# Object-Oriented Programming In Mechatronic Systems

## Summer School 2018

**Module 1 – Introduction to Programming**
Aachen, Germany

Cybernetics Lab IMA & IfU
Faculty of Mechanical Engineering
RWTH Aachen University

# Organization

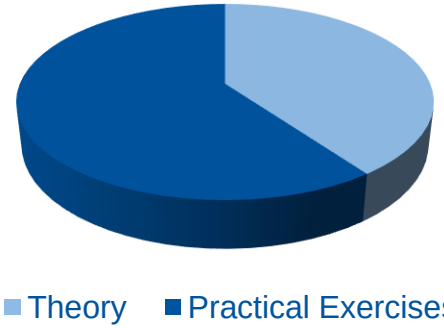
# Organization

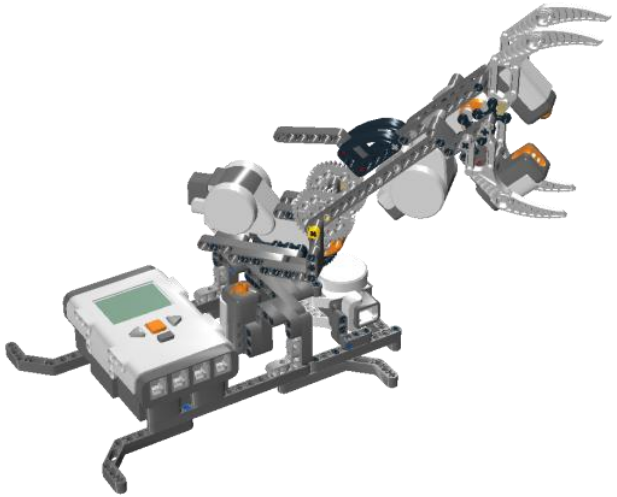
## Synopsis

*Today's mechanical engineering relies heavily on advanced software tools. Both industry and research expect you not only to use these tools but to design, develop and deploy them as well. During this course we teach you how.*

## Topics

- Java 101
- Object Oriented Software Engineering
- Software-Hardware Interaction



# Organization

## … at the Institute of Information Management in Mechanical Engineer (IMA)

### Information Management for Mechanical Engineering

Prof. Dr.-Ing.
**Tobias Meisen**

Management Director
of IMA

**Andreas Kirmse**
M.Sc.

Researcher
Industrial Big Data

**Alexander Paulus**
M.Sc.

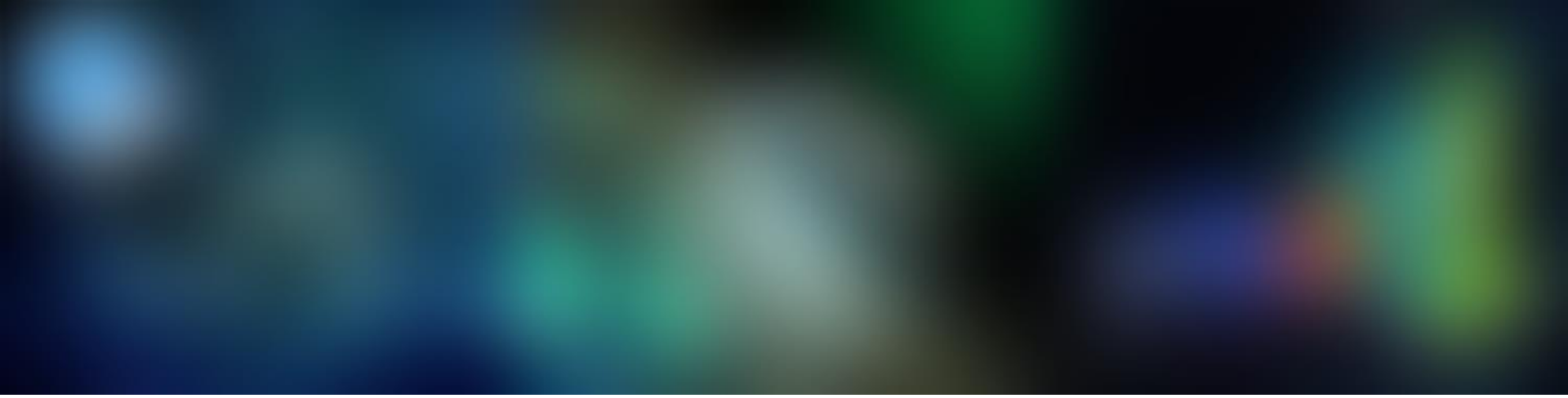
Researcher
Cognitive Computing

Dr.-Ing.
**Max Hoffmann**

Group Leader
Industrial Big Data

Dipl.-Inform.
**Daniel Lütticke**

Group Leader
Production Technology



# The Cybernetics Lab

# Who are we?

# Interdisciplinary at the Cybernetics Lab IMA & IfU

Univ.-Prof. Dr.-Ing.
**Christian Hopmann (IKV)**
Acting Head of Institute

apl.-Prof. Dr. habil.
**Ingrid Isenhardt**
Deputy Head of Institute

Dr. rer. nat.
**Frank Hees**
Vice Deputy Head of Institute

apl.-Prof. Dr. rer. nat.
**Sabina Jeschke**

**Administration** | **Public Relations** | **IT & Media Technology**

## IMA
**Inst. of Information Management in Mechanical Engineering**

Information Management | Knowledge Management

Jun.-Prof. Dr.-Ing.
**Tobias Meisen**
Managing Director

Dr. phil.
**Max Haberstroh**
Managing Director

| | |
|---|---|
| Ing. Industrial<br>**Alexia Fenollar Solvay**<br>Mobility and Logistics | Dr. rer. nat.<br>**Stefan Schröder**<br>Innovation- & Work Science |
| Dipl.-Inform.<br>**Daniel Lütticke**<br>Production Technology | Dr. phil.<br>**Kathrin Schönefeld**<br>Knowledge Engineering |
| Dipl.-Inform.<br>**Christian Kohlschein**<br>Cognitive Computing & eHealth | Dr. phil.<br>**Valerie Stehling**<br>Digital Learning Environments |
| Dr.-Ing.<br>**Max Hoffmann** M.B.A.<br>Industrial Big Data | |

## IfU
**Associated Institute for Management Cybernetics**

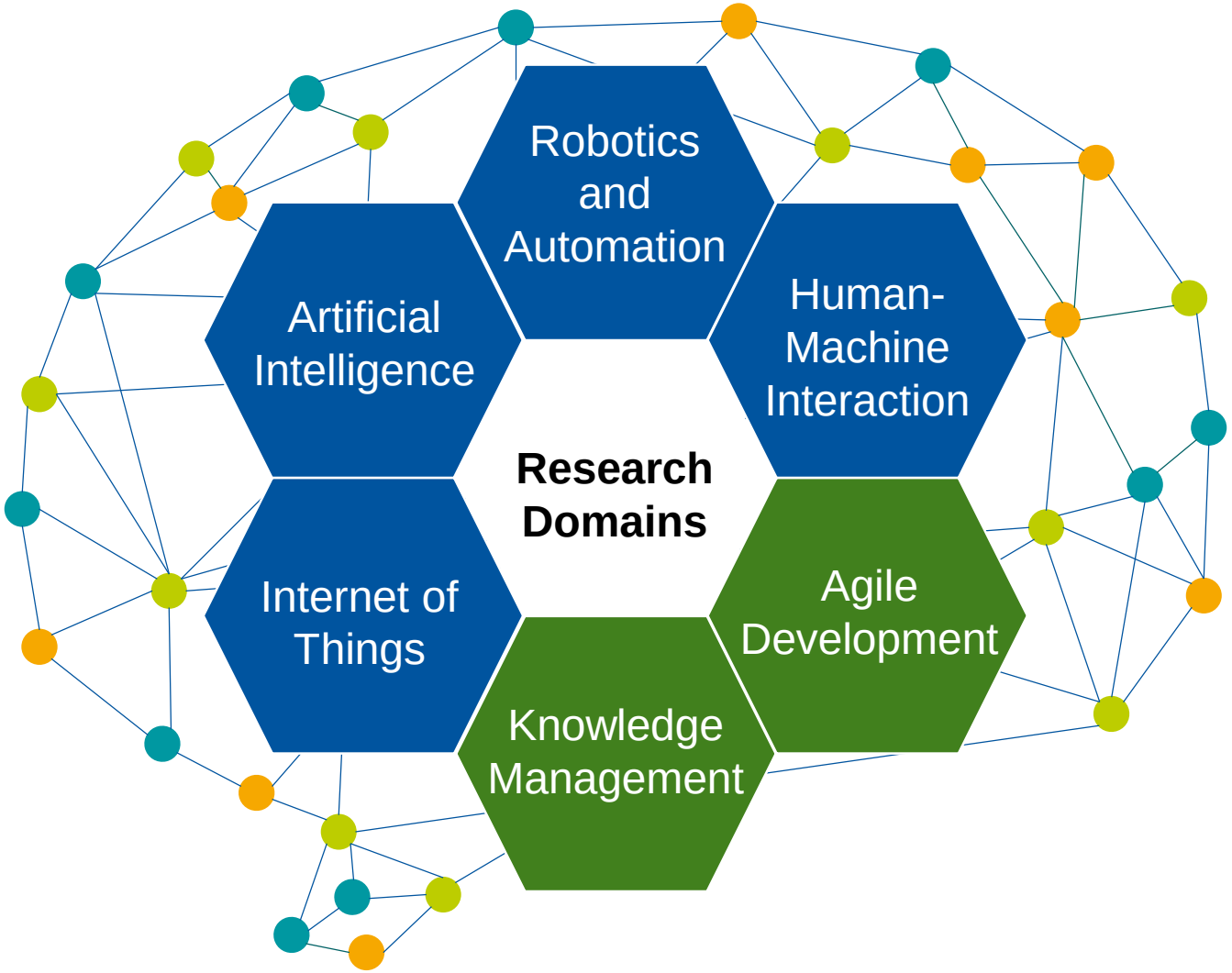
Dr. rer. nat.
**René Vossen**
Managing Director

| |
|---|
| Dr. phil.<br>**Daniela Janßen**<br>Economic and Social Cybernetics |
| Dr. rer. nat.<br>**Pia Bresenitz**<br>Technical Cybernetics |

Prof. Dr.-Ing. em.
**Klaus Henning**
Senior Advisor

What drives us

# Research Domains

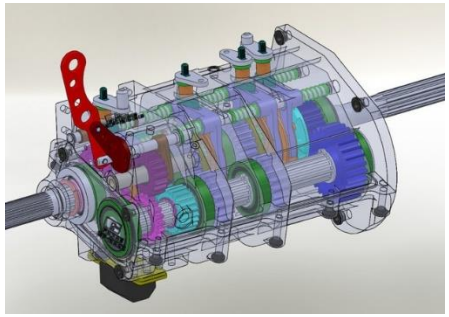


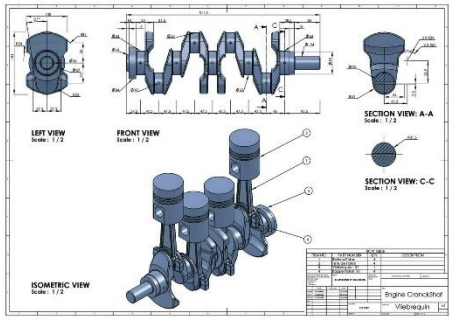
# Motivation

# Motivation

**Mechatronic Systems rely on Advanced Software Tools!**

## From Computer Aided Design (CAD) to Robotics …

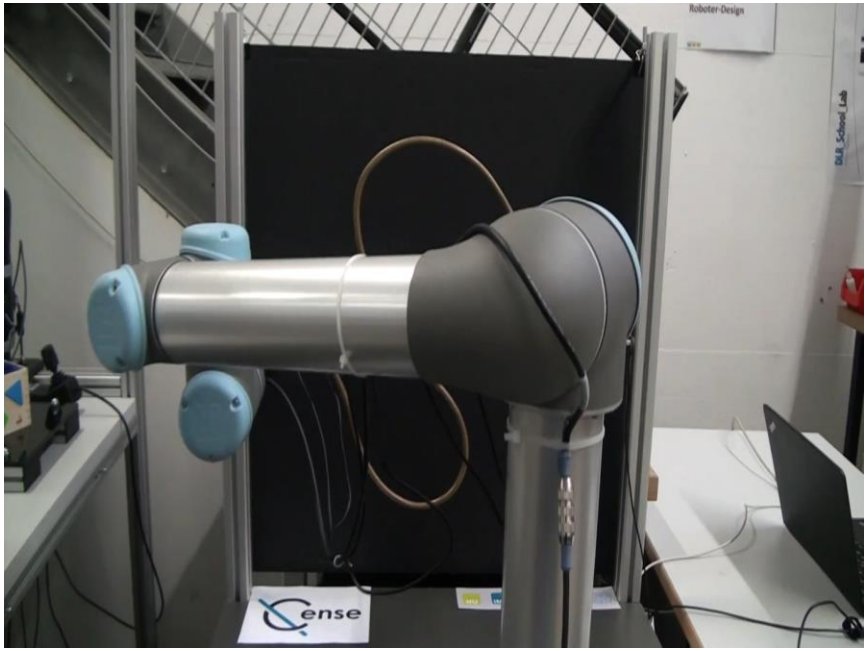





# Motivation

**Mechatronic Systems rely on Advanced Software Tools!**

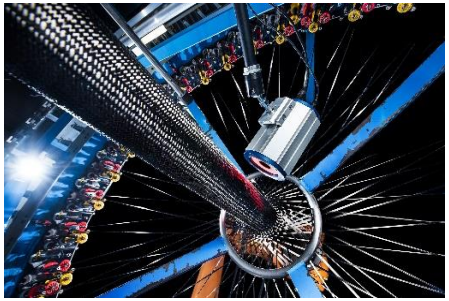
## … to learning robots! (at our institute)

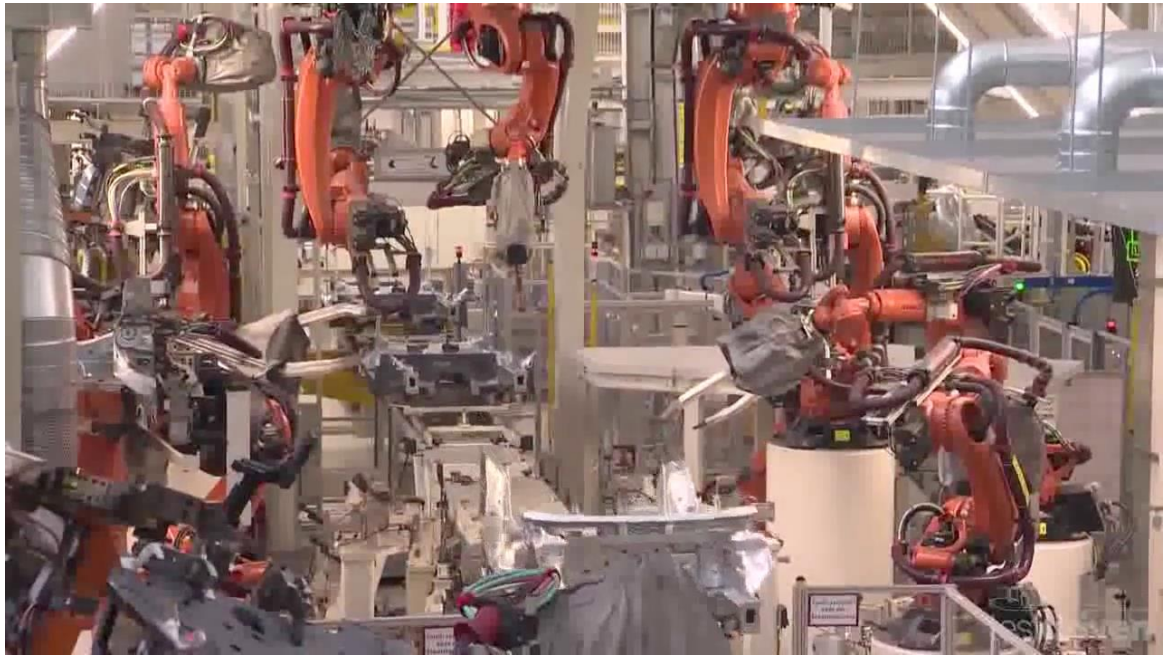


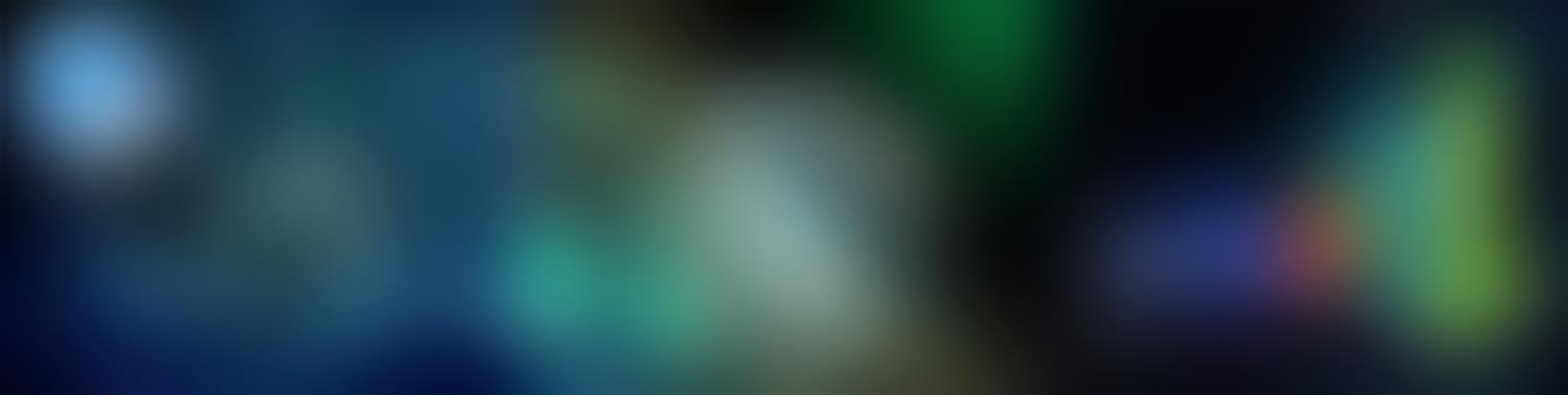
# Motivation

**Mechatronic Systems rely on Advanced Software Tools!**

## … to self-optimizing production systems!



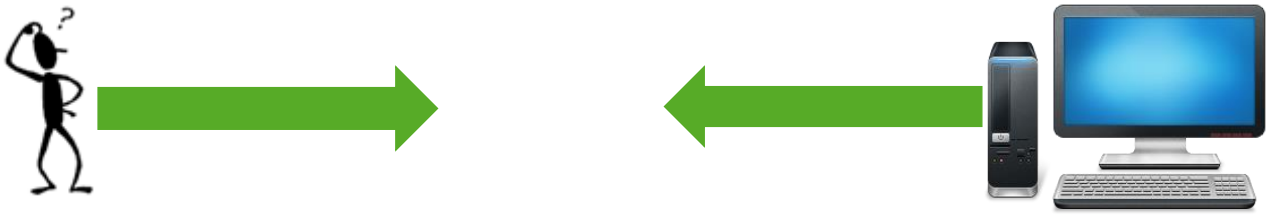




# Algorithms and Programming Languages

# Algorithms and Programming Languages

## We need an interface between human and computer



## Both have different requirements:

**Human:**

- Analog world
- Visual, haptic and auditory signals
- Comprehensive integration in contextual knowledge
- Fluent, “natural” language

**Computer**

- Digital World
- Electronic signals
- Majority: no information “outside”
- Structured statements: Algorithms

## How do we formulate a problem for the computer?

An **algorithm** is an unambiguous rule of action for solving a problem or a class of problems.

**Colloquially**:

- Algorithms are "somehow clever" methods that efficiently help to solve specific problems
- Not only arithmetic problems such as efficient addition or multiplication, but also everyday questions:
  - How do I find the exit from a labyrinth?
  - How do I calculate the shortest connection between two cities?
  - How do I search my warehouse shelf as quickly as possible?

## Example of an algorithm:

1. Put a filter in the filter container
2. Fill the filter with coffee powder
3. Pour water into the tank provided for this purpose
4. Check whether empty coffee pot is ready
5. If **yes**: Go to step 7 ← Branch
6. If **not**: empty the coffee pot and place it under the filter
7. Press the start button
8. Wait until the coffee is ready (typically: machine "gurgles", steam rises)

← Termination Condition

## Properties of algorithms:

**Finiteness**:

- Formulated in a finite text (static finiteness)
- Finally needs a lot of memory (dynamic finiteness)
- Finished in finally many steps (scheduling)

**Executability**:

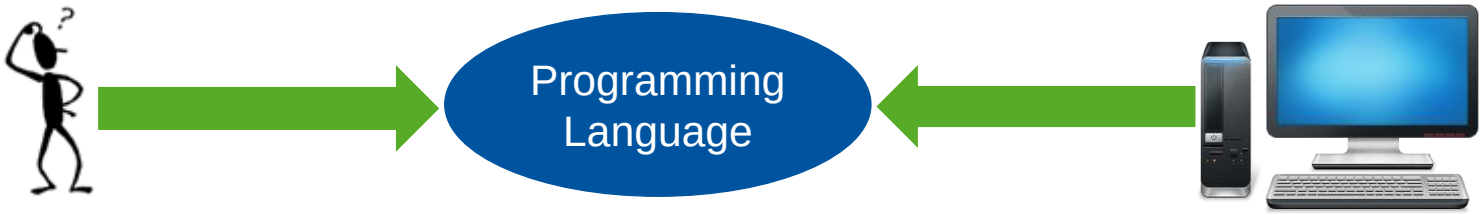
- Each step must actually be executable

**Uniqueness**:

- Always the same result under the same conditions (Determinacy)
- Only ever exactly one possibility of continuation (Determinism)

# Algorithms and Programming Languages

## Interface between Human and Computer




**Still, both have different requirements:**

**Human:**

- Natural language
- Legibility
- Expressiveness

**Computer**

- Simple translation into machine code
- Efficiency of the generated code

# Algorithms and Programming Languages

## Learning programming languages comparable to "natural" foreign languages

**Syntax**:

- Defines permitted strings (= vocabulary) and grammar
- In each language there are defined keywords

**Semantics:**

- Defines the meaning of the syntax
- Builds on syntax


**Syntactically correct, semantic nonsense:**
"A banana speculates purple the sunset."

**Syntactically incorrect, semantically correct:**
"A banana is fruit yellow."

**Syntactically correct, semantically correct:**
"A banana is a yellow fruit."

# Algorithms and Programming Languages

```java
1. class HelloWorld{
2.   public static void main(String[] args){
3.     System.out.println("Hello World!");
4.   }
5. }
```

**Java**

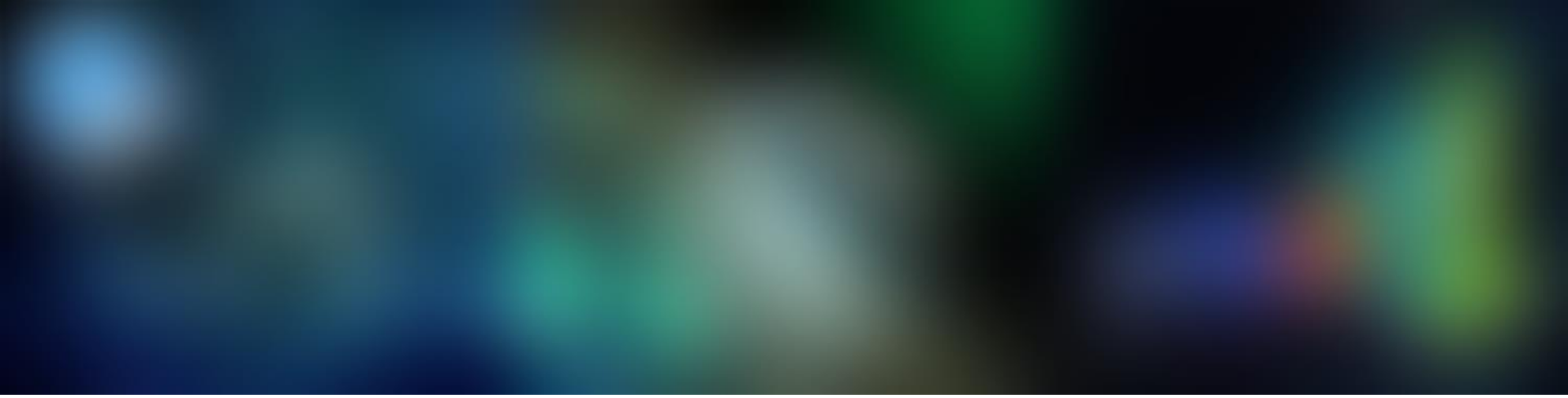
```fortran
1.Program Hello
2.Print *, "Hello World!"
3.End Program Hello
```

**Fortran**

```python
1.class HelloWorld(object):
2.    def __init__(self, args):
3.          print(„Hello World!“)
```

**Python**

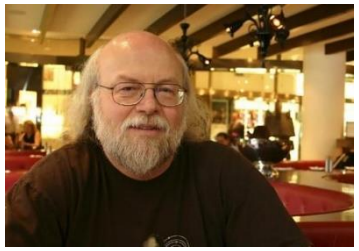
**Different syntax, identical semantics!**



# The Java Programming Language

## Brief History

- Java invented June 1991 by James Gosling at Sun (2010 acquired by Oracle)

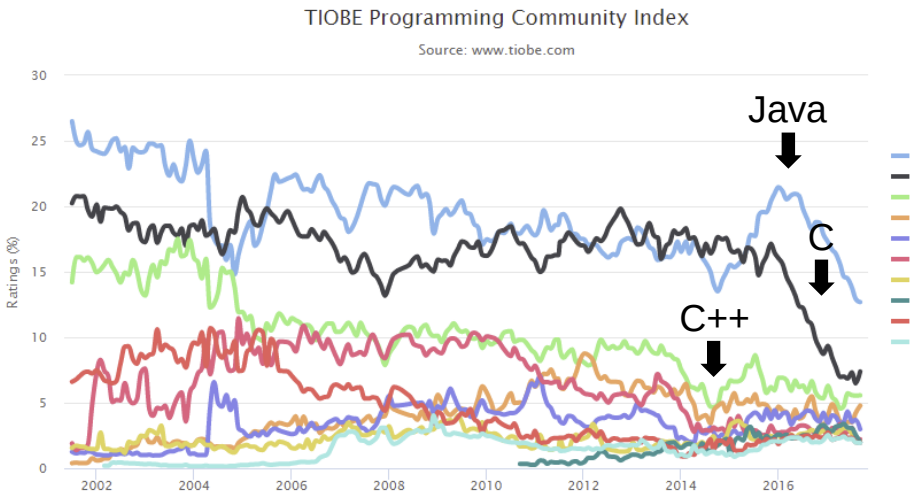


- **Five** Design Goals:
  - *“Simple, Object Oriented, and Familiar”*
  - *“Robust and Secure”*
  - *“Architecture Neutral and Portable”*
  - *“High Performance”*
  - *“Interpreted, Threaded, and Dynamic”*

# The Java Programming Language

## It’s widely spread!

### TIOBE 2015 (Popularity Index)




| Sep 2017 | Sep 2016 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 12.687% | -5.55% |
| 2 | 2 | | C | 7.382% | -3.57% |
| 3 | 3 | | C++ | 5.565% | -1.09% |
| 4 | 4 | | C# | 4.779% | -0.71% |
| 5 | 5 | | Python | 2.983% | -1.32% |

### Industry use (to name a few)

# Structure of a Java Program

## Structure of a Java Source File

Class Definition

```
public class Foo {




}
```

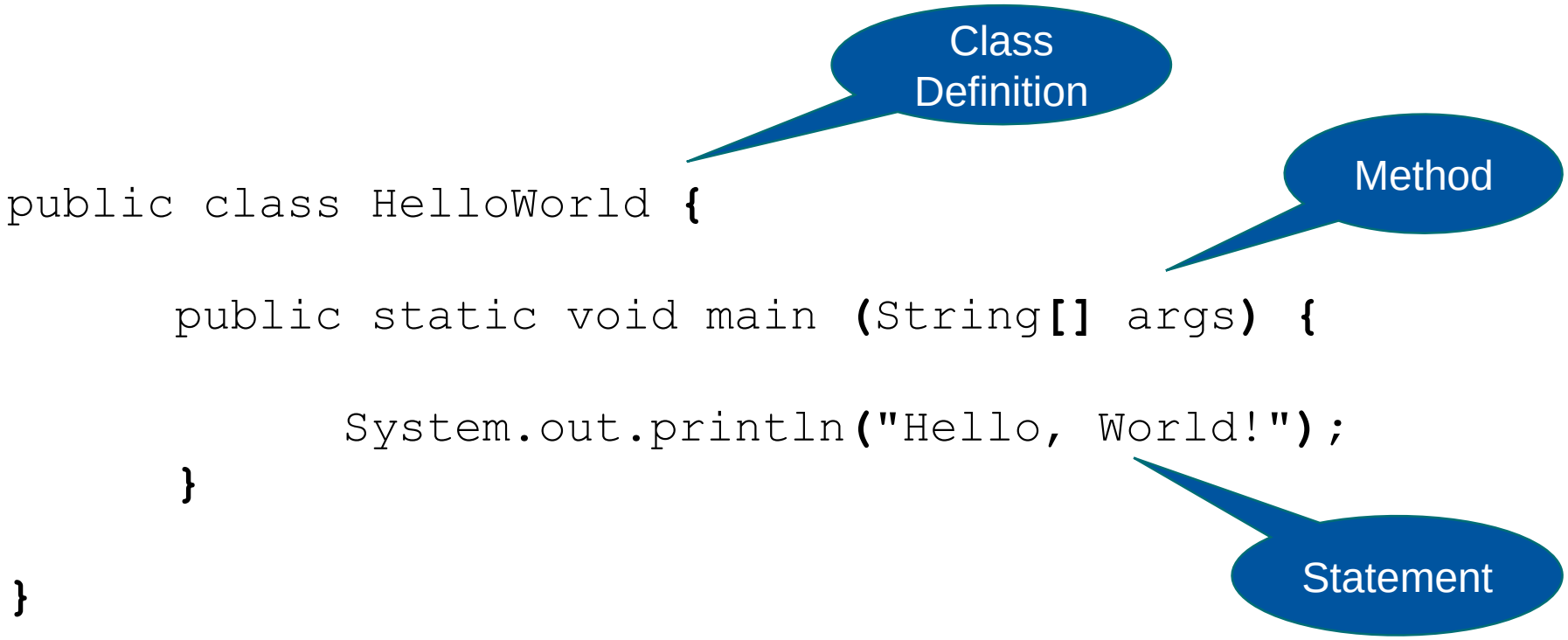
Methods

```
public class Foo {

    void bar() {




    }

}
```

Statements

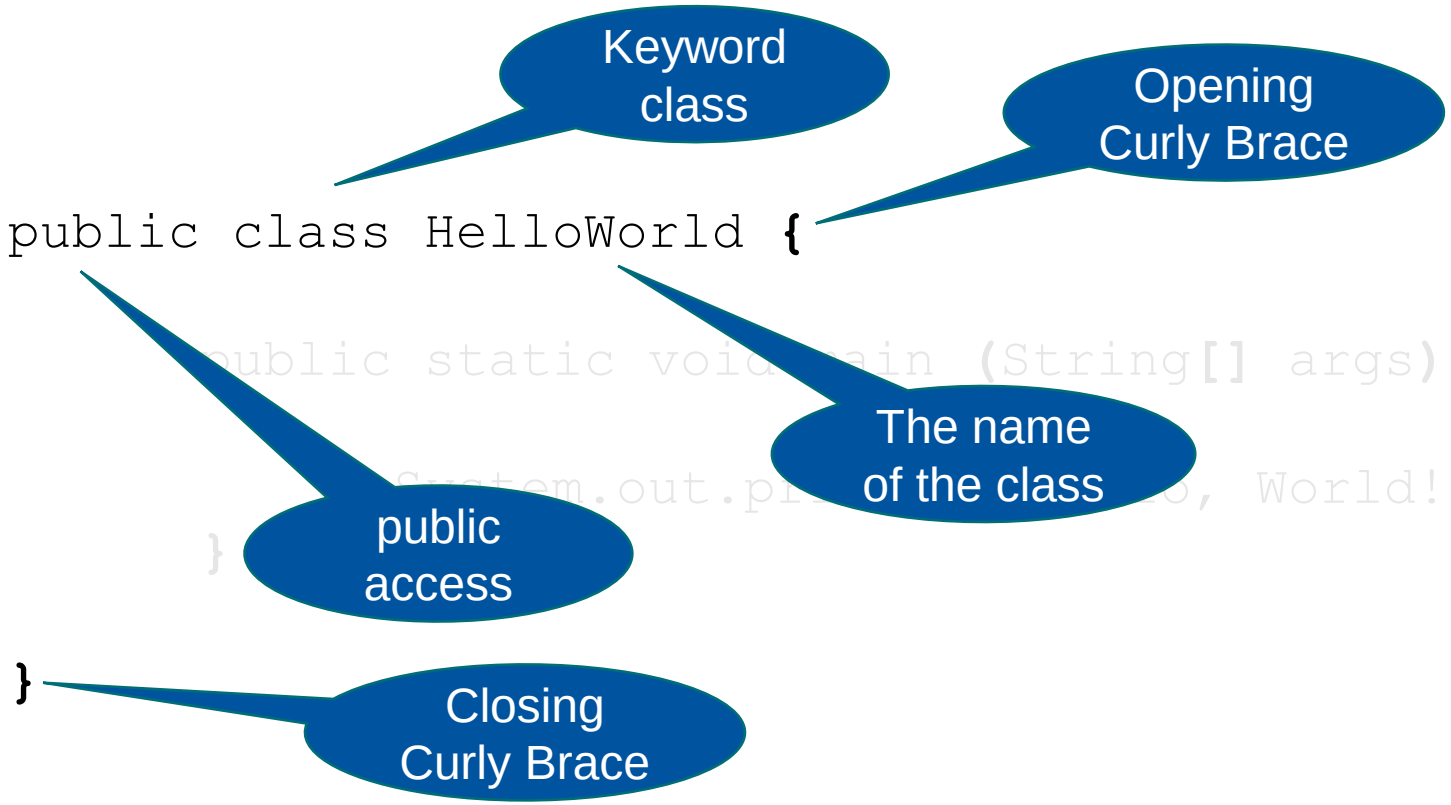
```
public class Foo {

    void bar() {

        statement1;

        statement2;

        statement3;
    }

}
```

## Structure of a Java Source File

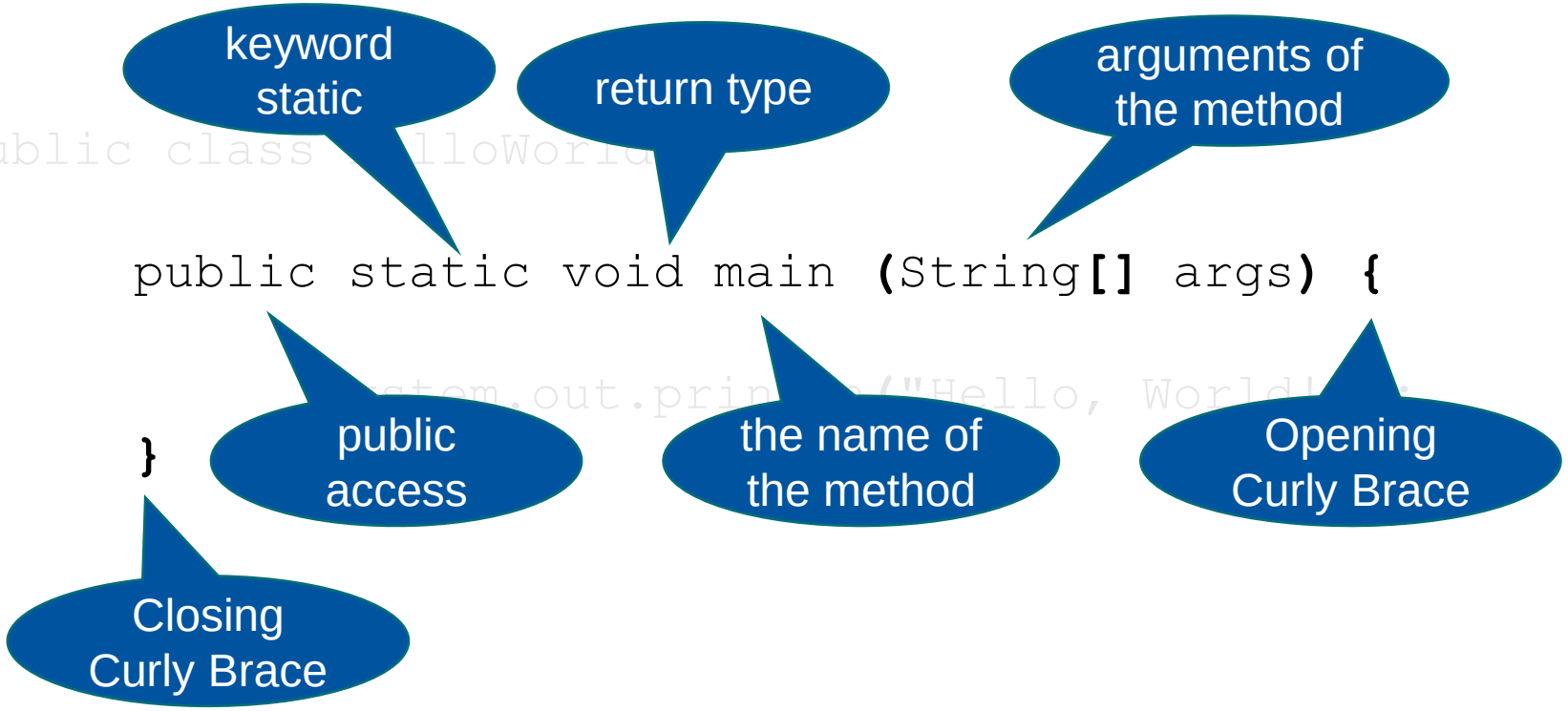



```
public class HelloWorld {

      public static void main (String[] args) {

            System.out.println("Hello, World!");
      }

}
```

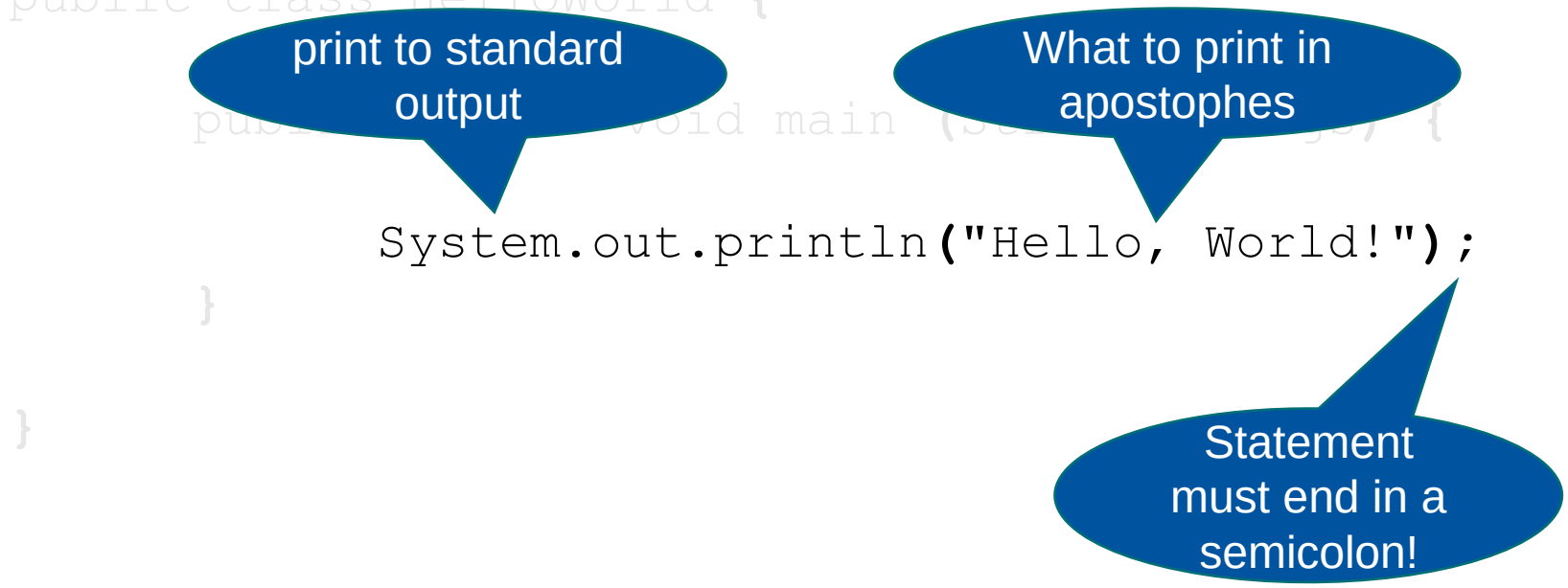
## Structure of a Java Source File. A closer look at the class.



## Structure of a Java Source File. A closer look at the method.



## Structure of a Java Source File. A closer look at the statement.



# Structure of a Java Program

## What are comments?

- Document the code and keep it readable
- Single line comment: `// myComment`
- Multiple line comment: `/* myMultiLineComment */`

## Examples

```
public class HelloWorld { // It's my first class!
  public static void main (String[] args) {
    /* I want to
    print on the command line */
    System.out.println("Hello, World!");
  }
}
```

# Variables

## What are variables?

- A container, a box or a cup. It contains something.
- They come in different kinds
- They got a name

## Examples

- `short numberOfEngines = 5;`
- `double temperature = 23.7;`
- `boolean engineStarted = true;`
- `char c = 'e';`
- `int depth = -343535;`

## Two ways of “constructing” variables

- **First**, declare, than initialize: `int length; length = 5;`
- **Second**, define them in one single statement: `int length = 5;`

## Examples

- `short numberOfEngines = 5;`
- `double temperature = 23.7;`
- `boolean engineStarted = true;`
- `char c = 'e';`
- `int depth = -343535;`

## Four Primitive Data Types in Java

- boolean, char, integer and floating point
- They got a default value
- They only hold one value

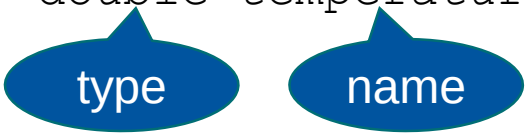
| Data Type | Example | Keyword |
|---|---|---|
| Logical value | true, false | `boolean` |
| Single character | a, b, … | `char` |
| Whole number | 1, -3, 87, … | `byte, short, int, long` |
| Real number | -2.6, 9.4, … | `float, double` |

For details (e.g. max or min values) see:
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

## Rules I

- Variables must have a **type**, e.g. `double`!
- Variables must have a **name**, e.g. `temperature`!

```
double temperature;
```

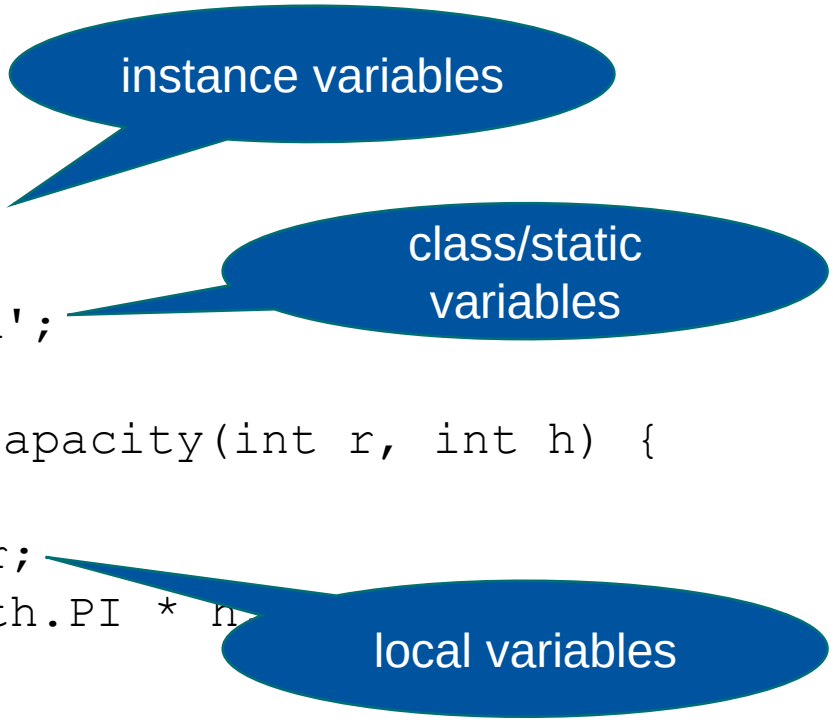



## Rules II and Good Practice

- No keywords are allowed as names, e.g. `class` or `while` are prohibited!
  https://en.wikipedia.org/wiki/List_of_Java_keywords
- Names must start with a letter, underscore (_) or a dollar sign ($)
- No special characters, e.g. `§`.
- **Choose meaningful names**, e.g. `currentVelocity` (as opposed to `cV`);

## Three kinds of variables in Java

```
public class Cylinder {

    public double cylinderCap = 0;

    public static char vendor = 'A';

    public double computeCylinderCapacity(int r, int h) {

            double rSquare = r*r;
            return rSquare * Math.PI * h

    }
}
```



## Defining Constants Variables

- Are all-round in Mathematics, physics, engineering ...
- Are declared with the keyword `final`
- Convention for naming of constants: UPPERCASE, e.g. PI or E

## Examples (the bad and the good)

```
double circumf = 2 * 3.1415 * r;
double area = r * r * 3.1415;
```

- Typing errors
- Changing code in different places
- Bad Readability

```
final double PI = 3.1415;
double circumf = 2 * PI * r;
double area = r * r * PI;
```

- Good readability
- DRY principle (don’t repeat yourself)

## Operators and Variables

- **Allocation (=)**
- **Arithmetic (+, -, *, /, %)**
- Comparison (==, !=, <, >)
- Unary (++, --)
- Logical (! (not), &&, ||)

## Allocation and Arithmetic Operator Examples

- ```
  int number; number = 5;
  ```
- ```
  int x = 5;
  int y = 7;
  int sum = x + y;
  int diff = 40 - y;
  double div = 30 / 4.3;
  ```

## Operators and Variables

- Allocation (=)
- Arithmetic (+, -, *, /, %)
- **Comparison (==, !=, <, >)**
- **Unary (++, --)**
- **Logical (! (not), &&, ||)**

## Comparison, Unary and Logical Operator Examples

- ```
  boolean isSmaller;
  int one = 1; int two = 2;
  isSmaller = one < two;
  ```
- ```
  int i = 3;
  int j = i++;
  ```
- ```
  boolean result = !true || false;
  ```
  ```

## Operators Priorities

- How does Java evaluate an complex expression? E.g.:
- ```
  int a = 5;
  int b = 7;
  int c = 2
  a = a – b – c % (a * c++); // (a is –4)
  ```
- With an internal priority table! Excerpt from: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

| Priority | Operator |
|---|---|
| 1 | Unary, e.g. ++ |
| 4 | Additive, e.g. + |
| 12 | Logical OR e.g. \|\| |
| 14 | Allocation, e.g. = |
  ```

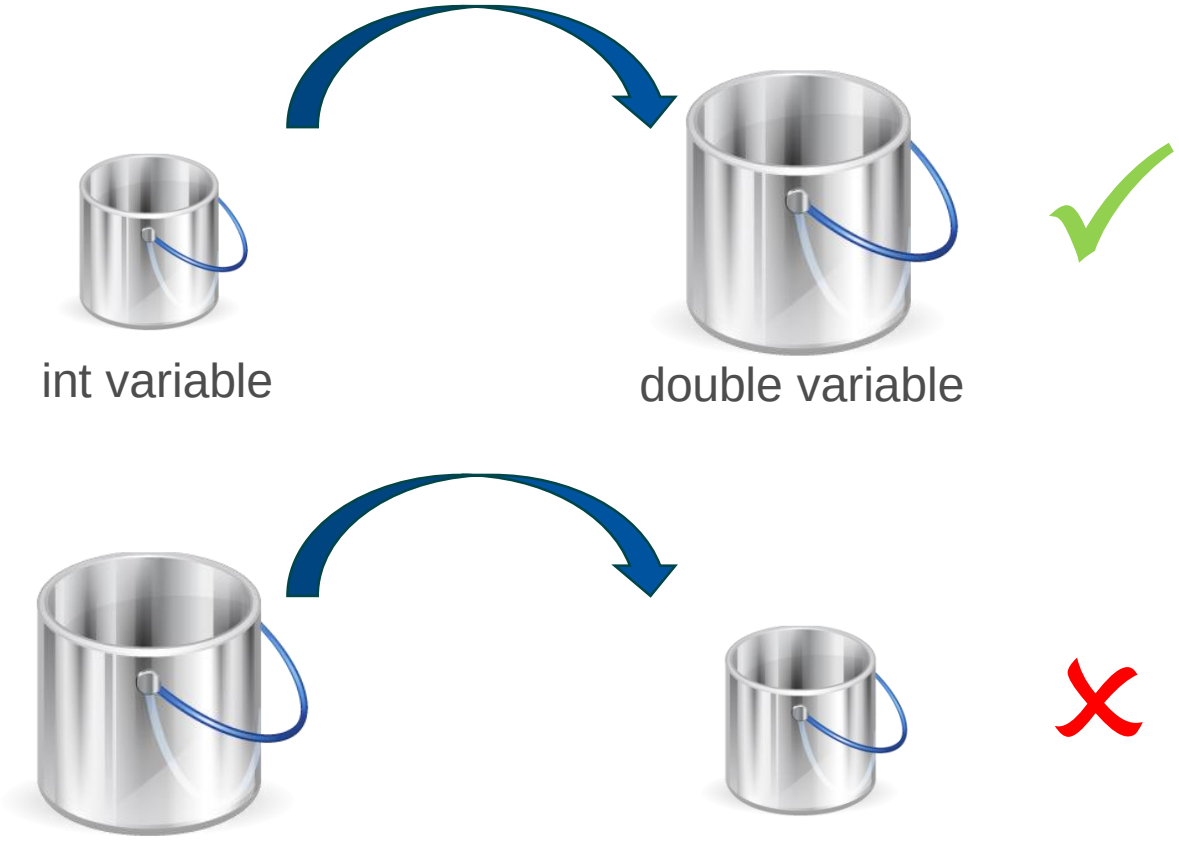
## Type Casts

- It can be necessary to convert one type of data into an other one
- There are two types of casts
- **Implicit** type casts. Target type is computed **automatically** via context. “Upgrading”.
- **Explicit** type casts. Target type **has to be explicitly defined**. “Downgrading”. Target type has to be defined in “(“ and “)” brackets

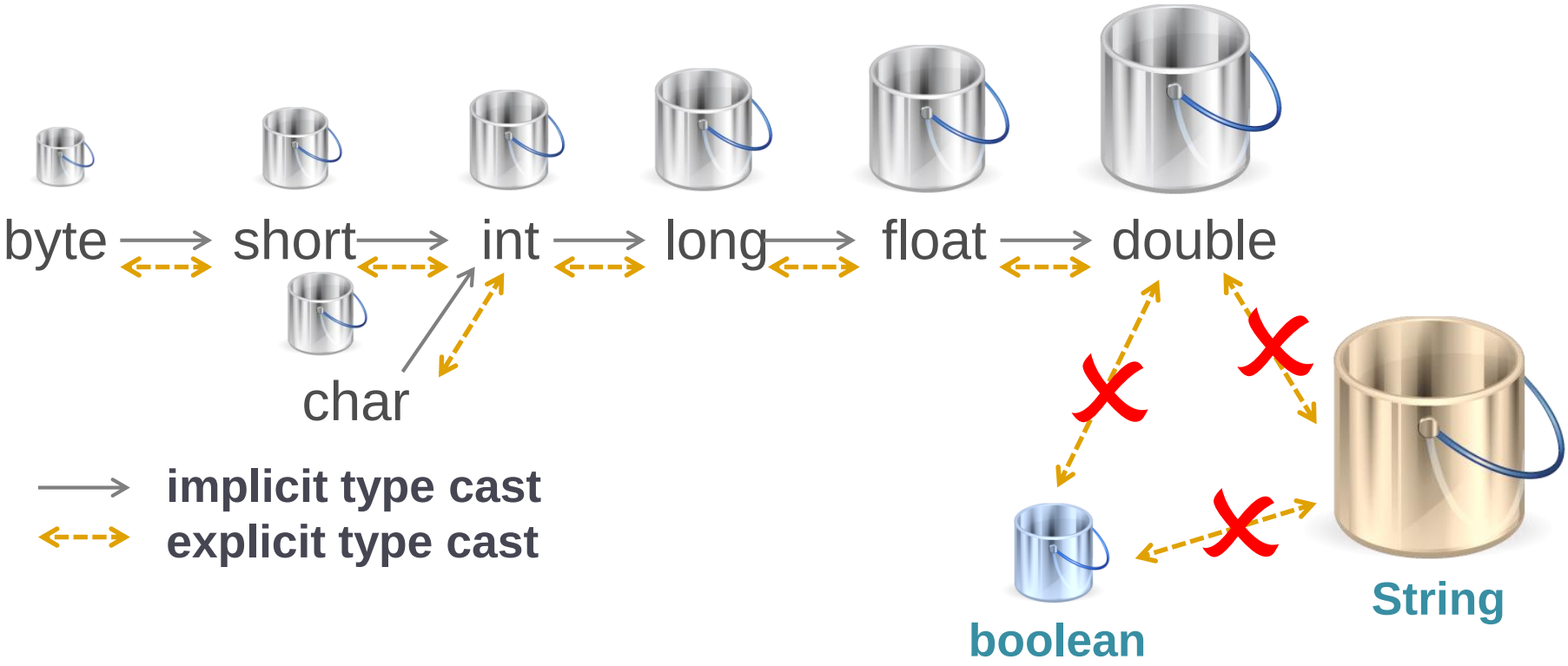
## Examples

- ```
  int i = 70;
  double radius = i; // radius contains 70.0
  ```
- ```
  double d = 70.3456;
  int num = (int)d; // num contains 70
  ```

## Implicit type cast only one way



## Type cast overview



## Output

- How does Java output information on the screen?
- Without linefeed: System.out.print (<output>)
- With linefeed: System.out.println (<output>)

## Examples

- ```
  System.out.print("Hello, World");
  System.out.print ("!"); // Output: Hello, World!
  ```
- ```
  System.out.println ("Hello, World");
  ```

# Practical Demonstration using Eclipse

## Getting started with Java


1. Install JDK 8 → 32 Bit!
2. Install Eclipse → 32 Bit!
3. Together we will do the following steps!

**<u>Listen up and exactly follow our instructions!!!</u>**


1. Open Eclipse
2. Create a new project
3. Create a new class
4. Implement a Main method
5. Implement a Variable inside the Main method
6. Implement a Variable outside the Main method
7. Print to console
8. Use the debugger


# Thank you very much!

IfU IMA RWTH AACHEN UNIVERSITY