2 Problem, Algorithm, Program

- What is a ‘problem’?
- What is an ‘algorithm’?
- What is a ‘program’?
- Insertion: Java examples, part 1
- Flow control of algorithms
- Insertion: Java examples, part 2
- Structured composition of algorithms

- A **problem** is identified by an initial state (‘problem exists’ or ‘problem identified’) and by the aim to reach a target state (‘problem solved’).
- An **algorithm** is a precise, finite method for solving a problem, i.e., for the transition from ‘problem identified’ to ‘problem solved’.
- A **program** is the encoding of an algorithm with a programming language for execution on a computer.

Specification of the problem:

- A precise algorithm requires a precise representation of the initial situation.
- We thus need to begin with an analysis of the problem in order to generate a specification that establishes the following:
  - information available in the initial situation.
    (input data and their allowed values, i.e., their domain)
  - desired information.
    (output data and their possible values, i.e., their range)
  - any conditions that need to be observed while deriving the desired information.
- The specification of the problem should be as detailed, as exact, and as complete as possible.

Examples for specifications of problems:

- Given: the three numbers 5, 7, and 11.
  Wanted: their arithmetic mean.
- Given: flour, water, olive oil, salt.
  Wanted: pizza dough made from these ingredients.
  Conditions: One can use at most 400g flour and 2 spoons of olive oil.
- Given: place of residence Trier, holiday destination Tahiti, possible dates for vacation, set of flight schedules.
  Wanted: proposals for trips.
  Conditions: departure on Friday or Saturday, duration 4 weeks, cost not exceeding 3000 euros

2 Problem, Algorithm, Program

- What is a ‘problem’?
- What is an ‘algorithm’?
- What is a ‘program’?
- Insertion: Java examples, part 1
- Flow control of algorithms
- Insertion: Java examples, part 2
- Structured composition of algorithms

The term **algorithm** dates back to the Arabic mathematician

Mohammed ibn Musa abu Djafar al Khowarizmi

(ca. 783 - 850) and his book

‘Kitab al muhtasar fi hisab al gebr we al muqabala’

(‘The Compendious Book on Calculation
by Completion and Balancing’).

In the Latin translation each section started with ‘Dixit algorismi’ (‘thus said al Khowarizmi’), which led to the later term algorithm.

What is an ‘algorithm’?

We need to clarify the following questions:

- Which processing steps are required in an algorithm?
- How can we control the flow of the processing steps?
- How can we represent and write down algorithms?
- Which properties of algorithms are interesting?

Intuitive algorithms

Examples from everyday life:

process	algorithm	example for processing step
knit a pullover	knitting pattern	purl – plain
construct a model car	assembly instructions	glue outside mirror to left door
make a cake	recipe	beat 3 eggs until fluffy
play music piece	sheet of music	play c”’

Such daily operation procedures are often not exact and leave room for interpretations, so they are not algorithms.

Example: starting with a car from the right curb as **intuitive algorithm**

1. fasten seat belt, check rear-view mirror, if necessary pull handbrake, insert idle.
2. briefly actuate ignition.
3. If engine does not start, repeat step 2.
4. look around and wait for a traffic gap
5. pedal the clutch with left foot and brake with right foot, enable the left indicator, release the handbrake.
6. hold clutch, engage 1st gear.
7. accelerate with right foot, turn the steering wheel to the left.
8. release the clutch, accelerate, turn the steering wheel such that the car reaches the right lane.
9. ...

Observations from the examples

- The individual steps are executed by a processor.
- A single processor executes one step after the other.
- Usually a step can only be executed after the previous step was successfully completed.
- The execution of a step can depend on a condition (*if a sufficiently large gap in the traffic exists then drive off, otherwise continue waiting*).
- Some steps must be repeated multiple times (*repeatedly briefly actuate ignition*).
- Repetitions always end when a certain condition gets true (*... until the engine starts*).

- In some cases the order in which some steps are executed does not matter. Thus transposition and parallel execution are possible (*pedal the clutch with left foot and brake with right foot*)
- In principle an algorithm executes a transformation from an initial state to a final state. Or: for a given input the application of an algorithm creates a well-defined output.
- In a mathematical sense an algorithm defines a map $\boldsymbol{f} : \boldsymbol{E} \rightarrow \boldsymbol{A}$ from the set of input data $\boldsymbol{E}$ to the set of output data $\boldsymbol{A}$.
- Usually the algorithm cannot be applied in every initial state.

2 Problem, Algorithm, Program

- What is a ‘problem’?
- What is an ‘algorithm’?
- What is a ‘program’?
- Insertion: Java examples, part 1
- Flow control of algorithms
- Insertion: Java examples, part 2
- Structured composition of algorithms

program = formal representation of an algorithm in a specialized language (**programming language**)

Example: Java

```
1  public class K2B01E_first_Example {
2    public static void main(String[] args) {
3
4      int index;
5      int count;
6
7      index  =  0;
8      count = 10;
9
10     while ( index <= count )
11     {
12       System.out.println( index * index );
13       index = index + 1;
14     }
15   }
16 }
```

2 Problem, Algorithm, Program

- What is a ‘problem’?
- What is an ‘algorithm’?
- What is a ‘program’?
- Insertion: Java examples, part 1
- Flow control of algorithms
- Insertion: Java examples, part 2
- Structured composition of algorithms

A first Java example

```
1  public class K2B01E_first_Example {
2    public static void main(String[] args) {
3
4      int index;
5      int count;
6
7      index  =  0;
8      count = 10;
9
10     while ( index <= count )
11     {
12       System.out.println( index * index );
13       index = index + 1;
14     }
15   }
16 }
```

Typical components:

- separation symbols like ; , , parenthesis like () , [] , {}
- header (lines 1-2), declarations (lines 4-5), assignments (lines 7-8,13), loops (lines 10-14)
- method calls (line 12)

Typical sequence for writing a program:

1. Write the program with a ‘simple’ ASCII editor
   - Text editors like MS Word, LibreOffice Write etc. are unsuitable!
   - alternative with MS Windows: Notepad, Notepad++, ...
   - with Linux: kwrite, nano, vi, gedit, geany, ...
   - or use an integrated development environment, e.g., Java-Editor, Eclipse, IntelliJ, ...
2. file name: `K2B01E_first_Example.java` (as in the header)
3. translate and start in the console:

```
javac K2B01E_first_Example.java
java K2B01E_first_Example
```

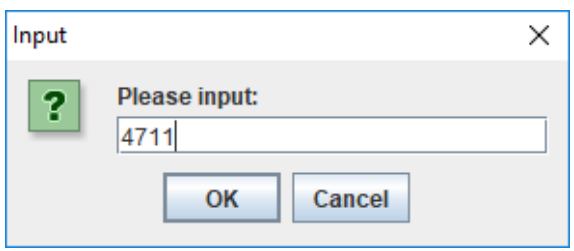
Example: user input/output via the console

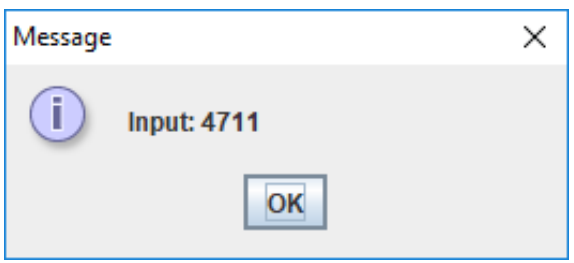
```
1  public class K2B02E_IO_Console {
2    public static void main(String[] args){
3      int    inputNumber;
4      String inputText;
5
6      inputText = System.console().readLine("Input: ");
7
8      inputNumber = Integer.parseInt(inputText);
9
10     System.out.println("Output: " + inputNumber);
11   }
12 }
```

```
1 > javac K2B02E_IO_Console.java
2 > java  K2B02E_IO_Console
3 Input: 1243
4 Output: 1243
```

Example: user input/output with popup windows

```
1  import javax.swing.JOptionPane;
2
3  public class K2B03E_IO_Dialogue {
4    public static void main(String[] args) {
5      int    inputNumber;
6      String inputText;
7
8      inputText=JOptionPane.showInputDialog(null,"Please input:");
9
10     inputNumber = Integer.parseInt(inputText);
11
12     JOptionPane.showMessageDialog (null, "Input: "
13                                                + inputNumber);
14   }
15 }
```





Example: user input as arguments to the program

```
1  public class K2B04E_IO_Parameters {
2    public static void main(String[] args) {
3      int    inputNumber;
4      String inputText;
5
6      inputText=args[0];
7
8      inputNumber = Integer.parseInt(inputText);
9
10     System.out.println("Input: " + inputNumber);
11   }
12 }
```

- as a standalone program in the console:

```
1 javac K2B04E_IO_Parameters.java
2 java  K2B04E_IO_Parameters 4711
```

- IDE: in **Eclipse** via **Run**$\mapsto$ **Run Configuration**, then configure under **Arguments**

Example: user input with ‘Scanner’

```java
import java.util.Scanner;

public class K2B05E_IO_Scanner {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);

    int inputNumber;

    System.out.println("Input: ");

    inputNumber = sc.nextInt();

    System.out.println("Output: " + inputNumber);
  }
}
```

```
> javac K2B05E_IO_Scanner.java
> java  K2B05E_IO_Scanner
Input:
1243
Output: 1243
```

Example: arrays for storing multiple numbers

```
1  public class K2B06E_Arrays {
2    public static void main(String[] args) {
3      int index;
4      int count;
5
6      int[] array;
7
8      count=5;
9      array = new int[count];
10
11     index=0;
12     while (index<count)
13     {
14       array[index] = index*index;
15       index=index+1;
16     }
17
18     index=0;
19     while (index<count)
20     {
21       System.out.println(array[index]);
22       if (index<count) System.out.println("-");
23       index=index+1;
24     }
25   }
26 }
```