Introduction to the course and basic programming concepts

Lecture 1 of TDA 540
Object-Oriented Programming

Jesper Cockx

Fall 2018

Chalmers University of Technology — Gothenburg University

About the course

Teaching team

- Jesper Cockx (lecturer)
- Krasimir Angelov (examinator)
- Ken Bäcklund (lab assistant)
- Gunnar Stenlund (lab assistant)
- Yazan Ghafir (lab assistant)
- Adam Jenderbo (lab assistant)

About me

- Postdoc in the Logic & Types group of the CSE department
- One of the developers of the Agda programming language
- First time teaching this course
- Moved to Göteborg from Belgium last year

Course objectives

In this course, you will learn...

... the basic concepts of computer science.

... how to solve problems by writing small programs in Java.

... how to understand and debug programs written by yourself or other people.

... how to structure programs using object-oriented programming.

Course overview

LP1: Basic programming concepts

- Primitive datatypes: `int`, `double`, `String`, ...
- Control structures: `if`, `while`, `for`, ...
- Writing your own methods
- Using single- and multi-dimensional arrays

LP2: Object-oriented programming

- Objects, classes, and interfaces
- The collections framework
- GUIs and event-driven programming
- Creating and handling exceptions

Course schedule

Course spans both LP1 and LP2.

- Lectures (in English) are on Mondays (LP1) and Tuesdays (LP2).
- Lab sessions (in Swedish) are organized 2-3 times per week.

Always check time and location on TimeEdit!

Course website

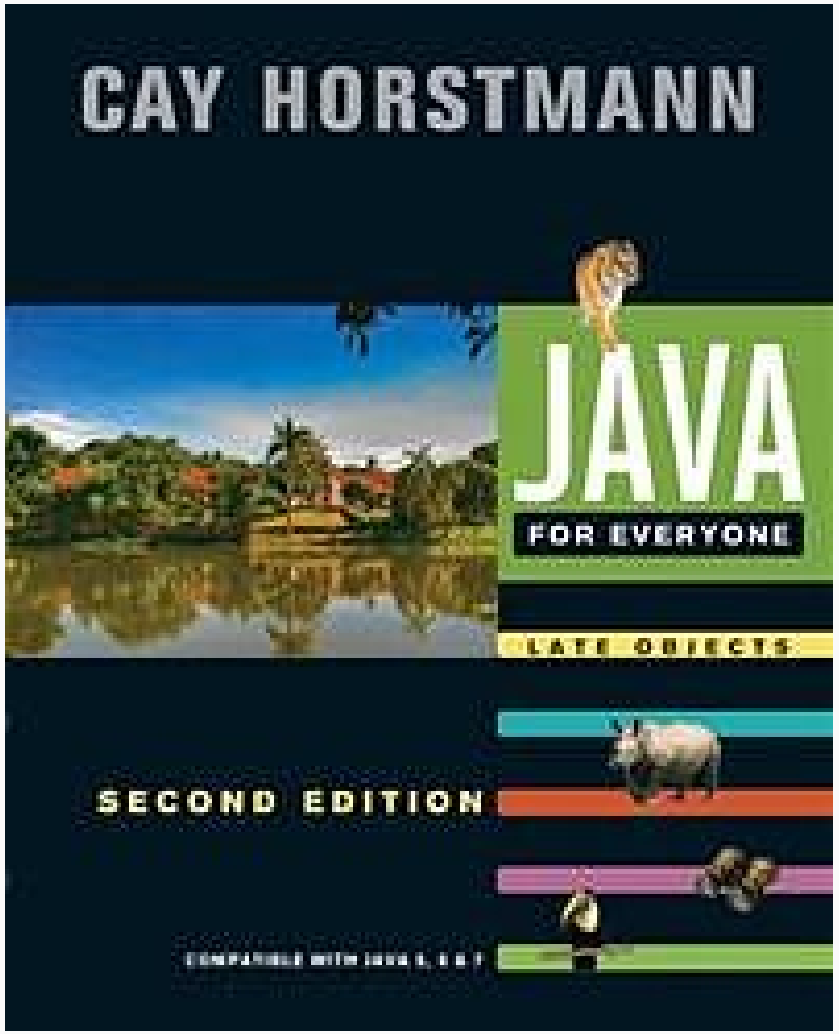
`www.cse.chalmers.se/edu/course/TDA540/`

- Latest updates
- Course schedule
- Lecture slides
- Assignments for lab sessions
- Self-study questions
- Contact information
- Links to useful resources

Course book

Horstmann:
Java For Everyone
(second edition)

Available at Cremona



Course examination

Two required components:

- 8 obligatory lab assignments (4 per block)
- Written exam in January.

Lab sessions

There are 8 *obligatory* lab assignments.

- You will work in groups of 2.
- The assignments are on the website.
- Submission of solutions via Fire.
- You can ask for help from one of the assistants via Waglys.

See website for links to Fire and Waglys.

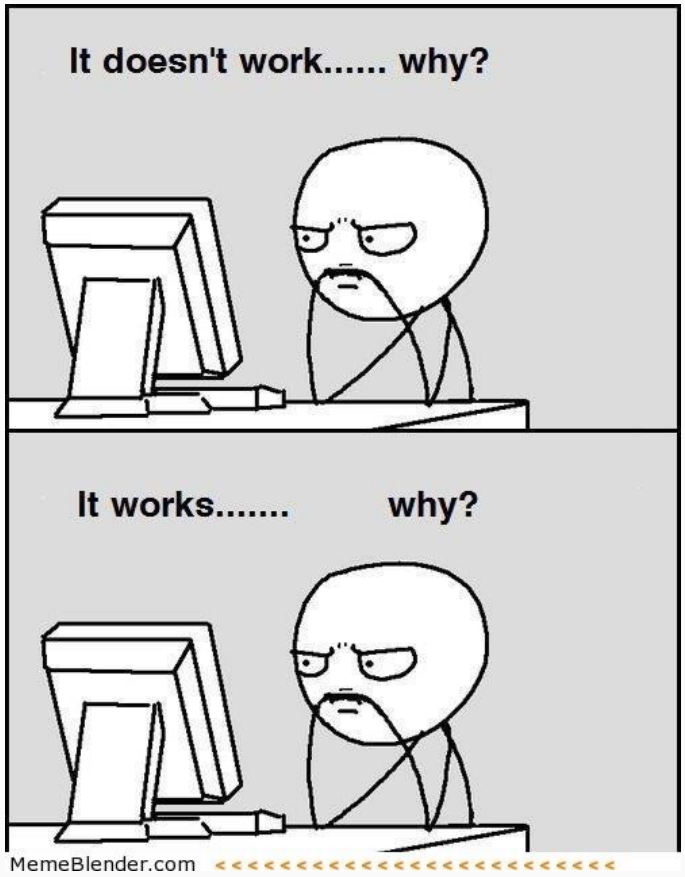
Getting help

- Fellow students
- Book, Java documentation, google, ...
- Discussion group on Google Groups
- Ask lab assistants
- Ask me before/after lecture or via email

Embrace the growth mindset

Ask questions!

Reflect!



What is programming?

What is programming?

Programming

=

telling the computer what to do

What is programming?

Programming

=

telling the computer what to do
using a programming language.

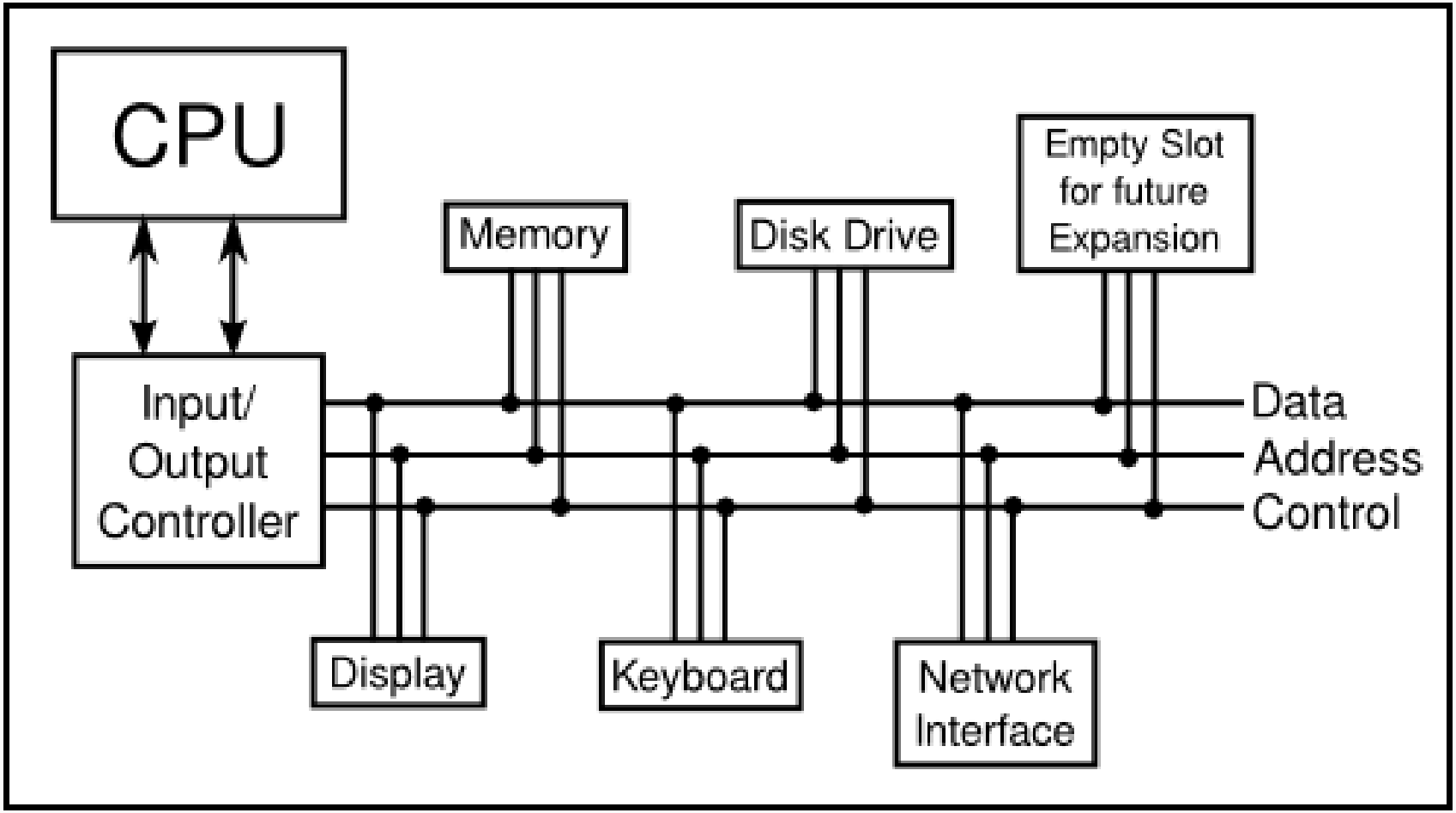
Why learn programming?

Knowing how to program is useful even if you're not a programmer:

- Computers and software are everywhere
- Understand possibilities and limitations of tools you use
- Programming = problem solving

Also: programming is fun!

How does a computer work?



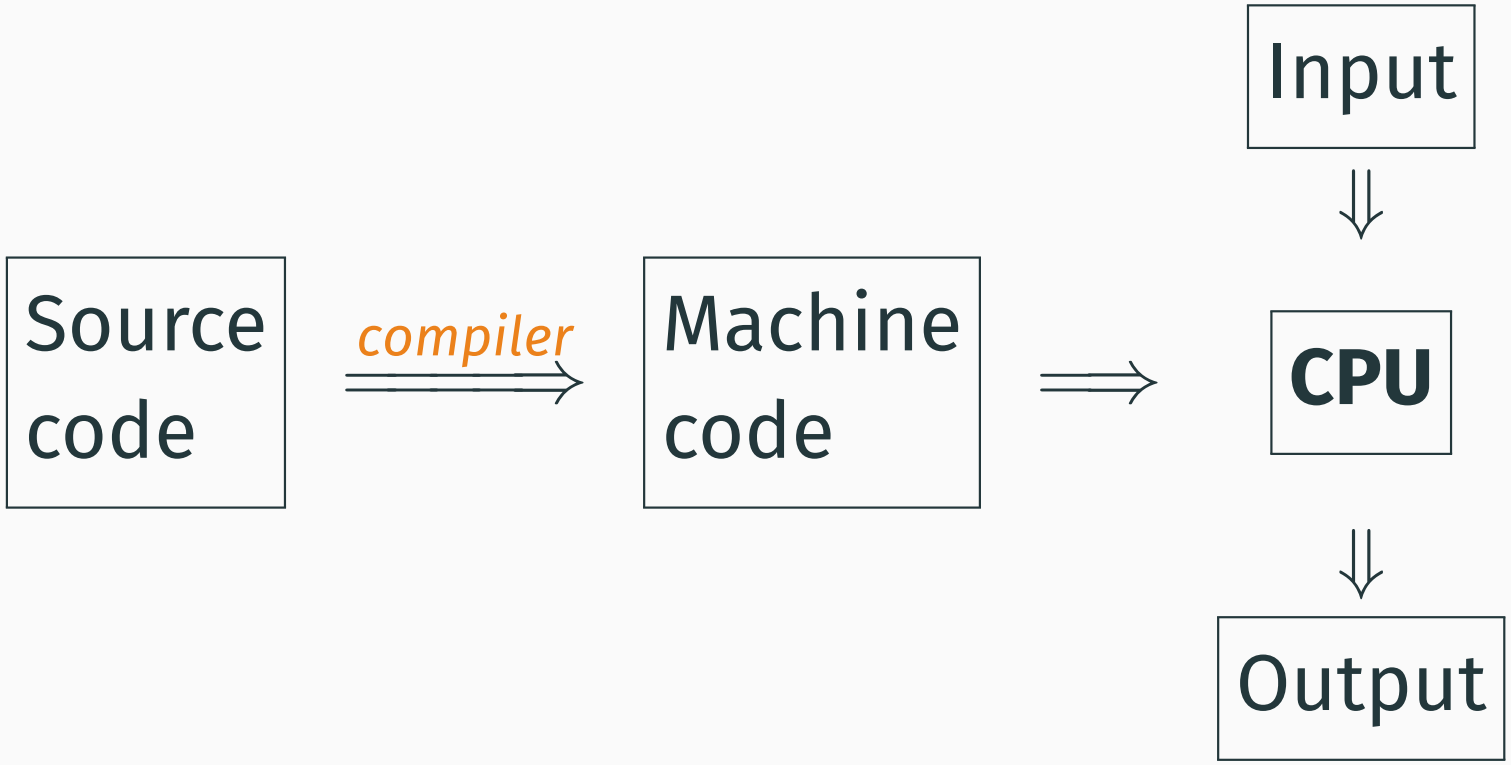
Memory is divided into code and data.

Machine code

```
FE30- 20 B4 FC 90 F7 60 B1 3C
*FDEDL

FDED-    6C 36 00     JMP    ($0036)
FDF0-    C9 A0        CMP    #$A0
FDF2-    90 02        BCC    $FDF6
FDF4-    25 32        AND    $32
FDF6-    84 35        STY    $35
FDF8-    48           PHA
FDF9-    20 78 FB     JSR    $FB78
FDFC-    68           PLA
FDFD-    A4 35        LDY    $35
FDFF-    60           RTS
FE00-    C6 34        DEC    $34
FE02-    F0 9F        BEQ    $FDA3
FE04-    CA           DEX
FE05-    D0 16        BNE    $FE1D
FE07-    C9 BA        CMP    #$BA
FE09-    D0 BB        BNE    $FDC6
FE0B-    85 31        STA    $31
FE0D-    A5 3E        LDA    $3E
FE0F-    91 40        STA    ($40),Y
FE11-    E6 40        INC    $40
*
```

Role of the compiler



Programming languages vs. natural languages

Why not use a natural language?

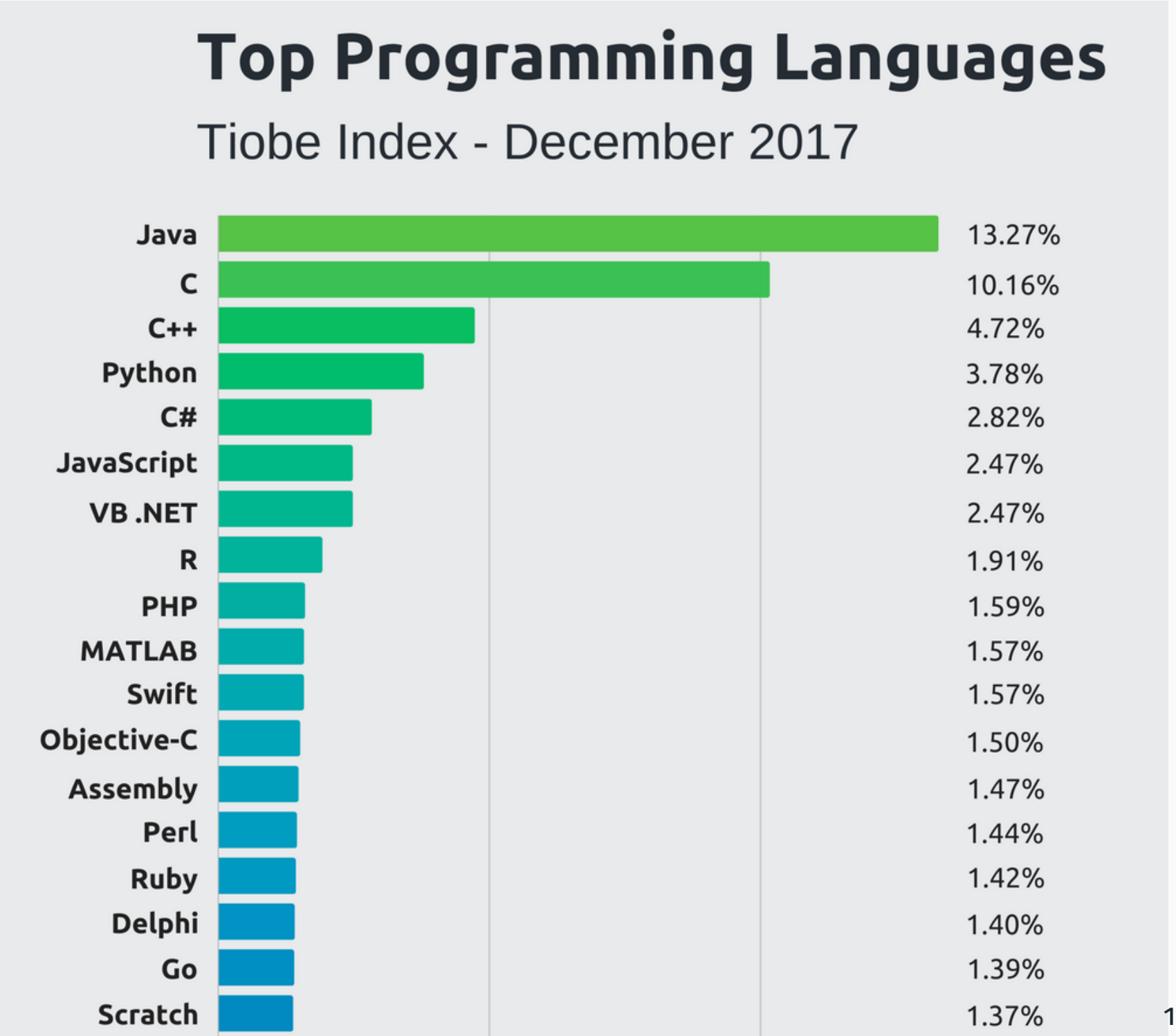
- Complexity
- Ambiguity
- Not strictly defined

A programming language *forces* you to give precise instructions that a computer can understand!

Programming languages and you

Have you programmed before?

What programming language did you use?


Top Programming Languages
Tiobe Index - December 2017
Java 13.27%
C 10.16%
C++ 4.72%
Python 3.78%
C# 2.82%
JavaScript 2.47%
VB .NET 2.47%
R 1.91%
PHP 1.59%
MATLAB 1.57%
Swift 1.57%
Objective-C 1.50%
Assembly 1.47%
Perl 1.44%
Ruby 1.42%
Delphi 1.40%
Go 1.39%
Scratch 1.37%

Programming paradigms

	Statically typed	Dynamically typed
Imperative	C, Go	JavaScript, Python, Ruby
Object-oriented	Java, C++, C#, Scala	JavaScript, PHP
Declarative	Haskell, ML	Lisp, Scheme, Prolog

What is an algorithm?

An algorithm

=

A sequence of elementary instructions for solving a given class of problems.

An algorithm must be:

- Unambiguous
- Executable
- Terminating

Two types of instructions in an algorithm

- *Atomic* instructions (e.g. increase x by 1, wait 1 second, launch missile, ...)
- *Control* instructions:

Sequence	First do x, then do y
Choice	If x is true, then do y, else do z
Iteration	As long as x is true, repeat y
Jump	Continue from point x
	...

Algorithms in real life

Where can you find algorithms in your everyday life?
(away from your computer/phone!)

Example programming problem

Write a program that asks the user for two numbers and shows the sum of these two numbers to the user.

Programming step by step

Analyse the problem

$\Downarrow$

Construct a model

$\Downarrow$

Write an algorithm

$\Downarrow$

Implement the algorithm

$\Downarrow$

Test the program

$\Downarrow$

Document the program

$\Downarrow$

Maintain the program

Algorithm for the example

1. Write “Give the first number: ”
2. Read the first number from the user
3. Write “Give the second number: ”
4. Read the second number from the user
5. Calculate the sum of the two numbers and save the result
6. Show the sum on the screen

Implementing the algorithm

To implement the algorithm, we need to know how to tell the computer to:

- Show text to the user
- Read input from the user
- Add two numbers
- Save a number in the computer memory
- Convert a number to text

15 min. break

Java and IntelliJ

The Java programming language

- Object-oriented programming language
- Platform independent
- Very large standard library
- Support for graphics, parallelism, etc.
- Just-in-time compilation

Not specifically designed for beginners,
so not everything will be clear at the start!

Your first Java program

```java
/* A simple greeter program.
   author: Jesper Cockx
*/

public class HelloWorld {

    // Shows a welcome message to the user.
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }

}
```

Comments in Java

Comment = explanation of the program for humans (ignored by the compiler).

```
// This is a single-line comment.

/* This is a multi-line comment
   (also called a block comment) */
```

Classes in Java

Java programs are organized in classes

```
public class ClassName {
  // here go the methods, variables,
  // and other parts of the program.
}
```

`public`: anyone is allowed to use this class.

Each program has exactly one main class containing a main method.

The main method in Java

```
public static void main(String[] args) {
  ...
}
```

`public`: anyone is allowed to use this method.

`static`: this method belongs to the class rather than to a specific object (see part II).

`void`: this method does not produce a result (but it may still have other effects).

`String[] args`: the user input to the program (not used most of the time).

The `System` class

`System` is a part of the Java standard library for basic interaction with the computer.

A method call consists of a class name (or object), the method name, and the method argument(s) between parenthesis.

- `System.out.println("some string");`
- `System.out.print("another string");`

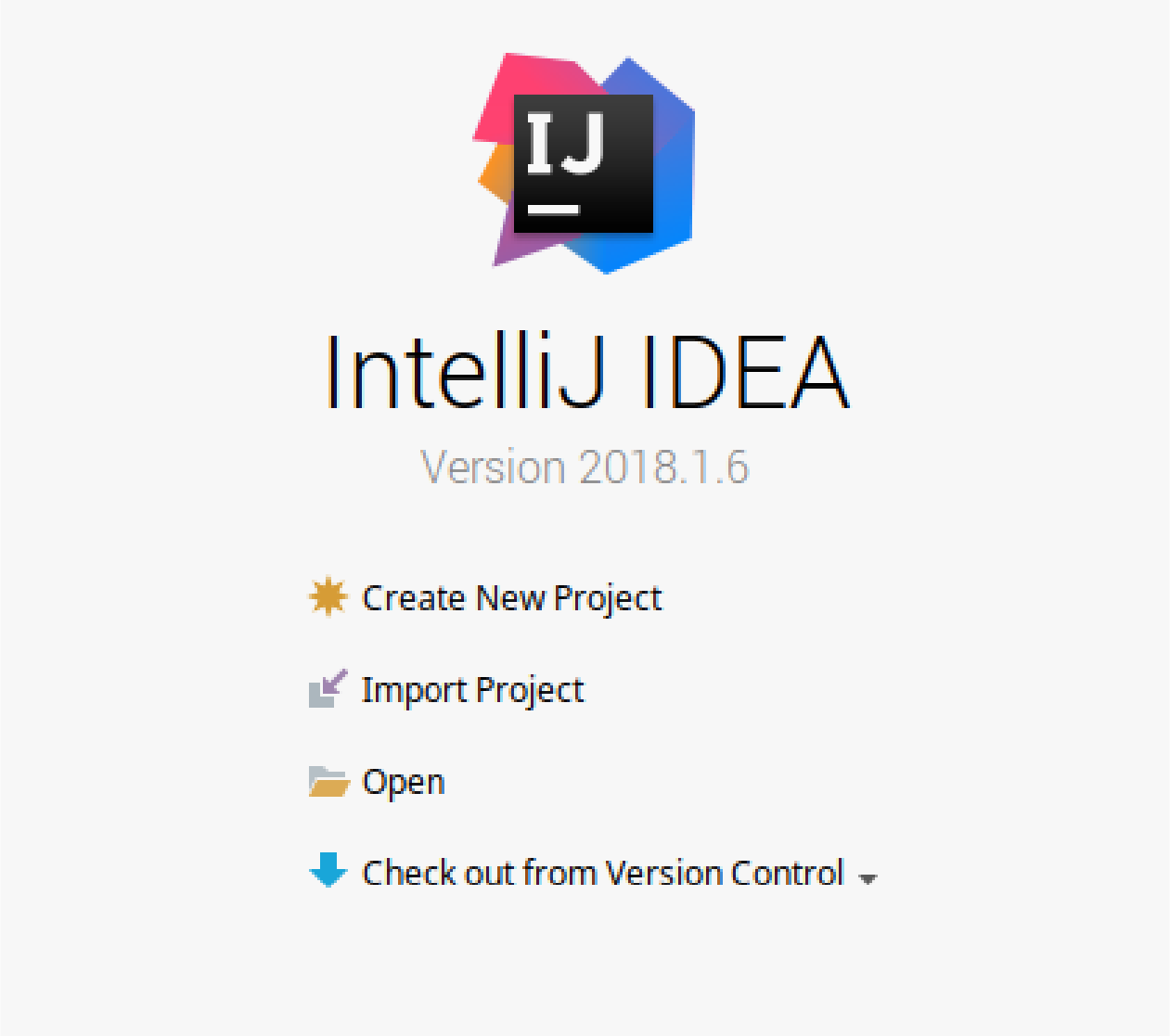
Class name can be skipped for methods in the same class.

The IntelliJ IDE (Integrated Development Environment)

- Text editor
- Code coloring
- Code completion
- Errors and warnings while programming
- Compiling and running code
- Automatic code refactoring
- ...

A good IDE makes programming much easier, try to get the most out of it!

Demonstration: creating your first IntelliJ project



Compile-time errors vs. run-time errors

Compile-time error = error while compiling a program (*wrong variable name, missing parenthesis, ...*)

Run-time error = error while running a program (*program crashes or gives wrong output*)

The `String` class

```
String greeting = "Hello, TDA540!";

String longString =
  "This is a veeeeeeeeeeeeeeeeeery"
  + "long string that doesn't fit"
  + "on one line.";

System.out.println(greeting);
```

Table 4 Java Number Types

Type	Description	Size
`int`	The integer type, with range –2,147,483,648 (`Integer.MIN_VALUE`) . . . 2,147,483,647 (`Integer.MAX_VALUE`, about 2.14 billion)	4 bytes
`byte`	The type describing a single byte consisting of 8 bits, with range –128 . . . 127	1 byte
`short`	The short integer type, with range –32,768 . . . 32,767	2 bytes
`long`	The long integer type, with about 19 decimal digits	8 bytes
`double`	The double-precision floating-point type, with about 15 decimal digits and a range of about $\pm 10^{308}$	8 bytes
`float`	The single-precision floating-point type, with about 7 decimal digits and a range of about $\pm 10^{38}$	4 bytes
`char`	The character type, representing code units in the Unicode encoding scheme (see Random Fact 2.2)	2 bytes

Table 2.4

Some operations on numbers

```
int number1    = 1;
int number2    = 1 + 1;   // addition
int number3    = 9 - 2;   // subtraction
int number4    = 2 * 3;   // multiplication
int number5    = 7 / 2;   // integer division (= 3)
int number6    = 7 % 2;   // remainder after division
int number7    = -1;      // negative number
double number8 = 1.5;     // fractional number
double number9 = 7.0 / 2; // real division (= 3.5)
```

Wrapper classes

Each primitive type has a wrapper class with additional operations:

```
int largest = Integer.MAX_VALUE   //  2147483647
int smallest = Integer.MIN_VALUE  // -2147483648
String numberString =
    Integer.toString(12345)         // "12345"
int myNumber =
    Integer.parseInt("123" + "45")  // 12345
```

Variables in Java

Variable = an intermediate result of a program.

```
int  favoriteNumber  =   6   ;
```

↑ *type* ↑ *name* ↑ *value*

Give meaningful names to variables!

Rules for naming variables

A variable name (or identifier) may consist of:

Letters: $a \ldots z$ and $A \ldots Z$

Numerals: $0 \ldots 9$

Special characters: # and _

A variable name can not start with a number

Convention: start variables with a small letter

Updating a variable

You only have to mention the type of a variable the first time:

```
int number1 = 1;
int number2 = 1;
number1 = number2 + 1;
number2 = number1 + number2;
```

Table 3 Variable Names in Java

Variable Name	Comment
`canVolume1`	Variable names consist of letters, numbers, and the underscore character.
`x`	In mathematics, you use short variable names such as x or y. This is legal in Java, but not very common, because it can make programs harder to understand (see Programming Tip 2.1 on page 38).
⚠ `CanVolume`	**Caution:** Variable names are case sensitive. This variable name is different from `canVolume`, and it violates the convention that variable names should start with a lowercase letter.
⊘ `6pack`	**Error:** Variable names cannot start with a number.
⊘ `can volume`	**Error:** Variable names cannot contain spaces.
⊘ `double`	**Error:** You cannot use a reserved word as a variable name.
⊘ `ltr/fl.oz`	**Error:** You cannot use symbols such as / or.

Table 2.3

Graphical interfaces with Swing

```java
import javax.swing.*;

public class HelloSwing {
  public static void main(String[] args) {
    JOptionPane.showMessageDialog(null,
      "Hello, world!");
    String name = JOptionPane.showInputDialog(
      "What is your name?");
    String greeting = "Hi, " + name + "!";
    JOptionPane.showMessageDialog(null, greeting);
  }
}
```

Live coding session

Assignment: *Write a program that asks the user for two numbers and shows the sum of these two numbers to the user.*

Algorithm:

1. Write “Give the first number: ”
2. Read the first number from the user
3. Write “Give the second number: ”
4. Read the second number from the user
5. Calculate the sum of the two numbers and save the result
6. Show the sum on the screen

What's next?

Next lecture (tomorrow!):
if-statements and the Java standard library.

To do:

- Read the book:
  - Today: parts of chapter 1 and 2
  - Next lecture: parts of chapter 2 and chapter 3
- Install Java and IntelliJ