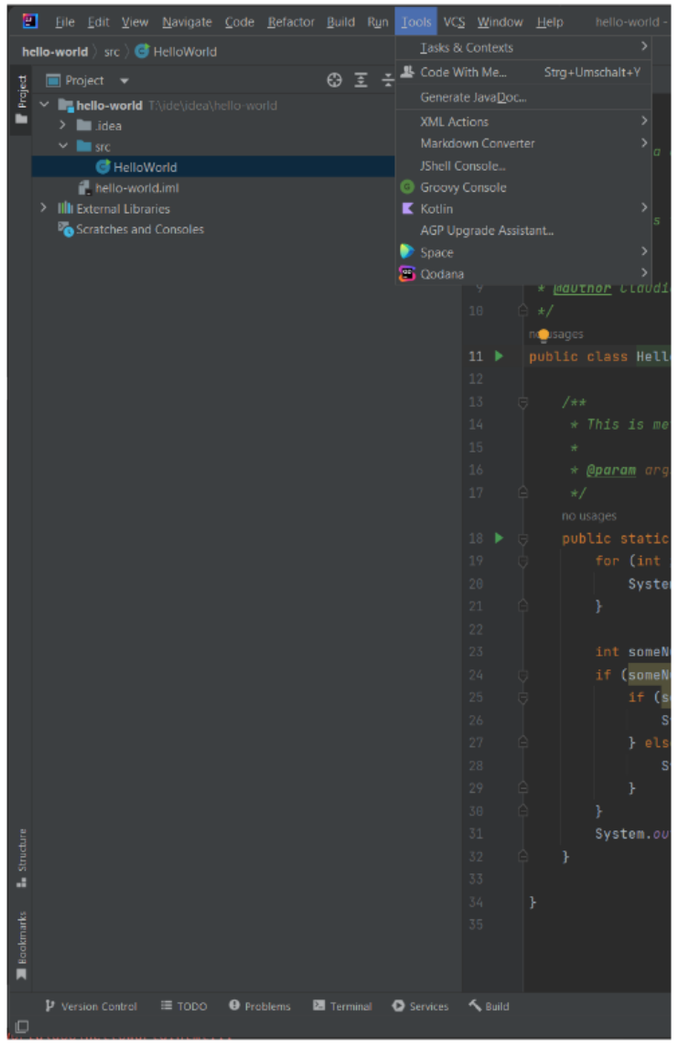
JavaDoc Tags

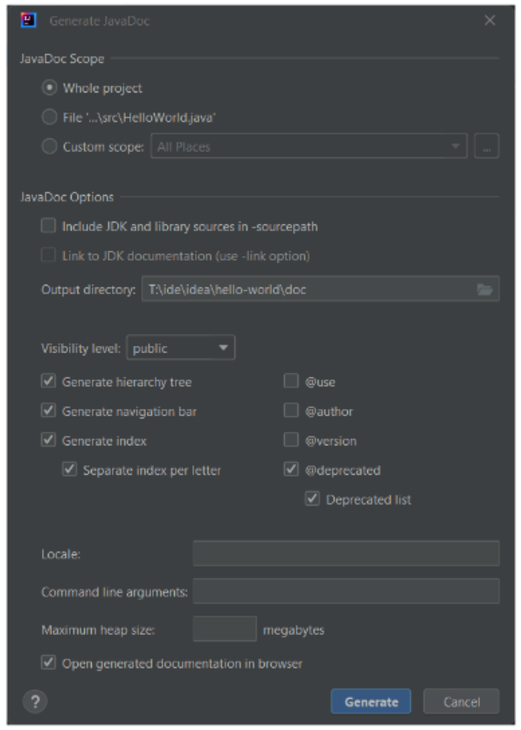
Tag	Description	Scope
`@author` name	author name	class, interface
`@version` version	version string	class, interface
`@since` version	API version when item was added	class, interface, field, method
`@see` reference	associated class name	class, interface, field, method
`@param` name description	parameter name and description	method
`@return` description	description of return value	method
`@throws` class name description	exception name and description	method
`@deprecated` description	declares item to be obsolete	class, interface, field, method
`{@inheritDoc}`	copies the description from the overridden method	overriding method
`{@link reference}`	link to other symbol	class, interface, field, method

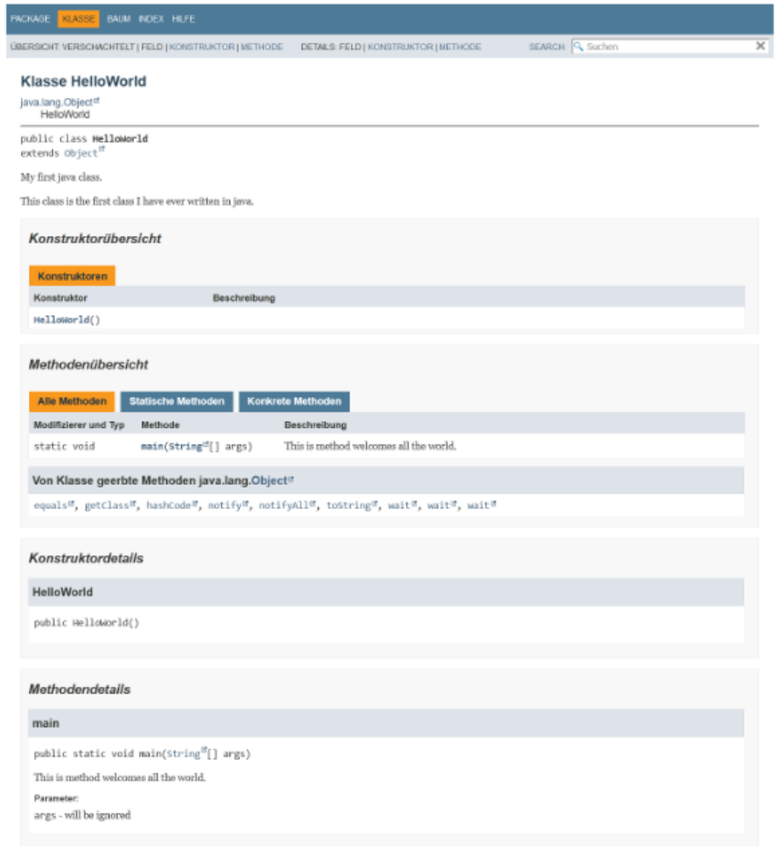

workspace - hello-world/src/HelloWorld.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
hello-world
JRE System Library [JavaSE-17]
src
(default package)
HelloWorld.java
Open Project
Close Project
Build All Ctrl+B
Build Project
Build Working Set
Clean...
Build Automatically
Generate Javadoc...
Properties
/**
* This is method welco
*
* @param args
* will be i
*/
public static void ma
for (int i = 0; i
System.out.prir
}
int someNumber = 0;
if (someNumber ≥ 0
if (someNumber
System.out.
} else {
System.out.
}
}
System.out.println(
}
}


Generate Javadoc
Javadoc Generation
Select types for Javadoc generation.
Javadoc command:
E:\opt\OpenJDK\jdk-17.0.1+12\bin\javadoc.exe
Configure...
Select types for which Javadoc will be generated:
hello-world
src
(default package)
HelloWorld.java
Create Javadoc for members with visibility:
Private
Package
Protected
Public
Public: Generate Javadoc for public classes and members.
Use standard doclet
Destination:
C:\Users\cmade\workspace\hello-world\doc
Browse...
Use custom doclet
Doclet name:
Doclet class path:
< Back
Next >
Finish
Cancel

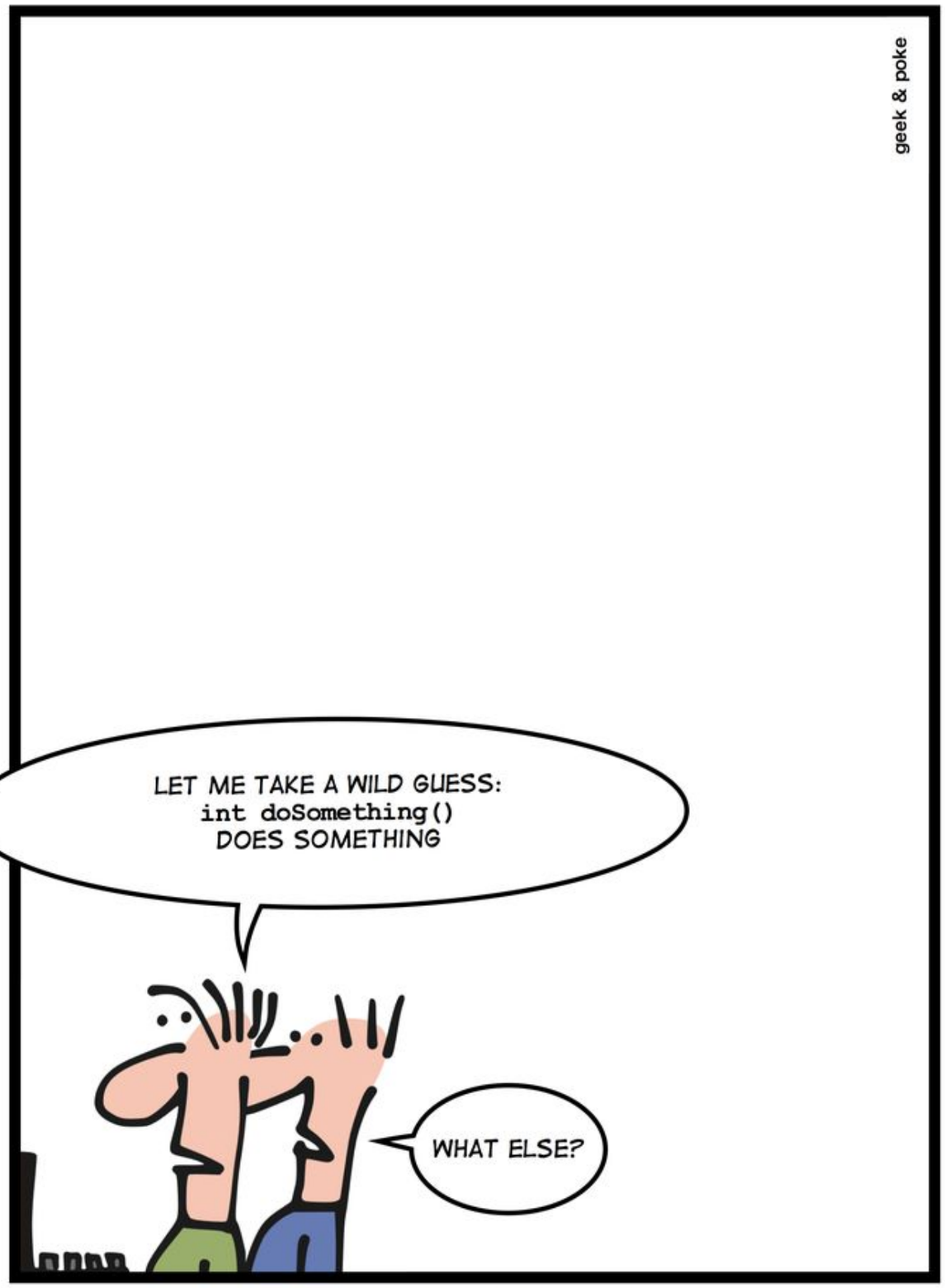

PACKAGE CLASS USE TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
SEARCH:
Search
Class HelloWorld
java.lang.Object
HelloWorld
public class HelloWorld
extends Object
My first java class.
This class is the first class I have ever written in java.
Author:
Claudia
Constructor Summary
Constructors
Constructor
Description
HelloWorld()
Method Summary
All Methods
Static Methods
Concrete Methods
Modifier and Type
Method
Description
static void
main(String[] args)
This is method welcomes all the world.
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Details
HelloWorld


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help hello-world -
hello-world › src › HelloWorld
Project
hello-world T:\ide\idea\hello-world
.idea
src
HelloWorld
hello-world.iml
External Libraries
Scratches and Consoles
Tasks & Contexts
Code With Me... Strg+Umschalt+Y
Generate JavaDoc...
XML Actions
Markdown Converter
JShell Console...
Groovy Console
Kotlin
AGP Upgrade Assistant...
Space
Qodana
* @author Claudi
*/
no usages
public class Hell
/**
* This is me
*
* @param arg
*/
no usages
public static
for (int
Syste
}
int someN
if (someN
if (s
S
} els
S
}
}
System.ou
}
}
Structure
Bookmarks
Version Control
TODO
Problems
Terminal
Services
Build


Generate JavaDoc
JavaDoc Scope
Whole project
File '...\src\HelloWorld.java'
Custom scope:
All Places
JavaDoc Options
Include JDK and library sources in -sourcepath
Link to JDK documentation (use -link option)
Output directory:
T:\ide\idea\hello-world\doc
Visibility level:
public
Generate hierarchy tree
@use
Generate navigation bar
@author
Generate index
@version
Separate index per letter
@deprecated
Deprecated list
Locale:
Command line arguments:
Maximum heap size:
megabytes
Open generated documentation in browser
Generate
Cancel


PACKAGE KLASSE BAUM INDEX HILFE
ÜBERSICHT VERSCHACHTELT | FELD | KONSTRUKTOR | METHODE
DETAILS: FELD | KONSTRUKTOR | METHODE
SEARCH:
Suchen
Klasse HelloWorld
java.lang.Object
HelloWorld
public class HelloWorld
extends Object
My first java class.
This class is the first class I have ever written in java.
Konstruktorübersicht
Konstruktoren
Konstruktor
Beschreibung
HelloWorld()
Methodenübersicht
Alle Methoden
Statische Methoden
Konkrete Methoden
Modifizierer und Typ
Methode
Beschreibung
static void
main(String[] args)
This is method welcomes all the world.
Von Klasse geerbte Methoden java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Konstruktordetails
HelloWorld
public HelloWorld()
Methodendetails
main
public static void main(String[] args)
This is method welcomes all the world.
Parameter:
args - will be ignored

SIMPLY EXPLAINED



SELF DOCUMENTING CODE

Imperative Language Features

Expressions

- produce a result, or value, when evaluated
- the value of an expression can be
  - a numeric type
  - a reference type
  - `void`
- the type of an expression is known at compile time

Operators

Operator	Type		Precedence	Description
`++`, `--`	unary	arithmetic	1	increment, decrement
`+`, `-`	unary	arithmetic	1	plus, minus
`~`	unary	integral	1	bitwise complement
`!`	unary	boolean	1	logical complements
`(type)`	unary	any	1	cast
`*`, `/`, `%`	binary	arithmetic	2	multiplication, division, remainder
`+`, `-`	binary	arithmetic	3	addition, subtraction
`+`	binary	String	3	String concatenation
`<<`	binary	integral	4	left shift
`>>`	binary	integral	4	right shift with sign extension
`>>>`	binary	integral	4	right shift with no extension
`<`, `⩽`, `>`, `⩾`	binary	arithmetic	5	numeric comparison
`instanceof`	binary	object	5	type comparison
`==`, `≠`	binary	primitive	6	equality, inequality of value
`==`, `≠`	binary	object	6	equality, inequality of reference

Operators

Operator	Type		Precedence	Description
&	binary	integral	7	bitwise AND
&	binary	boolean	7	boolean AND
^	binary	integral	8	bitwise XOR
^	binary	boolean	8	boolean XOR
|	binary	integral	9	bitwise OR
|	binary	boolean	9	boolean OR
&&	binary	boolean	10	conditional AND
||	binary	boolean	11	conditional OR
? :	ternary		12	conditional ternary operator
=	binary	any	13	assignment
*=, /=, %=, +=	binary	arithmetisch	14	assignment with operation
<<=, >>=, >>> =	binary	integral	14	assignment with operation
&=, ^=, |=	binary	integral	14	assignment with operation
+=	binary	String	14	assignment with operation

Arithmetic Conversion

```
byte a = 1;
byte b = 1;

byte c = a + b; // compile error!
byte d = (byte) (a + b);
```

Unary minus/plus

```
int a = 1;
int b = -a;
int c = -1 * a;

int i = - - - 2 + - + 3;
```

Increment/Decrement

```
int i = 10;
int j = 20;

System.out.println(++i); // 11
System.out.println(--j); // 19
System.out.println(i); // 11
System.out.println(j); // 19
```

```
int i = 10;
int j = 20;

System.out.println(i++); // 10
System.out.println(j--); // 20
System.out.println(i); // 11
System.out.println(j); // 19
```

Increment/Decrement

```
double d = 12.0;
System.out.println(--d); // 11.0

double e = 12.345;
System.out.println(++e); // 13.345
```

Increment/Decrement

```
int i = 1;
i = ++i;
System.out.println(i); // 2

int j = 1;
j = j++;
System.out.println(j); // 1
```

Increment/Decrement

```
int a = 10;
a = a++ + a + a-- - a-- + ++a;
```

Modulo

```
System.out.println(5 / 3); // 1
System.out.println(5 % 3); // 2

System.out.println(5 / -3); // -1
System.out.println(5 % -3); // 2

System.out.println(-5 / 3); // -1
System.out.println(-5 % 3); // -2

System.out.println(-5 / -3); // 1
System.out.println(-5 % -3); // -2
```

Conditional AND/OR

```
if (obj != null && obj.valid() || obj.obsolete()) {
    ...
}
```

Conditional AND/OR

```
int a = 0;
System.out.println(true || a++ == 0);
System.out.println(a);

int b = 0;
System.out.println(true | b++ == 0);
System.out.println(b);
```

```
int c = 0;
System.out.println(false && c++ == 0);
System.out.println(c);

int d = 0;
System.out.println(false & d++ == 0);
System.out.println(d);
```

bit Operators

a	b	~a	a&b	a|b	a^b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Shift Operators

```
System.out.println(4 << 2);
System.out.println(-4 << 2);

System.out.println(16 >> 2);
System.out.println(-16 >> 2);

System.out.println(16 >>> 2);
System.out.println(-16 >>> 2);
```

Ternary Operator

```
return obj == null ? "" : obj.getName();
```

Assignments

```
int counter = 1;
counter = 2;
counter = counter++;

String name = "Herbert";
```

Combined Assignments

```
int a = 10;
a = a + 1;
a = a - 3;
a = a * 5;
a = a / 4;
a = a % 2;
```

```
int a = 10;
a += 1;
a -= 3;
a *= 5;
a /= 4;
a %= 2;
```

Combined Assignments

```
boolean a = true;
boolean b = false;

a = a & b;
a = a | b;
a = a ^ b;
```

```
boolean a = true;
boolean b = false;

a &= b;
a |= b;
a ^= b;
```

Combined Assignments

```
int a = 16;

a = a << 2;
a = a >> 2;
a = a >>> 2;
```

```
int a = 16;

a <<= 2;
a >>= 2;
a >>>= 2;
```

Statements

Empty Statement

Execution of an empty statement always completes normally.

```
;
```

Blocks and Scopes

```
String question = "What's the time?";
{
    String question = "What's the time?"; // compile error
    String answer = "It's Thursday.";
    System.out.println(question);
    System.out.println(answer);
}
System.out.println(answer); // compile error
```

if Statements

```
if (condition) {
    statements;
}
```

```
if (condition) {
    statements;
} else {
    statements;
}
```

```
if (condition) {
    statements;
} else if (condition) {
    statements;
}
} else {
    statements;
}
```

```java
int guess = 73;

if (guess == 83) {
    System.out.println("You guessed my number!");
}
```

```
int guess = 73;

if (guess == 83) {
    System.out.println("You guessed my number!");
} else if (guess < 83) {
    System.out.println("Your number is too small!");
} else {
    System.out.println("Your number is too large!");
}
```

```
int someNumber = 0;
if (someNumber >=0)
if (someNumber == 0) System.out.println("first stop");
else System.out.println("second stop");
System.out.println("third stop");
```

Short-Circuit Evaluation

```
int a = 0;
int b = 10;
if (a != 0 && b / a >= 5) {
    System.out.println("Division möglich");
} else {
    System.out.println("Division durch 0 verhindert");
}
```

switch Statement

```
switch (condition) {
    case OPTION_A:
        statements;
        break;
    case OPTION_B:
        statements;
        break;
    default:
        statements
        break;
}
```

```
switch (dayOfTheWeek) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        System.out.println("This is a weekday.");
        break;

    default:
        System.out.println("This is a weekend day.");
        break;
}
```

while Statement

```
while (condition) {
    statements;
}
```

```
final int maxValue = 1_000_000;
int product = 1;

while (product < maxValue) {
    product *= 2;
}
```

do-while Statement

```
do {
    statements;
} while (condition);
```

```java
int input = 0;

Scanner scanner = new Scanner(System.in);
System.out.println("Enter a number (exit with 0)");

do {
    input = scanner.nextInt();
    System.out.println("Your input: " + input);
} while (input != 0);

scanner.close();
```

for Statement

```
for (expression; condition; step) {
    statements;
}
```

```
System.out.println("It's the final countdown:");

for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}
```

for-each Statement

```
String[] strings = {"one", "two", "three"};

for (String item: strings) {
    System.out.println(item);
}
```

break/continue

```
int i = 0;
while (true) {
    i++;
    int j = i * 27;
    if (j == 1269) {
        break; // Out of loop
    }
    if (i % 10 != 0) {
        continue; // Top of loop
    }
    System.out.println(i);
}
```

Labels

```
OuterLoop: for (int i = 2;; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0) {
            continue OuterLoop;
        }
    }
    System.out.println(i);
    if (i == 37) {
        break OuterLoop;
    }
}
```

return Statement

```
public String getGreeting(String name) {
    return String.format("Hello %s!", name);
}
```

Unreachable Statements

If a statement cannot be executed because it is **unreachable** leads to a compile-time error.

Contact

Moodle Discussion Board

claudia.maderthaner@fh-hagenberg.at