Software Engineering and Programming Basics

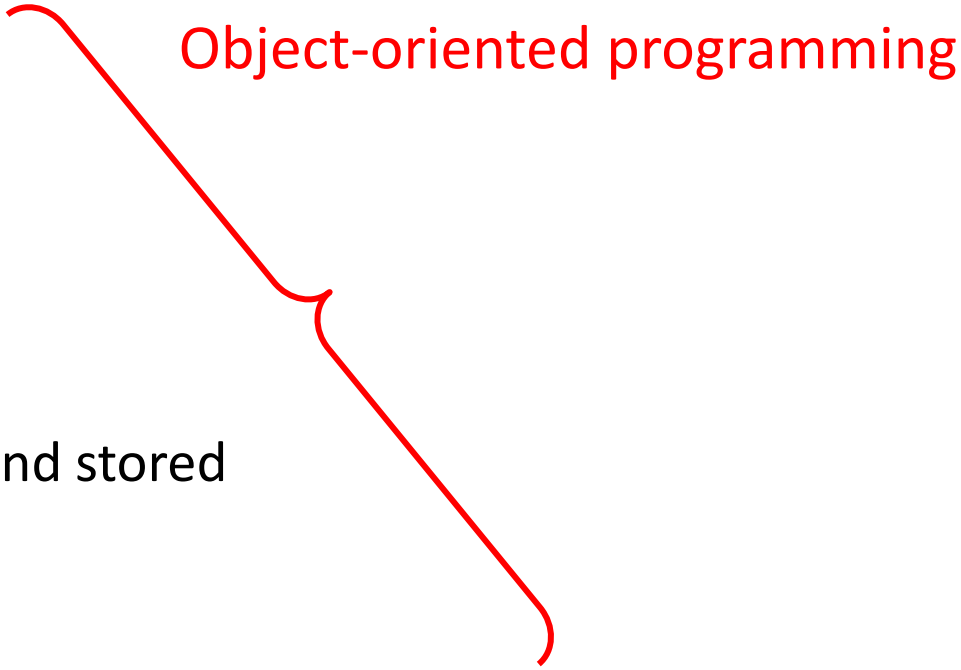
Data types, expressions, and methods

Authors of slides:
Prof. Dr.-Ing. Janet Siegmund
Prof. Dr.-Ing. Norbert Siegmund
Prof. Christian Lengauer
Partly extracted from script of PD Dr. Christian Bachmaier

- Bring your laptops to the excercise

Java Structure I

- Java consists of two parts:
  - Structure
    - Classes
    - Objects
    - Data types
    - How data/information is described and stored
  - Behavior
    - Methods
    - Which operations are executed on data
- Example:
  - Components of a car vs. driving, filling up gas, scrapping it



Learning Goals

- Introduction to behavior of Java programs (i.e., how to describe what they should do)
- Get to know methods (for reusing statements)
- Get to know fundamental statements in Java
- Get to know data types, especially primitive data types

Primitive and Complex Data Types in Java

Why Data Types?

- What is the result?

'1' + '2' =99

1 + 2 = 3

'1' + 2 =51

```
class Person {
        givenName;
        name;
        age;
        address;
}
```

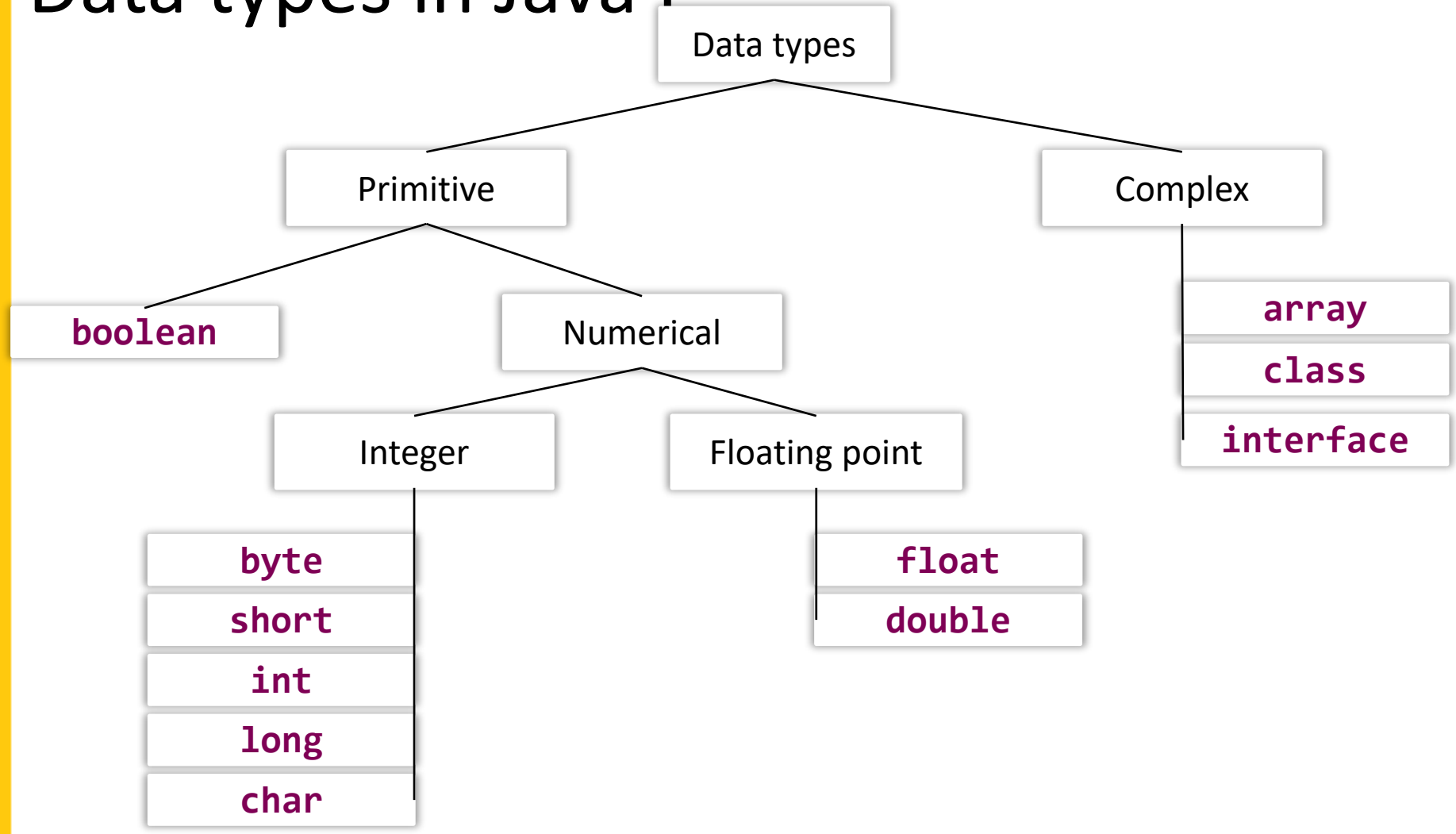
→

What can I do with these?
Print to screen? Calculate? Combine? Search?

Does `givenName` + `name` make sense?
What would be the result

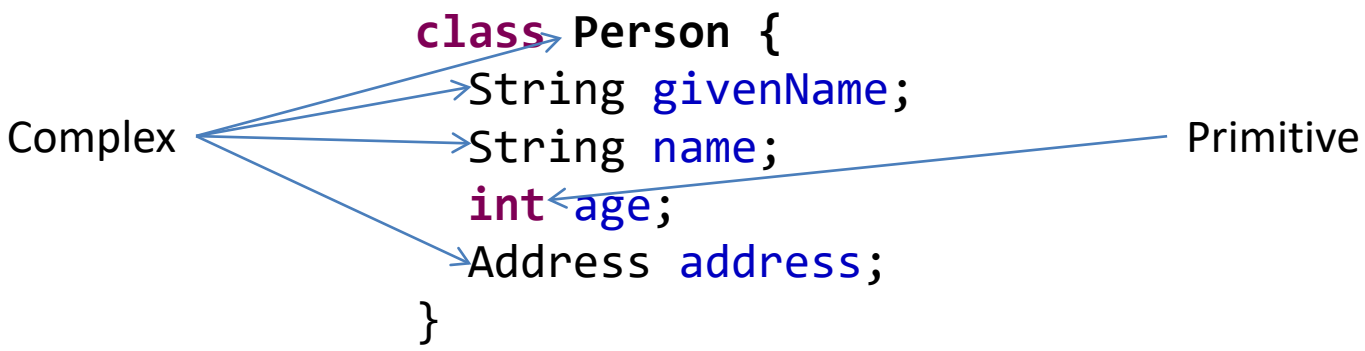
- Data types define, what we can do with variables and constants, that is, which operations are necessary

Data types in Java I



Data Types in Java II

- **Primitive data types** are fundamental elements of every program
- **Complex data types** are composed of primitive data types and other complex data types



Internal Representations of Integers

- Computer understands: Current or no current
- Source of storage was 1 bit: Either 1 or 0
- Thus, numbers are represented as binary numbers

8 Bit

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
[0;1]	[0;1]	[0;1]	[0;1]	[0;1]	[0;1]	[0;1]	[0;1]

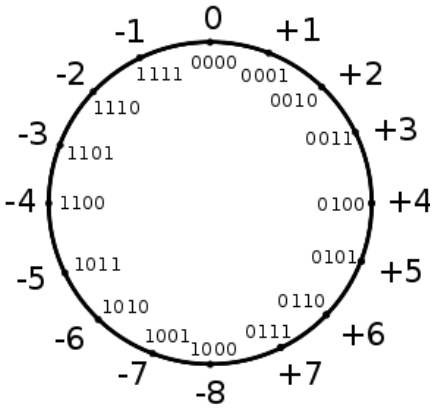
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
10 =	0	0	0	0	1	0	1	0
=					8	+	2	

Primitive Data Types I

Type	Length in bytes	Range
`byte`	1	-2^7 to $2^7 - 1$ (-128 … 127)
`short`	2	-2^{15} to $2^{15} - 1$ (-32768 … 32767)
`int`	4	-2^{31} to $2^{31} - 1$ (-2147483648 … 2147483647)
`long`	8	-2^{63} to $2^{63} - 1$ (–9223372036854775808 … 9223372036854775807)
`float`	4	±(1.40239846E–45f . . . 3.40282347E+38f)
`double`	8	±(4.94065645841246544E–324 . . . 1.79769313486231570E+308)

Negative Numbers in Binary

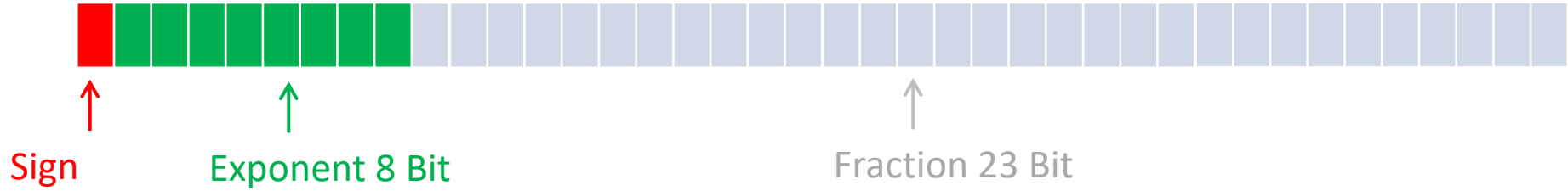
- Ones' complement
  - Bit complement of absolute value
  - $+5 = 0101_2$; $-5 = 1010_2$
  - Range $[-2^{b-1}+1; 2^{b-1}-1]$; b = 4 Bits
  - Problem: $0 = 0000_2 = 1111_2$
- Two‘s complement
  - Bit complement of absolute value + 1
  - $-5 = 1010_2 + 1_2 = 1011_2$
  - Range $[-2^{b-1}; 2^{b-1}-1]$; b = 4 Bits
  - Careful with overflow
  - 5+5 = -6



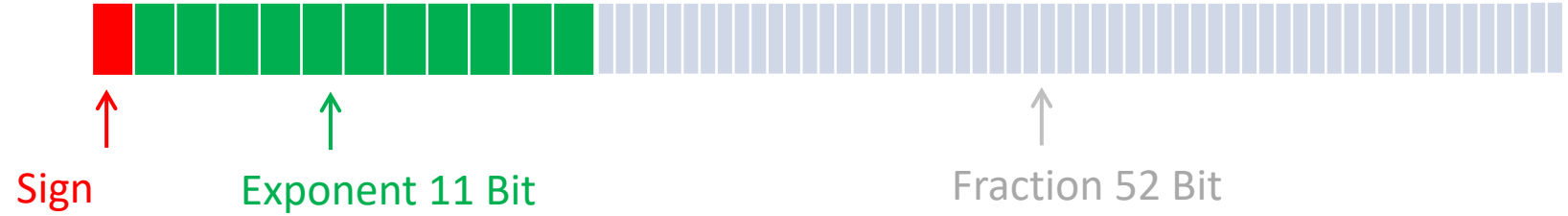
	0	1	0	1
+	0	1	0	1
=	1	0	1	0

Internal Representation of Floating Point Numbers

- **`float:`** Floating Point according to IEEE 754-Standard with 32 Bit



- **`double:`** Fließkommazahl nach IEEE 754-Standard mit 64 Bit



- Fraction = binary coded posive natural number
- Exponent = binary coded posive natural number
- Number x = Fraction * 10^2^Exponent

Example

	Sign	Exponent	Mantissa
Value:	+1	2^4	1.4656250476837158
Encoded as:	0	131	3905946
Binary:	☐	☑ ☐ ☐ ☐ ☐ ☐ ☑ ☑	☐ ☑ ☑ ☑ ☐ ☑ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☑ ☐

You entered	23.45	+1
Value actually stored in float:	23.450000762939453125	
Error due to conversion:	7.62939453125E-7	-1
Binary Representation	01000001101110111001100110011010	
Hexadecimal Representation	0x41bb999a	

Primitive Data Types II

Type	Length in Bytes	Range
char	2	16-Bit Unicode Character ('\u0000' … '\uffff')

- Characters:
  - All characters on the keyboard
  - National special characters ('ä', 'ß', 'ê,…)
  - Characters of other languages (e.g., k
  - Zeichen aus anderen Sprachen (z.B., Cyrillic)
  - In Java labeled with „' “
- Internal Representation:
  - 4-digit number as hexadecimal value represent characteres ('a' = '\u0061')
  - Complete tabel(s) http://www.unicode.org/
  - \u lets Java know that it is a unicode character

Primitive Datentypen III

- **`boolean:`** Two states: **`true`** oder **`false`**
- Convention: primitive data types start with a small letter, complex data types start with a capital letter

```
class Person {
  String vorname;
  String nachname;
  int alter;
  Ort wohnort;
}
```

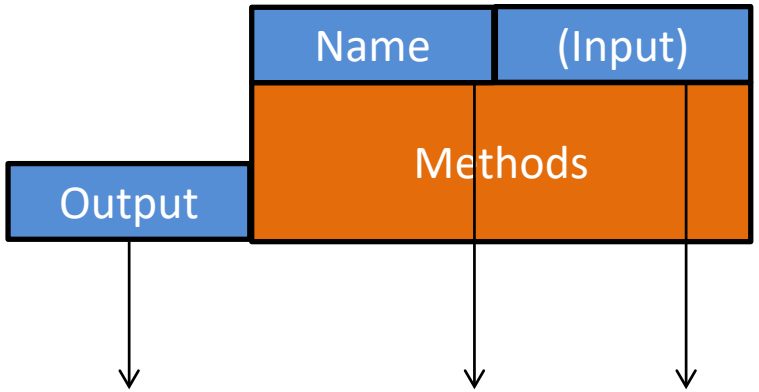
- The natural one exception: arrays

Behavior in Java: Methods

Methods in Java

Method head

```
{
  Method body
}
```



Method head → ReturnType Name([Parameter,…])

Type Name, Type Name, …

Method body → Sub program

[`return` expression];

Methods are also referred to as functions

Content of Methods

- Behavior of a program is encapsulated in methods
- Goal: Split entire behavior of program in smaller methods
  - Behavior of objects of a class are stored in methods of this class

- Possible content of methods:
  - Expressions: Computations/Operations/Assignments via data types
  - Creating (i.e., instantiating) an object: Construction of new objects
  - Method calls: Activating and executing a method

Expressions and Operators in Java

Expressions

- Constants and variables are combined with operators and assigned
- There are different kind of operators:
  - Arithmetic to calculate with numbers
  - Logical to calculate with boolean values
  - Assignments, to assign values to variables or constants

Arithmetic Operators

# Operators	Operation	Example
unary	+ -	int i = -5; int r = +5;

…	++ -- (at the same time computation and assignment)	int i = 5; i++; // i == 6
		++i; // i == 7
		--i; // i == 6
		i--; // i == 5

…	~	int i = 1; i = ~i + 1; // ~ is one‘s complement: i == -1
binary	* / % + –	Arithmetic Operations (% == modulo → remainder of a division)
…	<< >> >>>	int k = 2 << 3; // k == 16

	& ^ |	int k = 2 & 1; // k == 0
		k = 2|1; // k == 3

Arithmetic Operators are left associative:
1 / 2 / 4 == (1 / 2) / 4 == 0.125 but 1 / (2 / 4) == 2

Logical Operators

# Operators	Operations	Example
unary	! (NOT)	boolean a = false; a = !a; // a == true
binary	& (AND)	boolean a = true & false; // a == false
…	^ (XOR)	boolean a = true ^ true; // a == false
…	| (OR)	boolean a = true | false; // a == true
…	&& ||	Lazy Evaluation: In case the first argument is sufficient for evaluation, the second argument is not evaluated. Useful, when: (x != 0) && (10 / x > 1) If the enitre expression would be evaluated and x was 0, division by zero! → error!

Logical operators are also follow left assocization:

false & false || true == (false & false) || true == false || true == true

but false & (false || true) == false & true == false

Overview: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html

Relational and assignment operators

- Assignment operators (right associative)
  - = += -= *= /= %= >>= <<= >>>= &= ^= |=
  - var x [operation]= y short for x = x [operation] y
- Relational operators
  - < > >= <= == !=
  - Result is always boolean
- Conditional expression/assignment
- if [logical expression] ? [true case] : [false case]
  - int age = 20;
    String msg = age >= 18 ? „Access granted“ : „You are not old enough!";

Examples

```
byte a = 28, b = 100;
byte c = a + b;
//c = -128
int d = a + b;
//d = 128;

int e = d<<1;
// e = 256
e >>= 1;
// e = 128

e *= -1;
// e = -128

e >>= 1;
// e = -64
e >>>= 1;
// e = 2147483616
```

```
byte f = 010;
// f = 8

byte g = 15;
g = (byte) (f|g);
// g = 15

f = (byte) (f&g);
// f = 8
f++;
// f = 9
```

Comments on Behavior

- Increment is assignment at the same time

```
int num1 = 2;
int num2 = ++num1; //num1 == num2 == 3; num = num1 + 1;
//num2 = [Operation] num1; short for:
           num1 = num1 [Operation] 1;
           num2 = num1;

num1 = 2;
num2 = ++num1 * 3; // ?

//num2 = num1[Operation];
           short for num2 = num1;
           num1 = num1 [Operation] 1;
num1 = 2;
num2 = num1++ * 3; // ?
```

- Operators are overloaded, i.e., have more than one meaning

```
String s = "Hi" + " there!";
//s == "Hi there!"
```

Binding Priorities (What is Evaluated First?)

- Arithmetic before Comparison before logic before assignment
  - `boolean b = x + 1 < 10 && x >= 5;`
  - `b = (((x + 1) < 10) && (x >= 5)); // false with x = 2`
- Unary before Multiplication/Division before Addition/Subtraction
  - `int r  = -1 * ++x + 3;`
  - `r = ((-1) * (++x)) + 3;`
  - Hierarchy of priorties with 13 levels!
- Setting braces changes priorities
  - `int k = (2 + 4) * 5; // = 30 != 22`
  - Could increase readability!

Quiz!!!

- Arithmetic expressions: right or left associative?

- Which of the following signatures are correct?

```
void int compute(int a, int b) {…}
double increment(int) {…}
void delay(double time) {…}
boolean isSaturday(today) {…}
char charForNumber(int nb,) {…}
```

- What are the results of the following computations?

```
int a = 7;
int b = 2;
float c = 7/2;

c = ?
```

```
int t = -1;
int r = -1 * t++ + 3;

r = ? t = ?
```

```
int k = -1;
int l = -1 * ++k + 3;

l = ? k = ?
```

Learning Goals:

- Data types can be primitive or complex
  - Complex types conist of other types
  - Primitive types … you have to memorize them☺
- Methods allow encapsulating several statements behind one dedicated name
- Methods contain expressions to assign, change, or read values of variables and do computations

Coming Up Next

- Control structures: How can I control how my program executes?
  - If-then-else statements
  - Switch-Case
  - Go to statements
  - Loops

Coming Up Next Time

- Behavior of objects (not classes!)
- What are methods?
- How do I create objects of classes?
  - With the constructor!