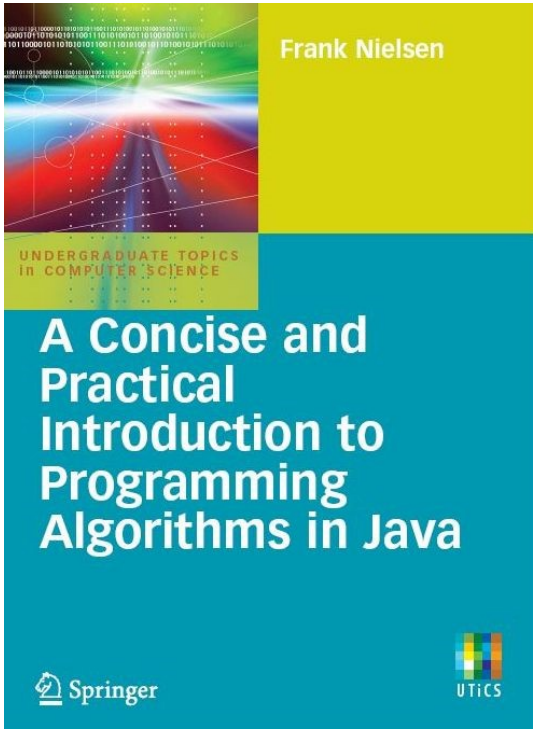
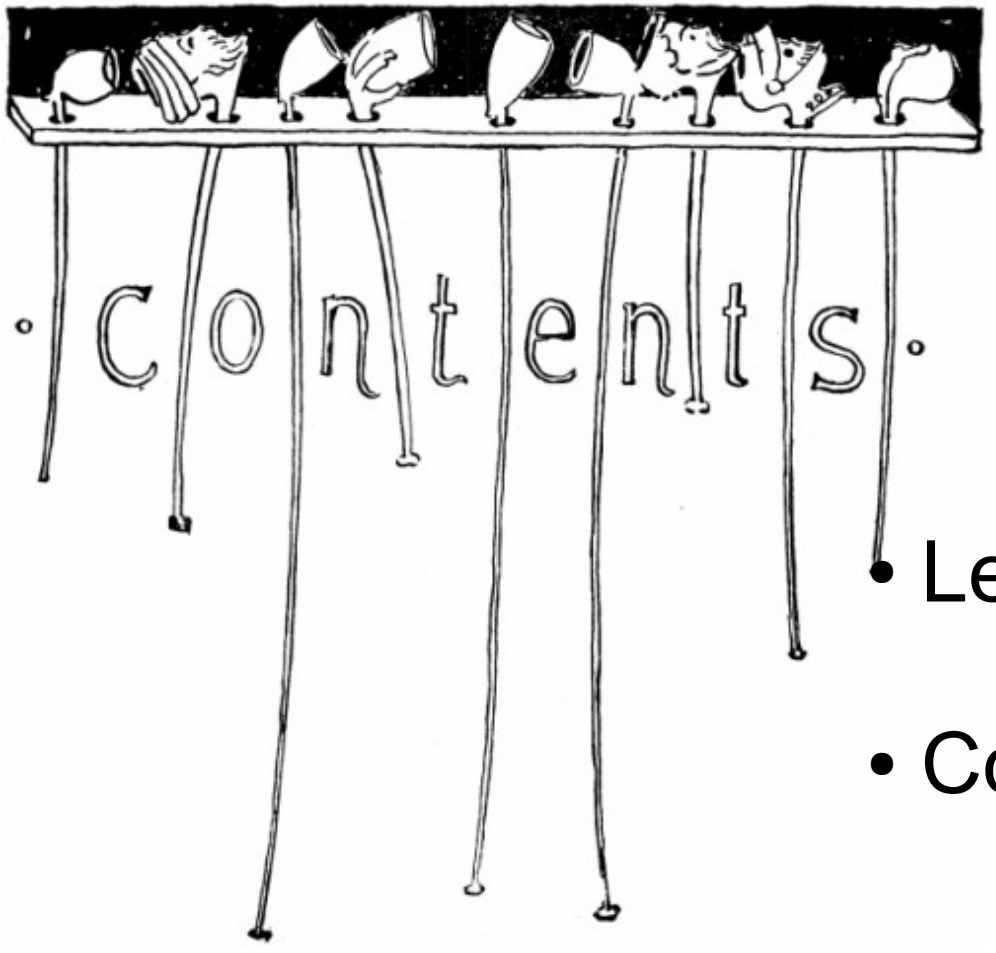
A Concise and Practical Introduction to Programming Algorithms in Java



Chapter 1: Expressions, Variables, and Assignments

Frank NIELSEN

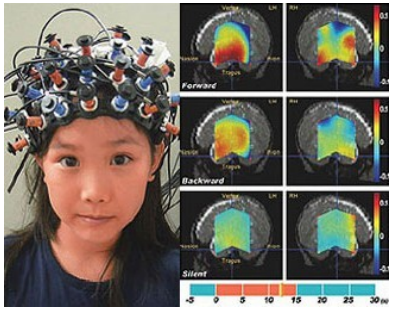
nielsen@lix.polytechnique.fr



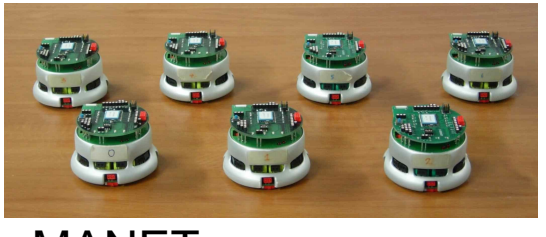
Contents

- Learn to **program** with/in Java
- Computing as a science
  (some basic principles)
- Popular (computer) science

HCI



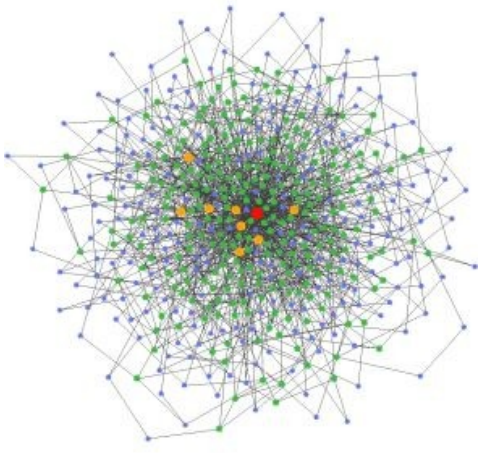
BMI



MANET

Jobs & Computer Science

- **Industry**
  - CS Industry
  - Others (information systems)
- **Administration**
- **Research & Development**

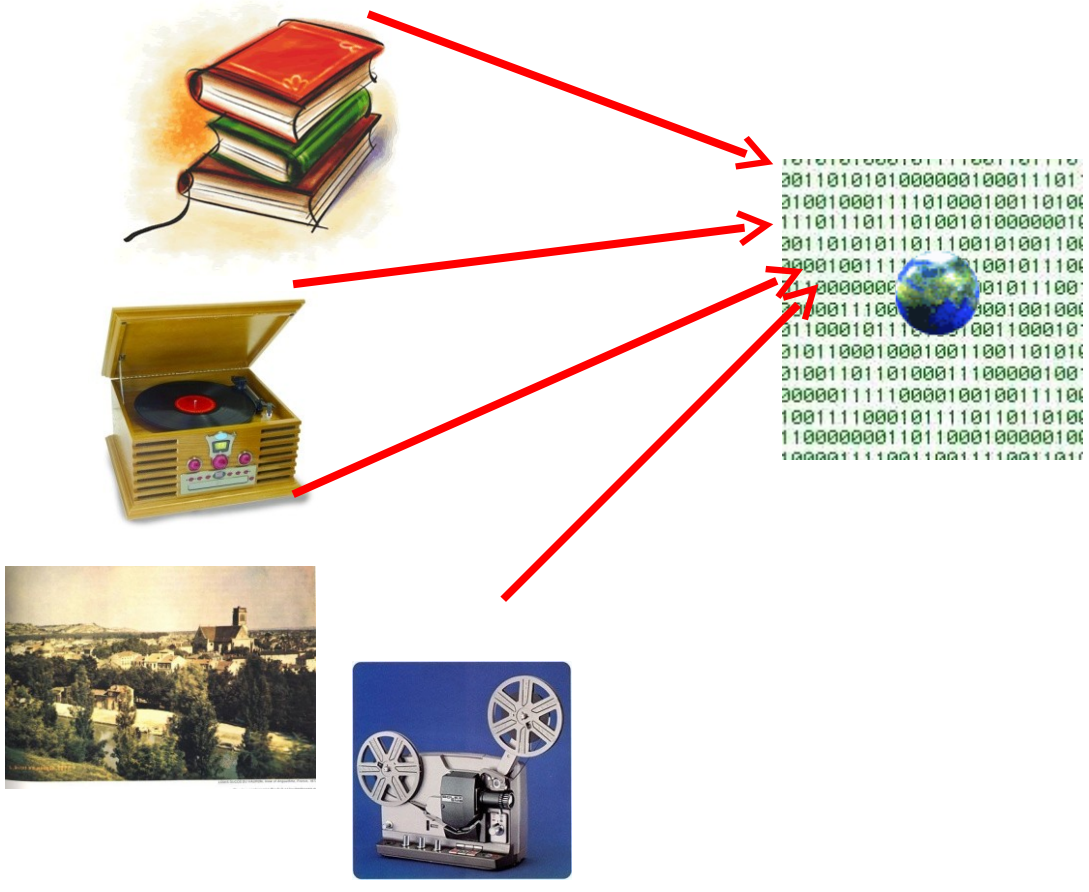


Not feeling fluent with CS today, is like not being able to drive a car !

Digital world

Benefits of the *analog-to-digital* paradigm shift?

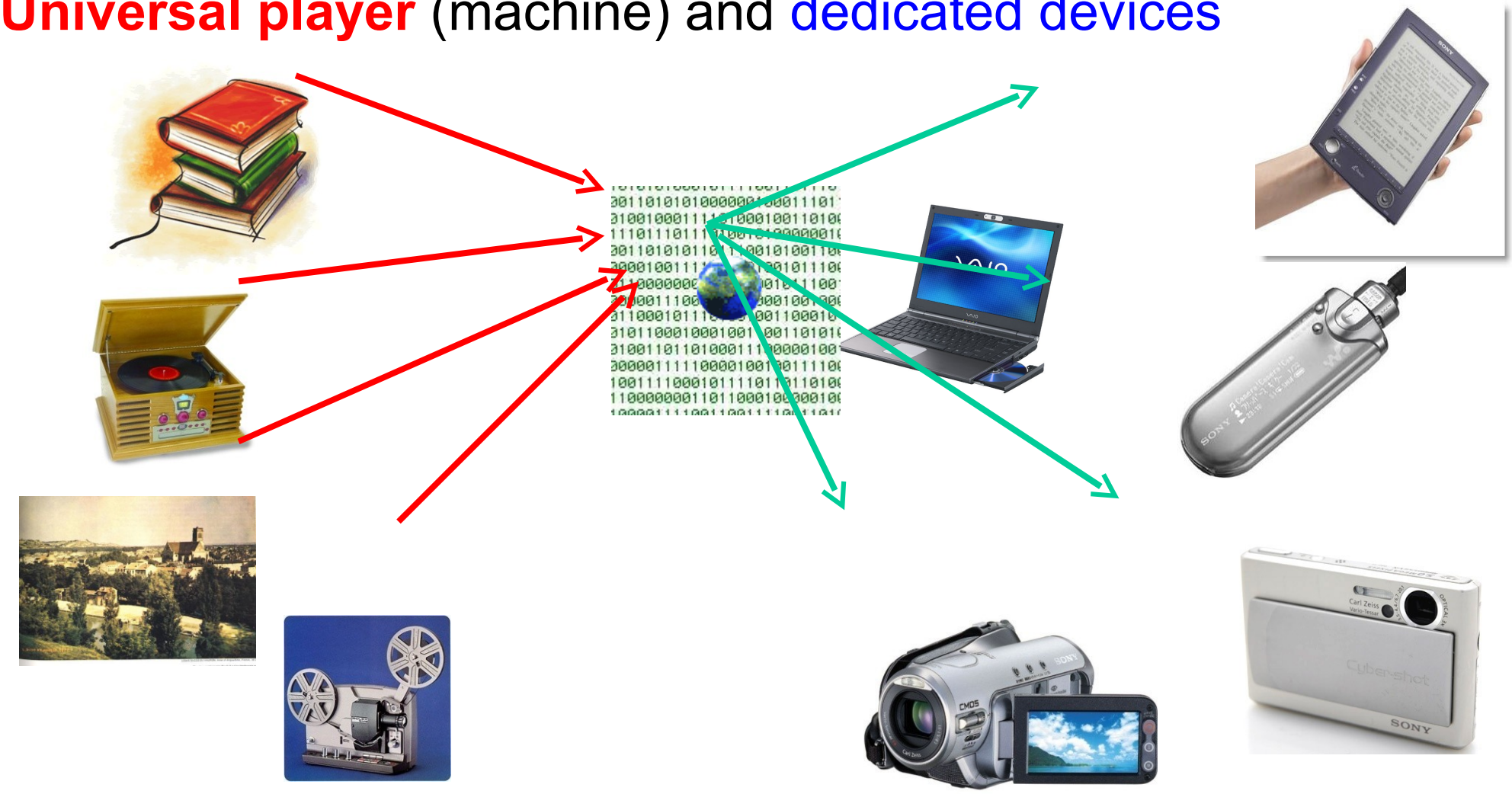
- ***Dissociate* contents from support** : digitize/“binarize”



Contents become mere binary 0/1 strings

Digital world

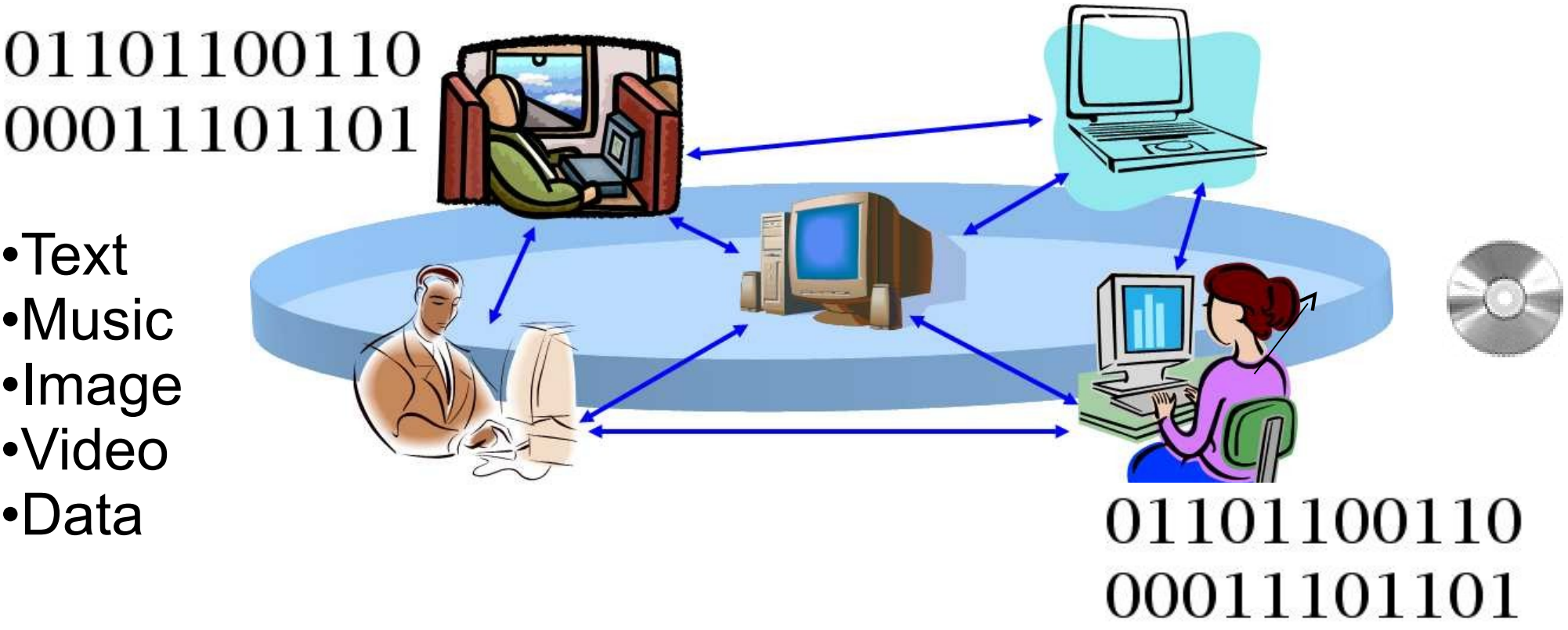
- **Universal player** (machine) and dedicated devices



"Multiple 0/1 readers"

Digital world

- **Generic algorithms**:
  copying, compressing, transmitting, archiving, etc.



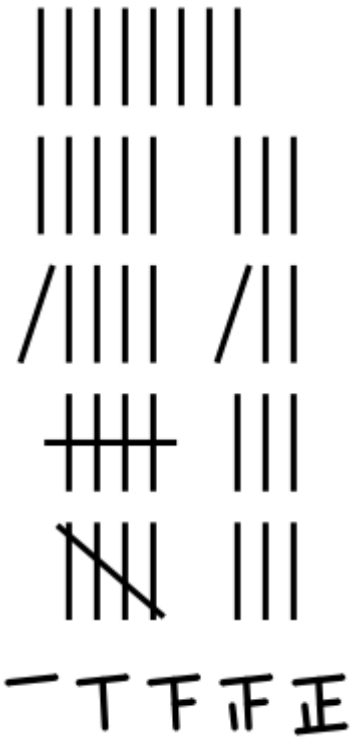
Raise the question: What is the (digital) information?

Digital world: Why 0/1 bits?

Information, first needs of counting...

Unary numeral systems:

8



Binary numeral systems:

0	0	0	0	→	0000
			1	→	0001
		1	0	→	0010
			1	→	0011
	1	0	0	→	0100
			1	→	0101
		1	0	→	0110
			1	→	0111
1	0	0	0	→	1000
			1	→	1001
		1	0	→	1010
			1	→	1011
	1	0	0	→	1100
			1	→	1101
		1	0	→	1110
			1	→	1111

4 bits
for counting
0 to 15

Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary:	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

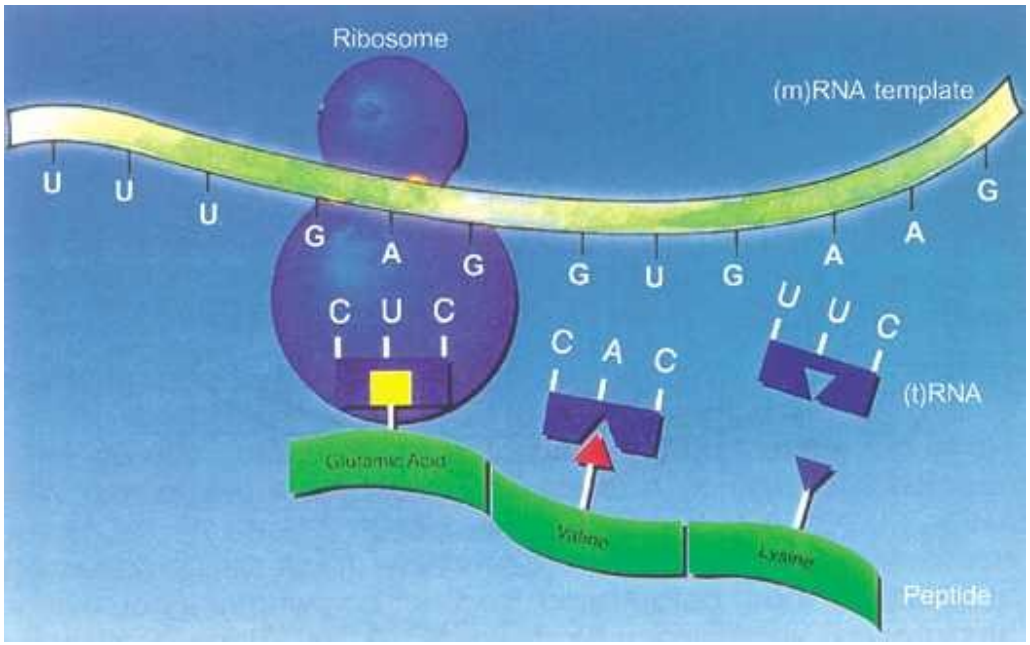
Linear number of bits for counting vs Logarithmic number of bits for counting

Nature of computing?

- **Generic algorithms**:
  copying, transmitting... ...genetics...

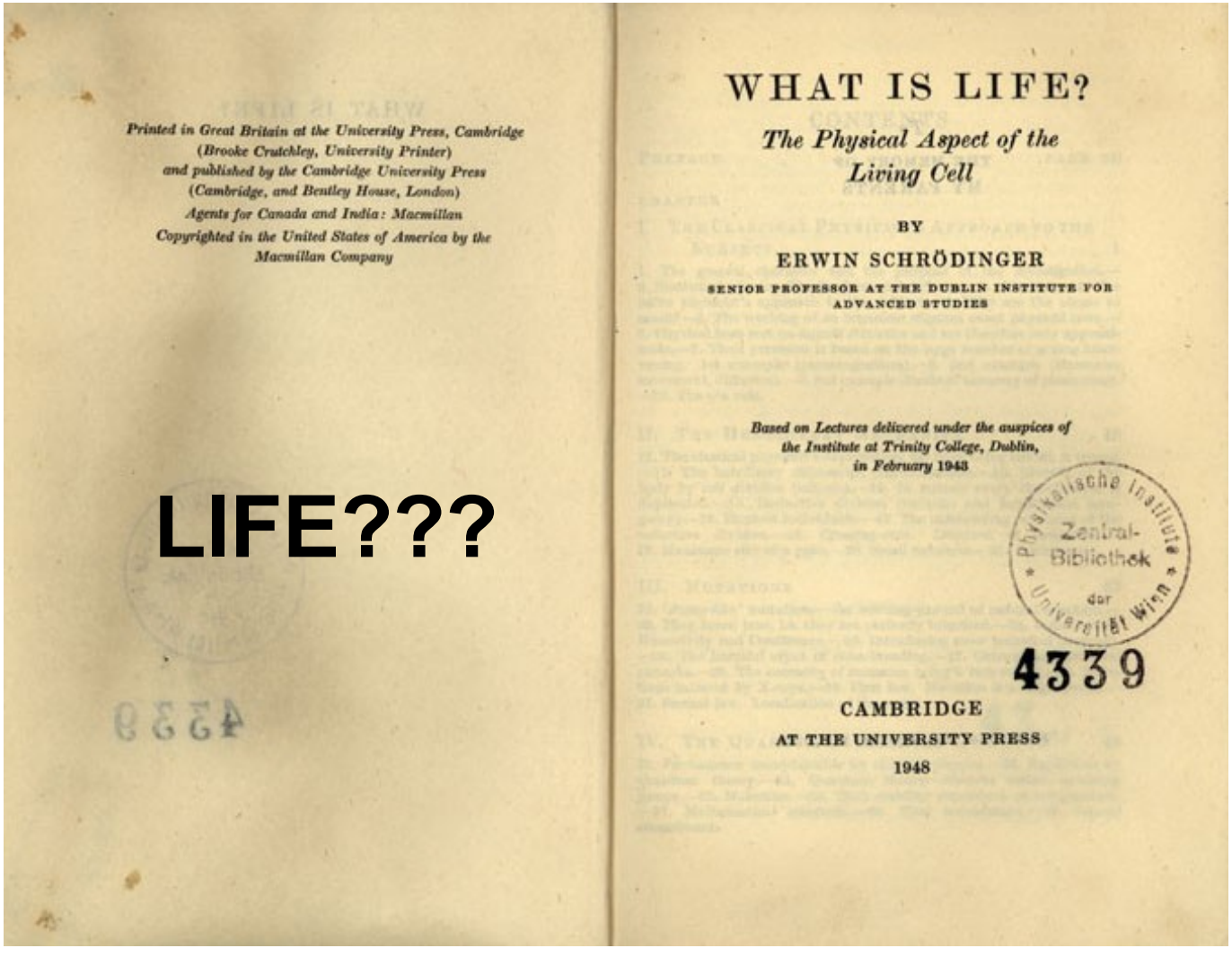
DNA (double-helix structure of DNA)

1953, James Watson and Francis Crick (Nobel prize)



Genetics

Nature of computing?



Printed in Great Britain at the University Press, Cambridge
(Brooke Crutchley, University Printer)
and published by the Cambridge University Press
(Cambridge, and Bentley House, London)
Agents for Canada and India: Macmillan
Copyrighted in the United States of America by the
Macmillan Company

LIFE???

WHAT IS LIFE?

The Physical Aspect of the Living Cell

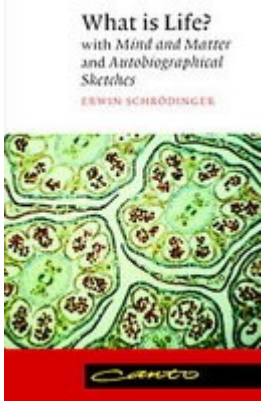
BY

ERWIN SCHRÖDINGER

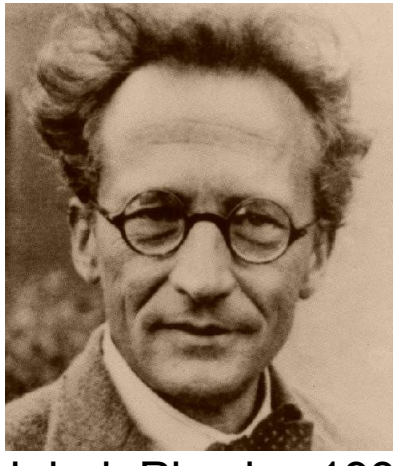
SENIOR PROFESSOR AT THE DUBLIN INSTITUTE FOR ADVANCED STUDIES

Based on Lectures delivered under the auspices of the Institute at Trinity College, Dublin, in February 1943

CAMBRIDGE
AT THE UNIVERSITY PRESS
1948



Transmit crystals?



Nobel, Physics 1933

First envisioned by Erwin Schroedinger (What is life?, 1944)

Digital world/computing

Ubiquitous computing= computing everywhere

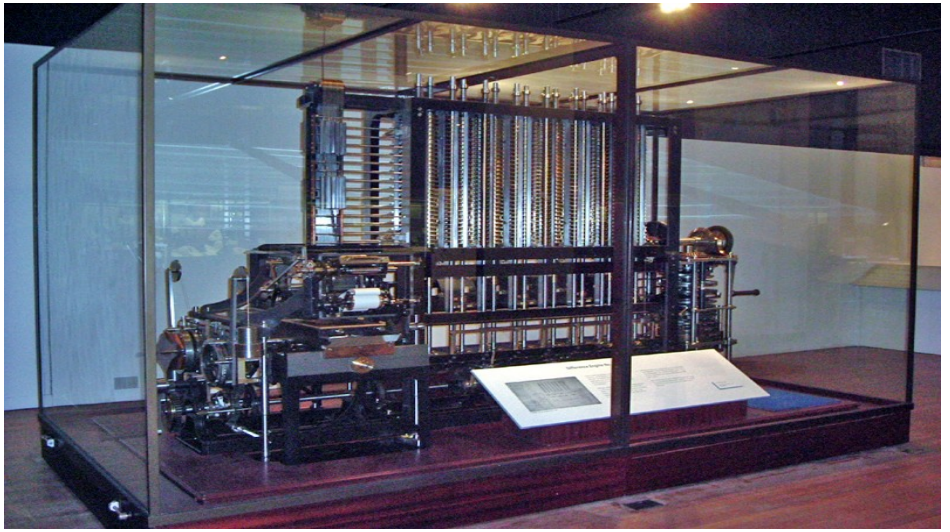
Digital = Binary + Calculations

← New features

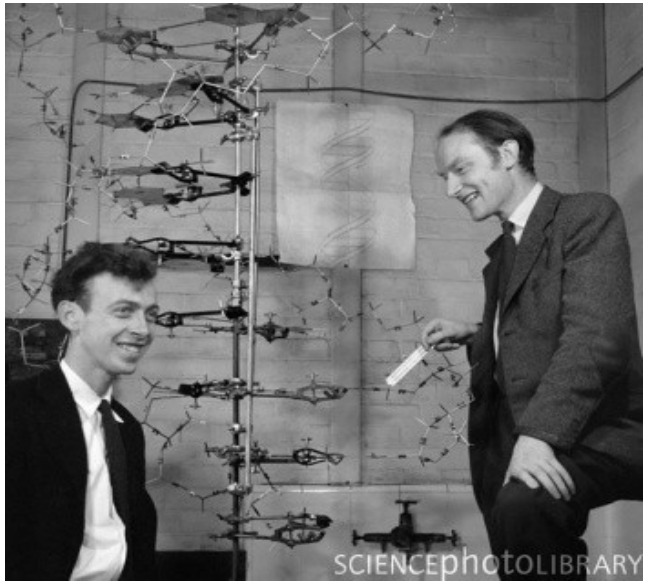
Example:
Computational photography

Computer science *is not* programming PCs

Computers
=
computing **machineries**



Difference engine of Charles Babbage
(conceived in 1822 on paper, built much later on)



Computing is a principle of reality (and science)
Watson and Crick 1951 (DNA double helix heredity)

Computing is 21st Century's Science of integration

INFORMATIQUE=INFORmation + autoMATIQUE

Information= Data sets, input (discrete binary sequences of 0/1)
Automatic= *General* recipe that works on *any* input
= **ALGORITHM**

Al-Khwarizmi
(790 - 840)

Al-Khwarizmi: Scholar of
scientifically flourishing Bagdad:

- Algorithmi (latinization) -> Algorithm
- Al jabr -> Algebra

A page from the Kitab al-jabr wa l-muqabala. (Esposito, J.L., editor, Oxford History of Islam. Oxford University Press, Oxford, 1999)

Provide readers a generic ***pipeline*** solution to solve a quadratic equation:

$$a x^2 + b x + c = 0$$

a: 1 b: 3 c: -4 Calculate Solutions

The solutions are:

x_1: 1 + 0 i

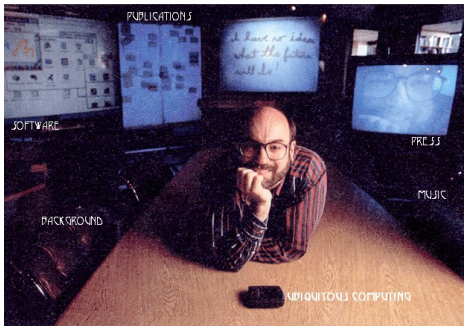
x_2: -4 - 0 i

http://www.akiti.ca/Quad2Deg.html

21st century computer science

- Computers (and computing) are **omnipresent**
  -> **Ubiquitous computing** (Mark Weiser)

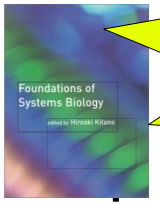
  Computers are abundant and versatile:



1952-1999
Xerox parc chief scientist

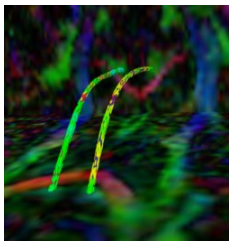
(Many more devices than PCs)

- Computing **impacted** all Sciences:
  Computational sciences
  Eg., Biology -> Systems biology
  (simulation-prediction-experience in wet lab)



Science of
Integration
(complex systems)

- The Science of computing is Computer Science (CS):
  Deep theoretical questions and important technologies
  (eg., medical imaging such as DT-MRI, economy)

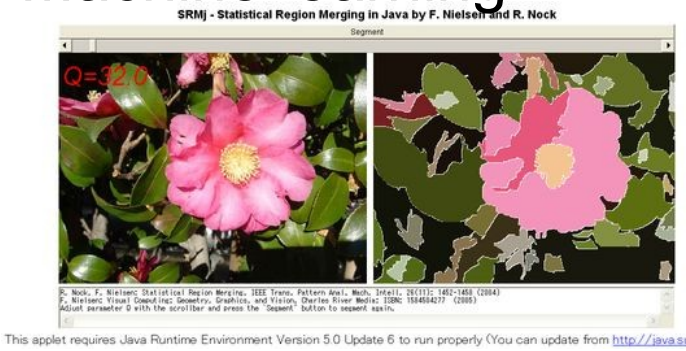


DW-MRI

Flavor of my research in computer science

Visual computing:

- Computational geometry,
- Computer vision,
- Computer graphics,
- Machine learning

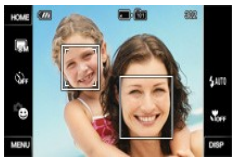


For example, tackling **computational photography**

Reinventing the photography: taking, sharing and experiencing photos...

Analog camera

Digital camera



Smile shutter

Everything has yet to be invented!!!

Beyond 2D pixels
Beyond single flash
etc...

Computer science is (also) for creative minds

Not only the hardcore mathematical problems to solve, but also innovation by unleashing the power of digital calculus for soft problems:

Human Computer Interactions (HCI), design

Example: computational photography project (2004)

Non-photorealistic camera (NPR)

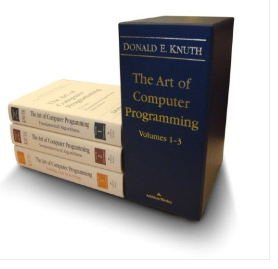
NPR Camera

Algorithms and their performances (resource/complexities)

There is usually *not* a single recipe for solving the task:

Eg., compute 5422x2319
(human decimal, machine binary, indian base 60, many tricks, etc.)

Donald Knuth



How to evaluate and compare different algorithms?

Clean framework for *assessing* the use of *ressources*:
- time,
- memory,
- #communications,
- etc.

Judge the *generic* algorithms not for a given instance.
Therefore, analyze:
- Worst-case complexity
- Average-time complexity
- Modern challenges (inplace, i/o bottlenecks & streaming, etc.)
- Etc.

Programming algorithms in Java

- Conceived by **Bill Joy** (SUN co-founder) and **James Gosling**


- Started in 1990, for the "next wave in computing"
- On-time for the Internet and WEB (applets are Java applications, Javascript, etc.)
  Cross-platform= runs on various operating systems (Windows, UNIX, Leopard, etc.)
- **Typed language** (a=b, with a and b from different types will generate a compiler error)
- **Object oriented** (OO, ease the conception and modularity of applications)

- Rich set of Applications Programming Interface (API)
- Free Software Development Kit on many platforms (SDK)
- Verbose for catching bugs and debugging applications.

Why programming languages?

Machines are “stupid”: they obey you **100%**
-> Need to fully and precisely specify your intentions
(no room for ambiguity, the bug is yours!!!)

... Machines only “understand” 0/1 binary sequences
(eg., instruction codes of microprocessors)

Machine = Processing + Peripherals (I/O)
... controlled by an Operating System (OS)



But Human masters “natural language”
... and we need to unleash *ease of programming*
ASSEMBLER, FORTRAN, ALGOL, BASIC, ...JAVA

Key principle of CS: **Bootstrapping**!
use existing languages to create more powerful languages:
Python, Ruby, etc.

My first (java) program

Programmers and CScientists cherrish...
... their “Hello World” programs

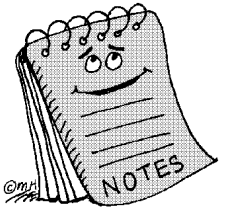
```
class FirstProgram{
  public static void main (String[ ] args){
    System.out.println("Hello INF311 !");
  }
}
```

First programs often looks magic!

Special function `main`: entry of the program

My first (java) program

- Type this program into a text editor (nedit, notepad)
  Save this “text” as FirstProgram.java



- Compile the program FirstProgram.java

```
prompt% javac FirstProgram.java
```

- Execute the compiled program

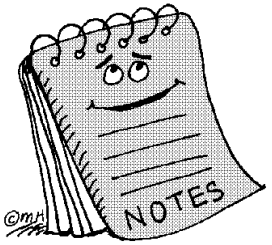
```
prompt% java FirstProgram

prompt% Hello INF311 !
```

```
Hello INF311 !
Press any key to continue...
```

My first (java) program

1) EDIT and SAVE

`FirstProgram.java`



High-level language
concepts/abstraction

2) COMPILE

`FirstProgram.class`

(Java Byte code in .class)

3) EXECUTE

```
java FirstProgram
```

(Java Virtual machine: JVM)
... low-level language
instructions for processors

My first algorithm in Java:
A solver for quadratic equations

In Java

→ Install the SDK
(you do not have to do this in room machines)

http://www.java.com/fr/

Input: a, b, c of the quadratic equations
Solution: the at most two real roots

$$a x^2 + b x + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

Programming: Solver for quadratic equations

QuadraticEquationSolver.java

```
class QuadraticEquationSolver
{
public static void main(String[] arg)
    {
    double a,b,c;

    a=Math.sqrt(3.0);
    b=2.0;
    c=-3.0;

    double delta=b*b-4.0*a*c;
    double root1, root2;

    root1= (-b-Math.sqrt(delta))/(2.0*a);
    root2= (-b+Math.sqrt(delta))/(2.0*a);

    System.out.println(root1);
    System.out.println(root2);

    System.out.println("Let us check the roots:");
    System.out.println(a*root1*root1+b*root1+c);
    System.out.println(a*root2*root2+b*root2+c);
    }
}
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

Programming simple formula

```
class QuadraticEquationSolver
{
public static void main(String[] arg)
    {
    double a,b,c;

    a=Math.sqrt(3.0);
    b=2.0;
    c=-3.0;

    double delta=b*b-4.0*a*c;
    double root1, root2;

    root1= (-b-Math.sqrt(delta))/(2.0*a);
    root2= (-b+Math.sqrt(delta))/(2.0*a);

    System.out.println(root1);
    System.out.println(root2);

    System.out.println("Let us check the roots:");
    System.out.println(a*root1*root1+b*root1+c);
    System.out.println(a*root2*root2+b*root2+c);
    }
}
```

Variable (declare)

Assignments

Declare+Assign

Programming simple formula

```
class QuadraticEquationSolver
{
public static void main(String[] arg)
    {
    double a,b,c;

    a=Math.sqrt(3.0);
    b=2.0;
    c=-3.0;

    double delta=b*b-4.0*a*c;
    double root1, root2;

    root1= (-b-Math.sqrt(delta))/(2.0*a);
    root2= (-b+Math.sqrt(delta))/(2.0*a);

    System.out.println(root1);
    System.out.println(root2);

    System.out.println("Let us check the roots:");
    System.out.println(a*root1*root1+b*root1+c);
    System.out.println(a*root2*root2+b*root2+c);
    }
}
```

Arithmetic expressions

Display

Programming: Solver for quadratic equations

Use *any* **text editor** to program
(**nedit** in UNIX, **notepad** under windows)

Indentation is up to you
-> helps read programs

quadraticequationsolver.java - Bloc-notes

Fichier Edition Format Affichage ?

```
class QuadraticEquationSolver{


public static void main(String[] arg)
        {
        double a,b,c;

        a=1.0;
        b=2.0;
        c=-3.0;


        double delta=b*b-4.0*a*c;


        double root1, root2;


        root1= (-b-Math.sqrt(delta))/(2.0*a);
        root2= (-b+Math.sqrt(delta))/(2.0*a);



        System.out.println(root1);
        System.out.println(root2);


        System.out.println("Let us check the roots:");
        System.out.println(a*root1*root1+b*root1+c);
        System.out.println(a*root2*root2+b*root2+c);
        }

}
```

Magic code for printing onto the **console**

Compiling and executing a Java program

```
prompt>javac filename.java
```

If no compile error happens, it produces a file **filename.class**

Then excute the compiled code.

```
prompt>java filename
```

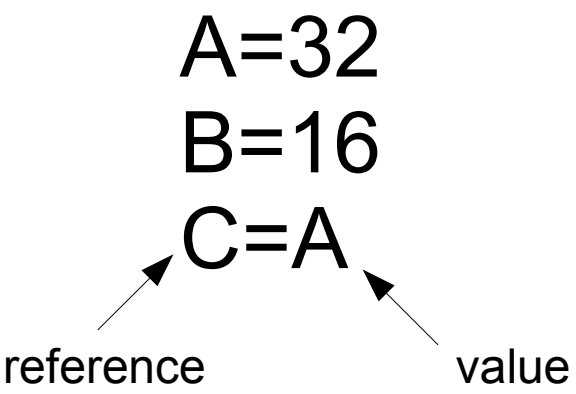
To store output to a file:

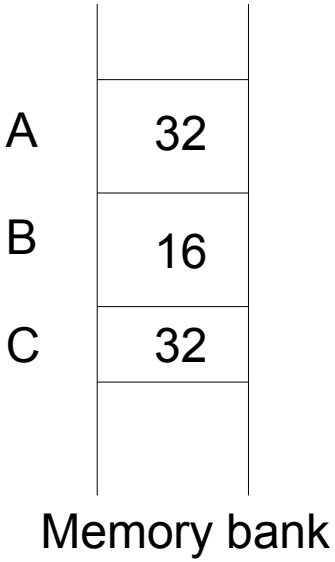
```
prompt>java filename  > result.txt
```

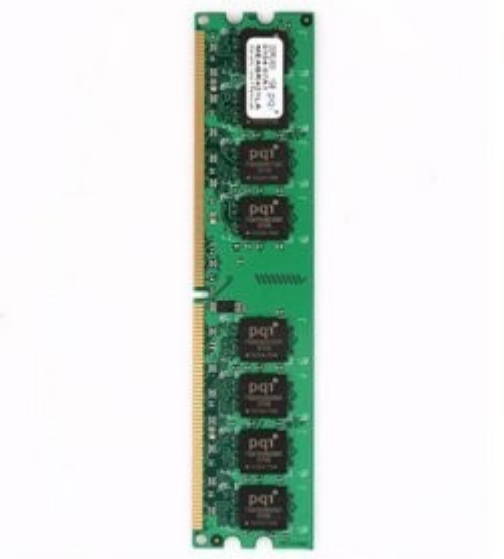
Redirect console to filename result.txt

Fundamentals of Java: Variables

- A variable is *uniquely* named (not a reserved keyword)
- A variable *stores* a value in a memory slot
- The value of a variable *is accessed* by its name
- The value of a variable *can be modified*







Left hand side (reference) and right hand side (value) of = means different things

Fundamentals of Java: Expressions

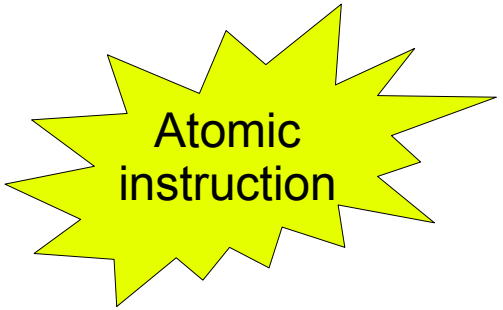
- Formed by variables, operators (+,-,/, x, etc.) and constants (1, Math.PI, etc.)
- Expressions are evaluated and return a result (eventually stored in a variable)
- Operators follow priority rules: 5x3+2 ?
  ...avoid overuse of parenthesis 5x3+2 = (5x3)+2

Few examples of expressions in Java:

```
// Expressions
5+3*x/y
"Hello "+"INF311!"

// Assignment (expressions) terminate with a ;
x=cx + r*Math.cos(theta);
y=cy+ r*Math.sin(theta);
```

Fundamentals of Java: Affectation (sign =)

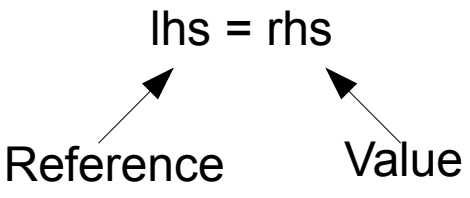
Var = Expression ;

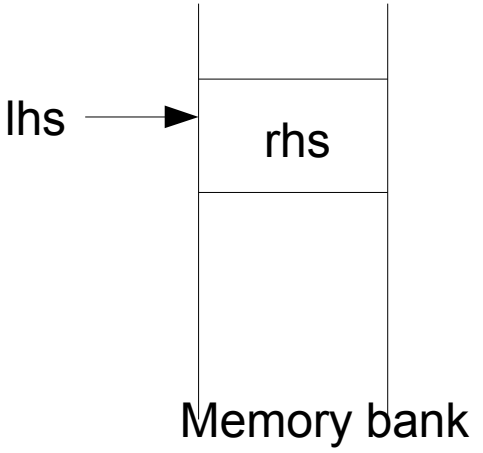


- Var is the name of a variable

- Expression is a well-formed expression

Assignment left hand side=right hand side is decomposed as :
- The Expression is evaluated yielding value v
- The reference (memory slot) of Var is determined
- Value v is stored in the memory slot of Var





Basic types

Type = Domain of values of variables

All variables must be typed in Java

Basic types (=basic data structures):

Integers:

`byte`	8 bits	
`short`	16 bits	
`int`	32 bits	[-2**31,2**31-1]
`long`	64 bits	[-2**63,2**63-1]

Reals:

`float`	(single precision, 32 bits)
`double`	(double precision, 64 bits)

`char` 16 bits (Unicode, world languages)

`boolean true` or `false`

Why do we type variables?

To ensure homogeneous operations

2  +3  =5 

3  +4  =7 

5  +2  =???

Basic types: casting expressions

Euclidean (integer) division versus usual (real) division

```
int p=2;
int q=3;
int quotient=p/q;
int reminder=p%q; // modulo

double div=p/q;

double realdiv=(double)p/(double)q;

System.out.print(quotient);
System.out.print(" ");
System.out.println(reminder);
System.out.println(div);
System.out.println(realdiv);
```

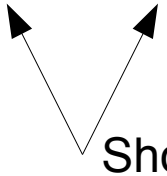
Cast (coercion)

```
0 2
0.0
0.6666666666666666
```

Casting expressions

Implicit casting for assignment

```
x=Expression;
```

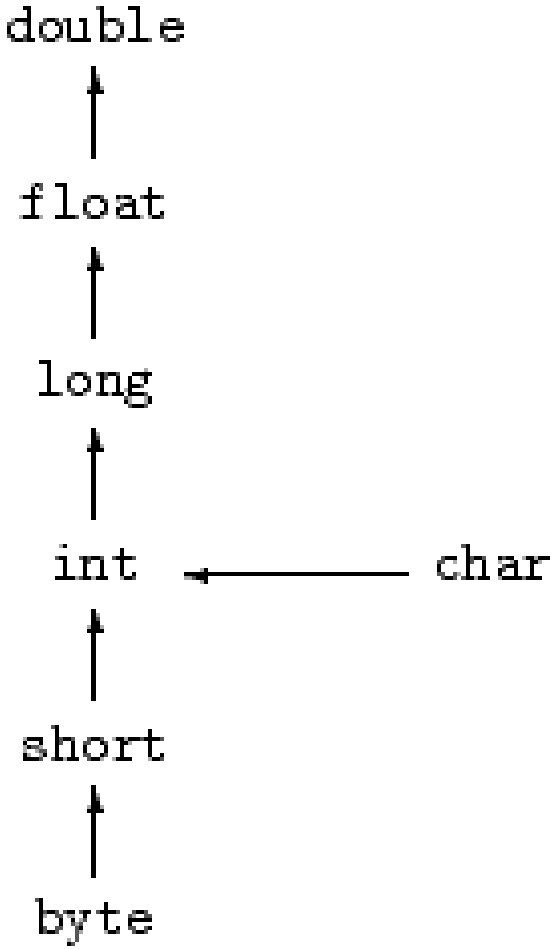


Should be of the same type. Casting: Var=(TypeOfVar)Expression;

```
double x=2; // implicit casting
double x=(double)2;// explicit
double x=2.0; // same type
```

Typing:
Safeguards for basic bugs in programs
Allows one to perform *static analysis of programs*

Implicit casting




```
char c='X';
int code=c;
System.out.println(code);
```

Answers 88 (ASCII code of X)

Implicit casting rules

Fundamentals of Java: Types

- *Everything* is typed (variables, constants, etc.)
- **Require** to declare types of variables
- The result of an expression has a type
- Variable and expression should have the **same type** for assignment

```
1  public class types
2  {
3
4  public static void main(String [] args)
5      {
6
7  double a,b;
8  float c,d;
9
10 int A,B;
11 boolean bool;
12
13
14 a=3.1415;
15 b=2.71;
16
17 c=1.3f;
18 d=5.0;
19
20 A=65556;
21 B=4*A;
22
23 bool=true;
24
25
26 bool=B;
27
28
29         }
30
31
32 }
```

ERROR

possible loss of precision | types.java | D:\Enseignements\INF311\Lectures2008 line 18

(d=5.0f)

ERROR

incompatible types | types.java | D:\Enseignements\INF311\Lectures2008 line 26

(different types)

Compiler warns you of implicit casting (possible loss of precision!)

Recap of simple (formula) programs

Declare variables of basic types: **Type var;**

```
double x;
int n,m; //separate with a comma variables
char c;
```

Assignment: **var=Expression;**

```
x=2.71;
n=2008;
c='X';
```

Arithmetic expression: **Expression1 Operator Expression2**

```
m=300%23;
delta=b*b-4*a*c;
```

Declare+Assign at once (shortcut):

```
int year2secs=365*24*60*60;
```

Incrementing/Decrementing

```
x=x+1;
x=x+step;
// Instructions equivalent to
x+=1;
x+=step;
// Decrement now
x-=3;

i=2;
i++; // equivalent to i=i+1;
++i; // similar, equivalent to i=i+1;
```

Incrementing is useful for loops

Pre- and post-incremention

compare...

```
i=5;
j=i++; // post-incrementation

ii=5;
jj=++ii; // pre-incrementation
```

Var++ returns the value of var and then increment it
++Var *first* increment var and then return its value

Thus j=5 but jj=6

Chopping Programs (Language)

Syntax of programs

Word		Reserved keywords Variables
Sentence		Instruction I;
Paragraph		Block (of instructions) {I;}
Chapter		Function
 Book		Program
 Library		Library (API)

Commenting programs

- Adopt conventions
  Eg., class ClassName .... stored in file ClassName.java

- Name variables explicitly (so that you can remember them easily)

- Comment programs (single line // or multiple lines /* */)

```
// Written for INF311

class CommentProgram
{
    /* This is a simple Java program that
     illustrates how to comment source code */


     // Entry of the program

     public static void main(String[ ] args)
     {// it does nothing
     }
}
```

A basic skeleton program in Java

```
// Basic skeleton program for INF311

class Prog
{
   public static void main(String[] arg)
   {
   int x=2008;
   System.out.println(x);
   }
}
```

Name of your program: Prog.java

Magic formula 1

Magic formula 2

```
> javac Prog.java
```

(builds a Prog.class file)

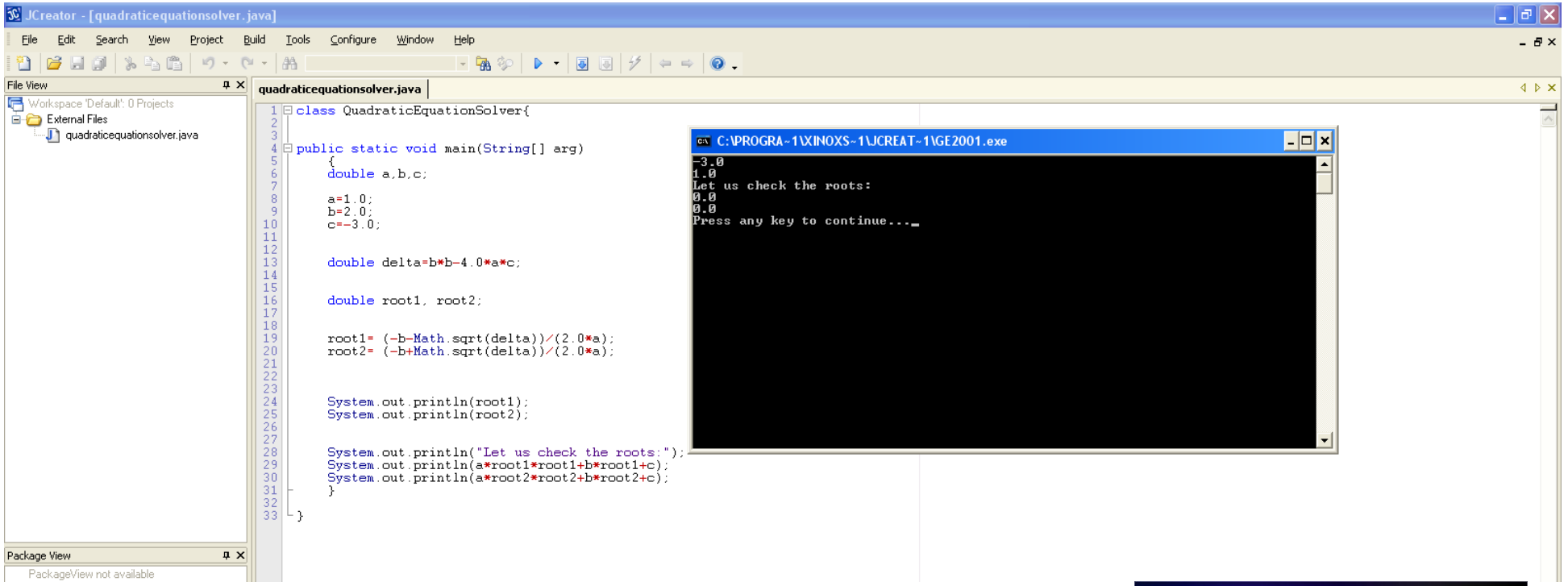
```
> java Prog
```

(execute the program)

```
2008
```

Integrated Development Environment (IDE)

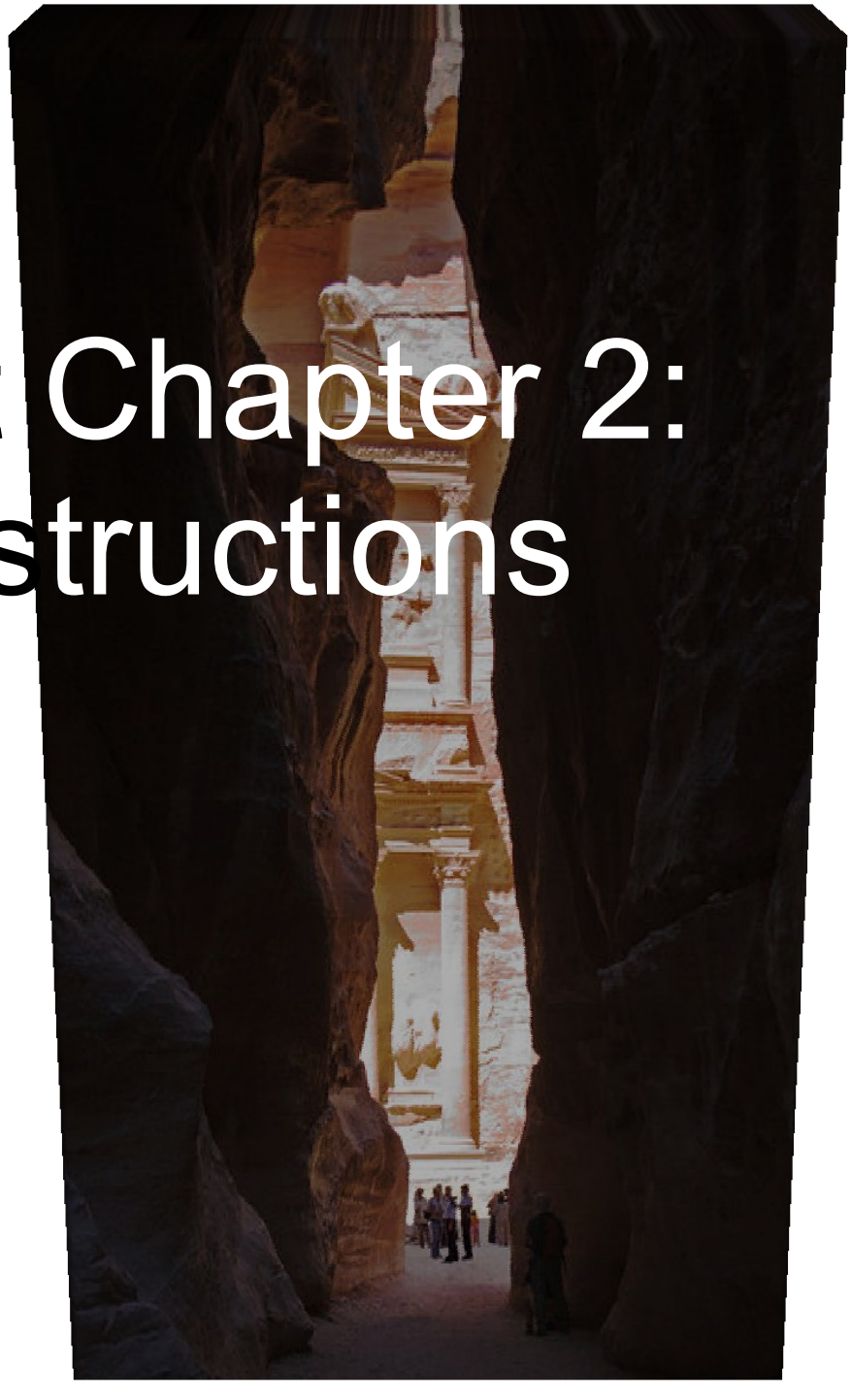
An IDE allows one to create, edit, compile and debug *seamlessly* applications at the tip of mouse clicks.



(Eg., Jcreator, www.jcreator.com/)

Eclipse

A Glimpse at Chapter 2: Block of instructions



Euclid's Greatest Common Divisor (GCD)

Input: Two numbers **a**,**b**
Output: Find the *greatest common divisors* **c** of **a** and **b**

Euclid's original algorithm

```
while b ≠ 0
    if a > b
        a := a - b
    else
        b := b - a
return a
```

30=2*5*3
105=7*5*3

For example, GCD of (30,105):

Mathematical proof:
GCD(30,105)
=GCD(30,75)
=GCD(30,45)
=GCD(30,15)
=GCD(15,15)
=GCD(15,0)

=> GCD(30,105)=15

History, proof, etc http://en.wikipedia.org/wiki/Euclidean_algorithm

Euclid's Greatest Common Divisor (GCD)

Input: Two numbers **a**,**b**
Output: Find the *greatest common divisors* **c** of **a** and **b**

Euclid's original algorithm

```
while b ≠ 0
    if a > b
        a := a - b
    else
        b := b - a
return a
```

```
class GCD {
    public static void main(String[] arg)
    {
        // Parse arguments into integer parameters
        int a= Integer.parseInt(arg[0]);
        int b= Integer.parseInt(arg[1]);

        while (a!=b)
    {
        if (a>b) a=a-b;
            else b=b-a;
    }

        // Display to console
        System.out.println(a);
    }
    }
```

Euclid's greatest common divisor (GCD)

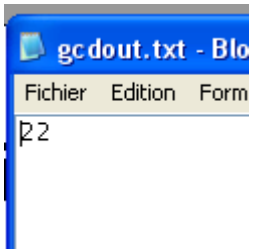
```
> javac  gcd.java
```

(compile in a gcd.class)

arg[0] arg[1]

```
> java gcd 234652 3456222    > gcd.txt
```

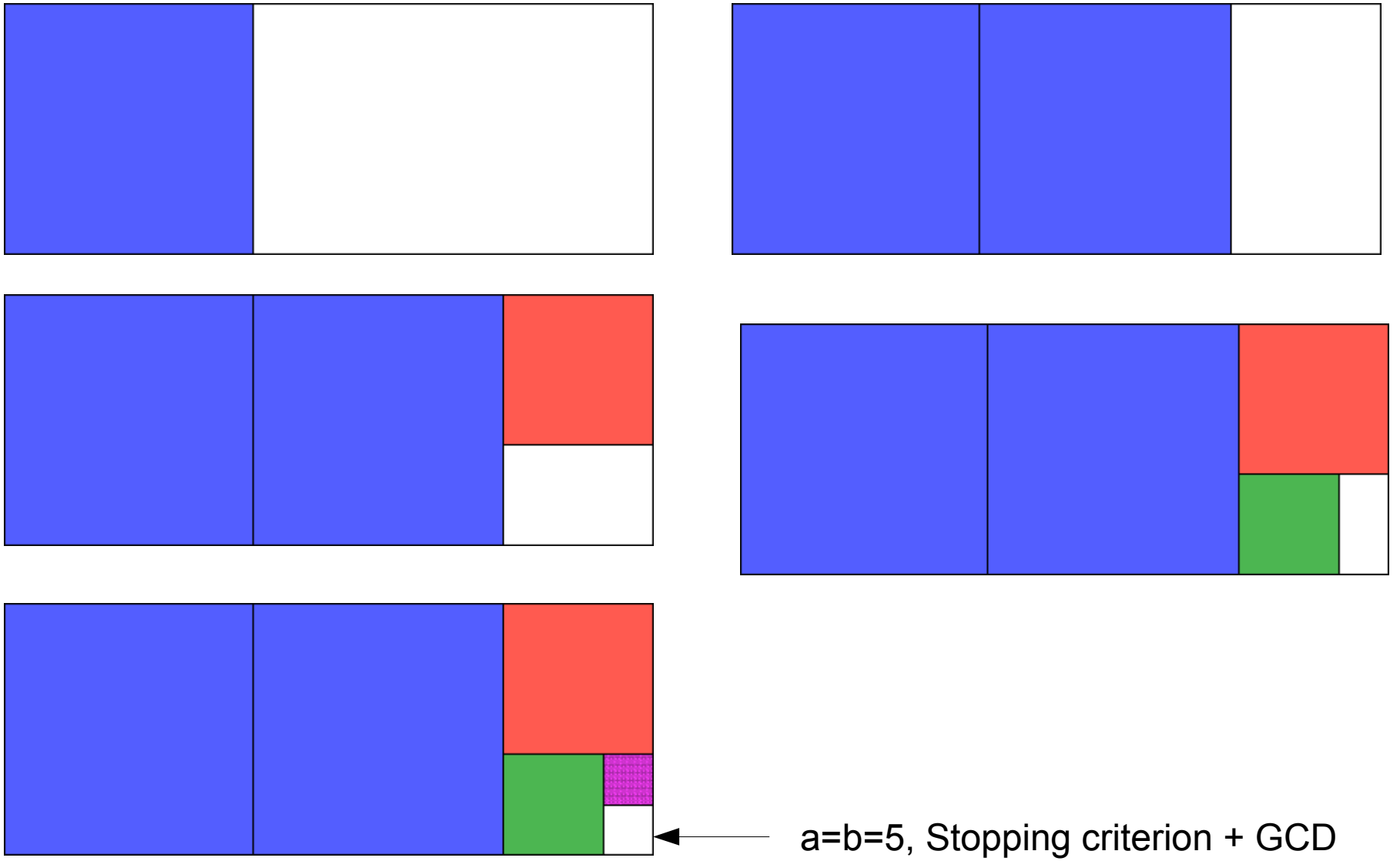
(execute and store result in gcd.txt)



Geometric interpretation of Euclid's GCD

```
while b ≠ 0
    if a > b
        a := a - b
    else
        b := b - a
return a
```

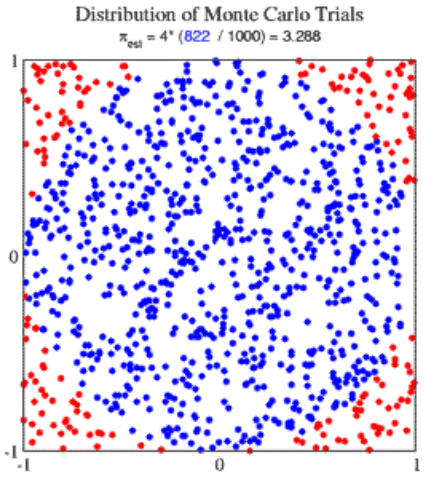
Visualize **a** (65) and **b** (25) on two axes



Programming is helpful for simulation

Simulation by Monte Carlo methods:

Eg., approaching PI=3.41592... using simulation



Draw a random point uniformly in a square:
Probability of falling inside a centered unit disk?

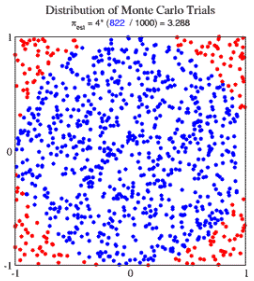
$$\frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi \cdot r^2}{(2 \cdot r)^2} = \frac{\pi}{4}$$

How do we get (pseudo-)random numbers in Java?
Call function random() of class Math

```
Math.random();
```

Monte-Carlo sampling extremely used in graphics and financial economy !!!

Monte-Carlo estimation of PI in Java




```
Estimation of PI: 3.1512 versus machine PI 3.141592653589793
Press any key to continue...
```

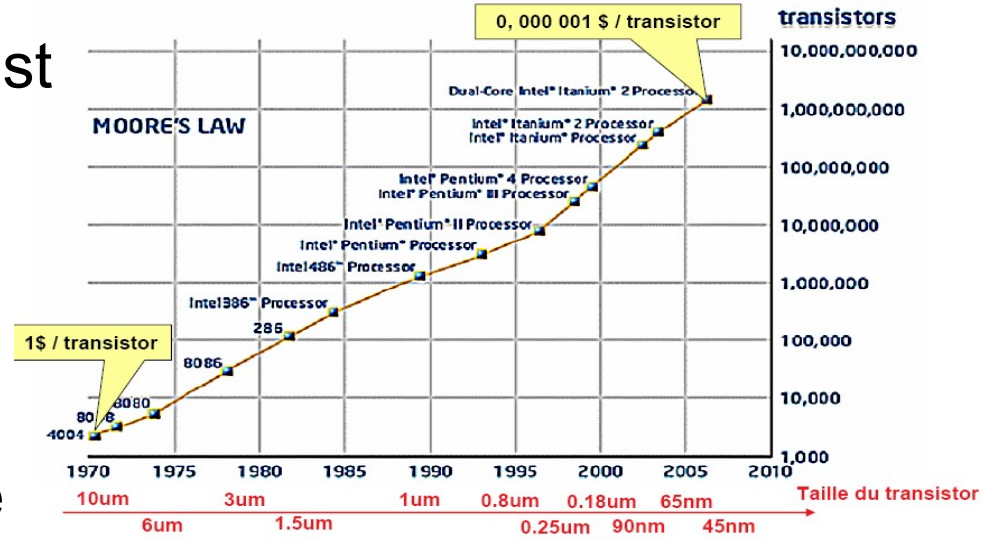
```
import java.util.*;

class MonteCarloPI
{
    public static void main(String [] args)
    {
        int iter = 10000000; // # iterations
        int hits = 0;

        for (int i = 0; i < iter; i++)
        {
            double rX = 2*Math.random() - 1.0;
            double rY = 2*Math.random() - 1.0;
            double dist = rX*rX + rY*rY;
            if (dist <= 1.0) //  falls inside the disk
                hits++;
        }

        double ratio = (double)hits/iter; // Ratio of areas

        double area = ratio * 4.0;
        System.out.println("Estimation of PI: " + area+ " versus
machine PI "+Math.PI);
    }
}
```

Monte-Carlo simulation techniques proved useful in computational sciences

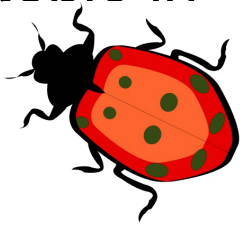
Human versus Machine

#transistors x2 every 18 months

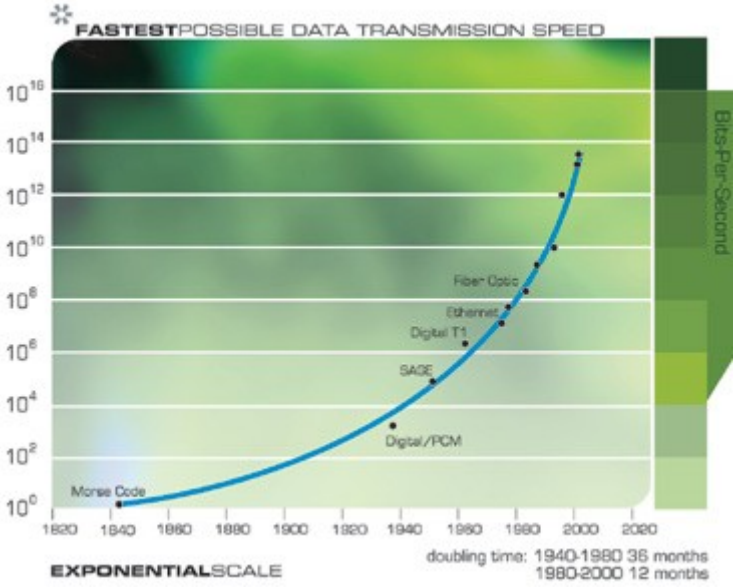
- Machines are dull but *extremely* fast
- Designing software **is difficult**
  (as difficult as building an Airbus)
- Artifical intelligence (AI) is a
  key topic in Computer Science



Bug:



- Abnormality of the system
- Not by the faulty machine but by the programmer!
- Small bugs, big consequences!!!
  (Ariane 501, Intel's Pentium division bug, etc.)
- Cost 100 billion$/ years (Dept. of Commerce)

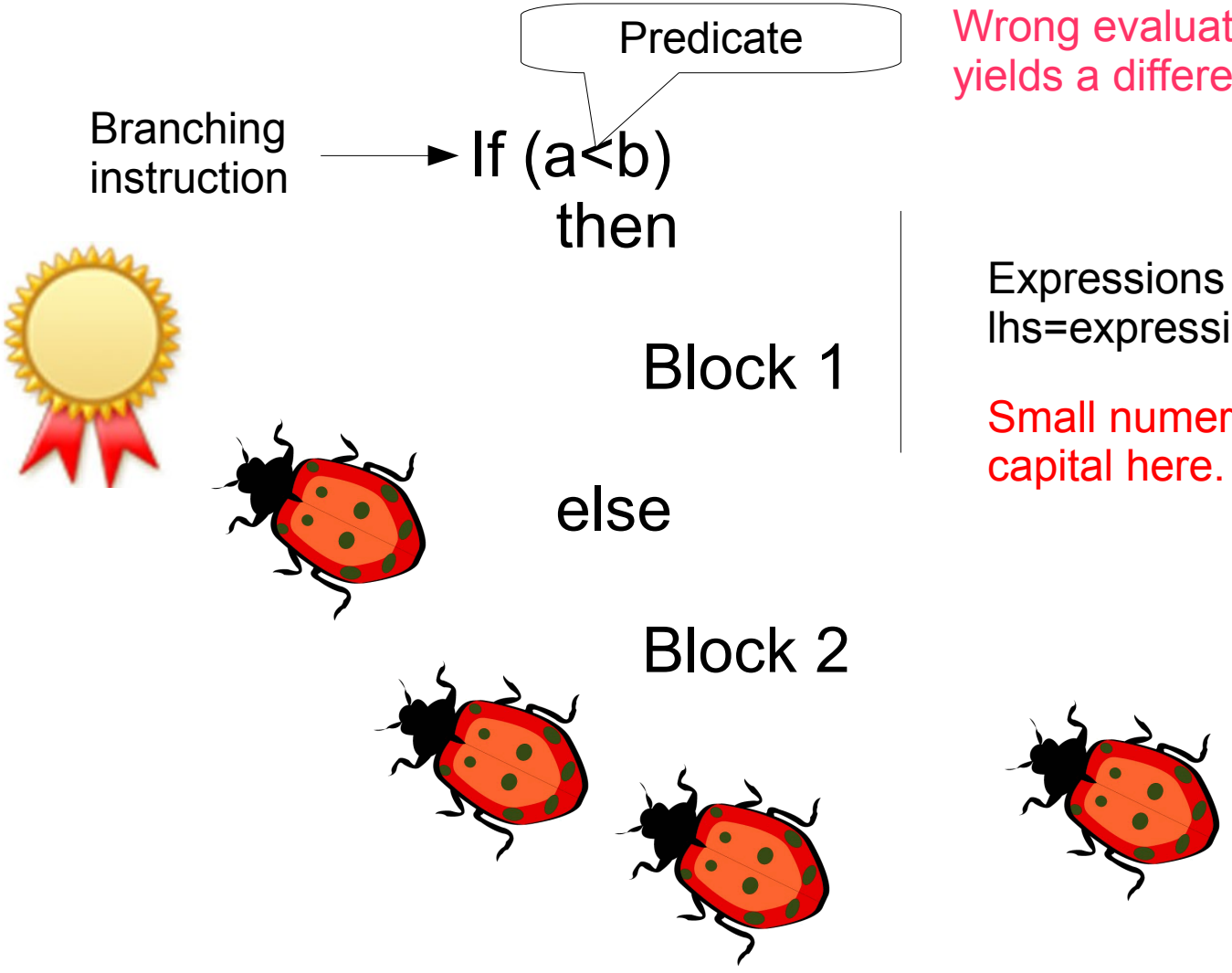


The Law of Accelerating Returns of Ray Kurzweil

http://www.kurzweilai.net/articles/art0134.html?printable=1

Small bugs, big consequences: Numerical errors

Finite precision, roundings of arithmetic operations may cause devastating effects

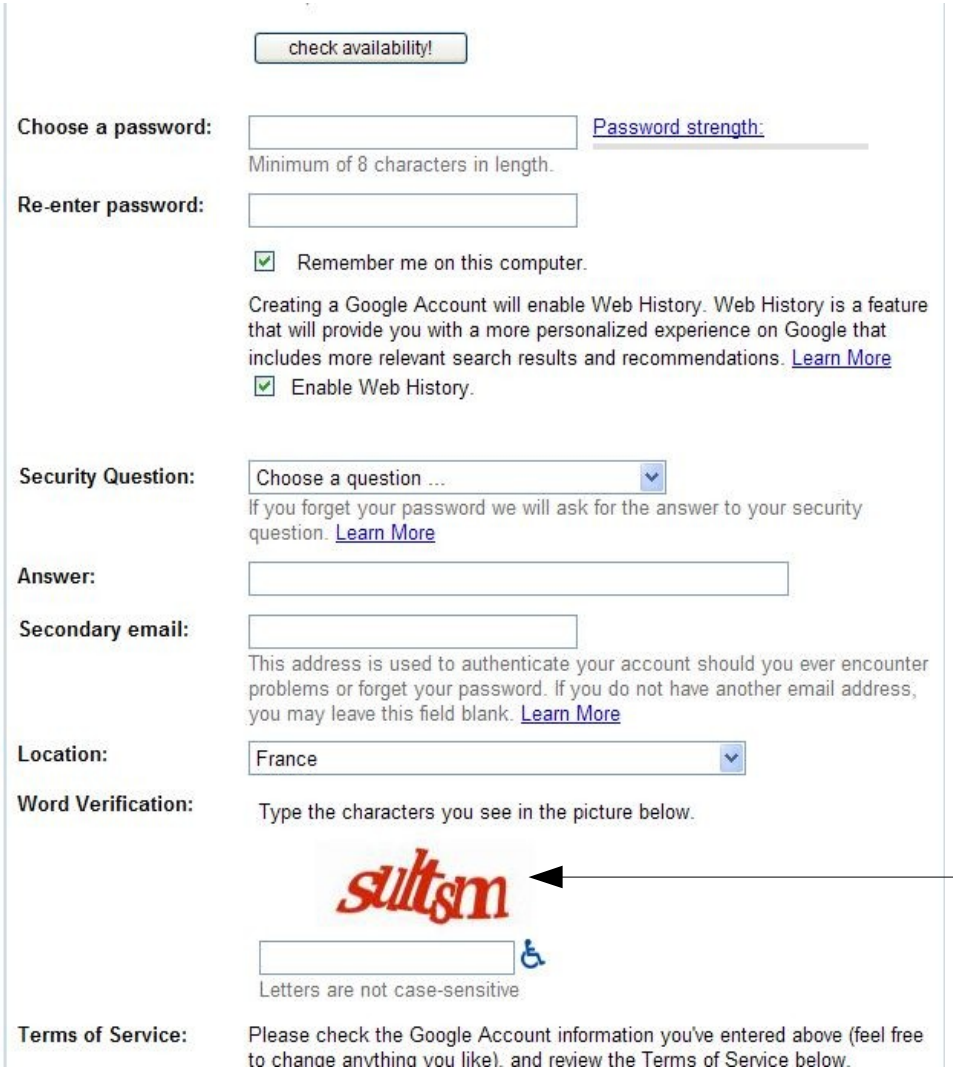


Wrong evaluation of a **predicate** yields a different path of instructions: Bug!

Expressions
lhs=expression(rhs)

Small numerical errors may not be so capital here.

CAPTCHA versus SPAM (Human vs Machine)



To fight undesirable bulk spam, we need to differentiate whether it is the action of a human or an automated jam program.

Image-recognition CAPTCHAs:
Difficult task (OCR, segmentation, etc.)

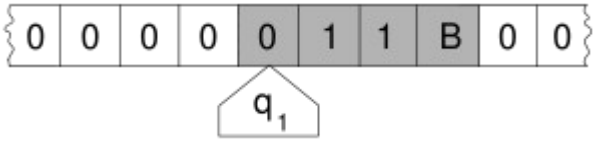
(visual) CAPTCHA

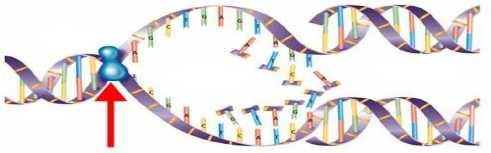
Completely Automated Public Turing test to tell Computers and Humans Apart

Turing test...

Alan Turing, 1912-1954
(41 years old)

Pioneer of modern computer science





DNA, ribosome

Proposed the "universal" Turing machine:
A *ribbon*, a *head*, a *state* and an *action table* (automaton)

Turing test: proposal for a test of machine's capability to demonstrate intelligence. Originally, for natural language conversation (and processing). Initially, by text-only channel such a teletype machine

Association for computing machinery (ACM)'s Turing Award (250000$)
[Nobel prize in computer science]

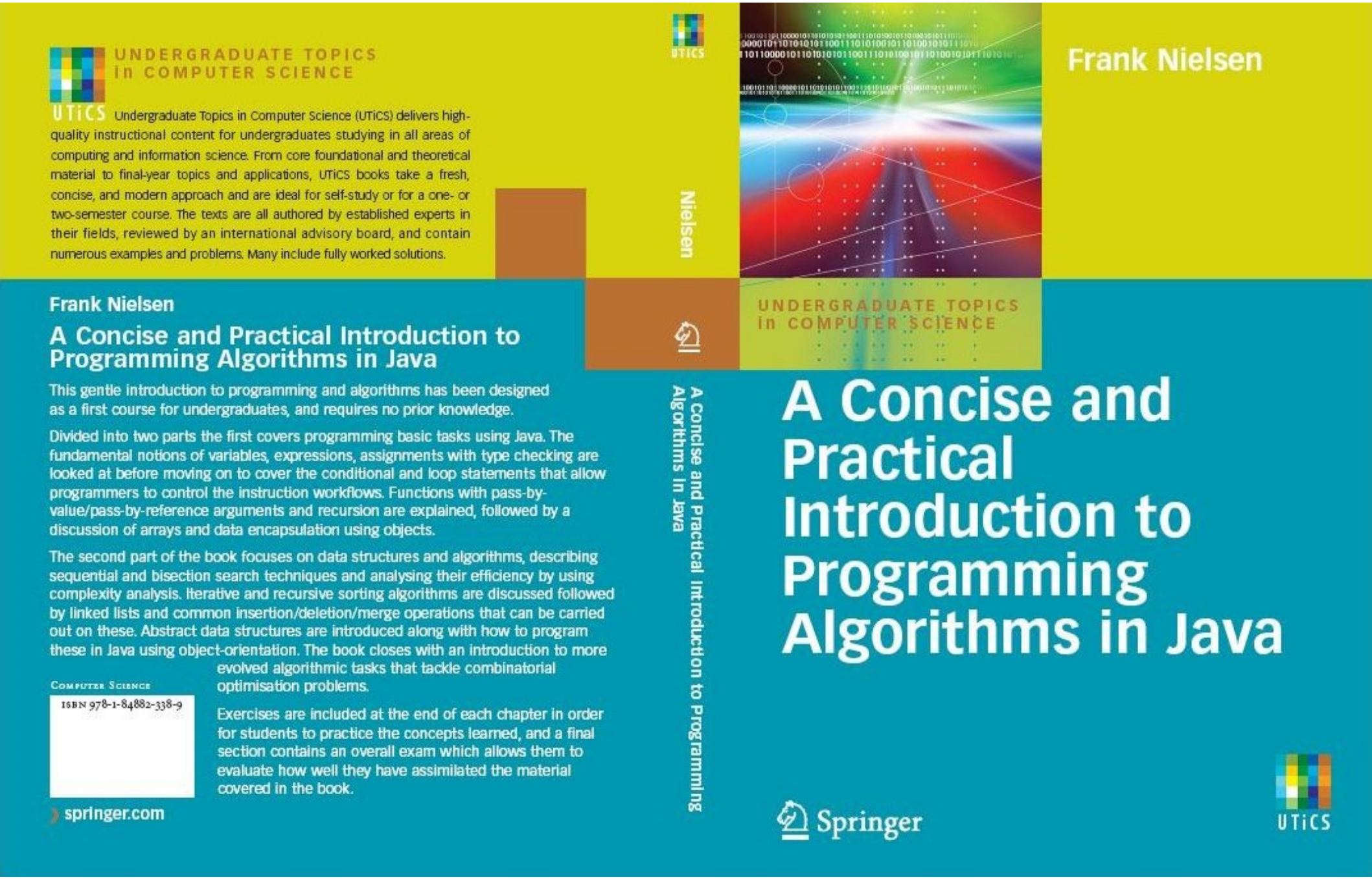
Versatility of Turing tests

:: Question

Can we make the distinction between music played by a human and music played by a machine ?

The Continuator of F. Pachet (Sony CSL)

www.csl.sony.fr


UNDERGRADUATE TOPICS
in COMPUTER SCIENCE
UTiCS
Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.
Frank Nielsen
A Concise and Practical Introduction to Programming Algorithms in Java
This gentle introduction to programming and algorithms has been designed as a first course for undergraduates, and requires no prior knowledge.
Divided into two parts the first covers programming basic tasks using Java. The fundamental notions of variables, expressions, assignments with type checking are looked at before moving on to cover the conditional and loop statements that allow programmers to control the instruction workflows. Functions with pass-by-value/pass-by-reference arguments and recursion are explained, followed by a discussion of arrays and data encapsulation using objects.
The second part of the book focuses on data structures and algorithms, describing sequential and bisection search techniques and analysing their efficiency by using complexity analysis. Iterative and recursive sorting algorithms are discussed followed by linked lists and common insertion/deletion/merge operations that can be carried out on these. Abstract data structures are introduced along with how to program these in Java using object-orientation. The book closes with an introduction to more evolved algorithmic tasks that tackle combinatorial optimisation problems.
Exercises are included at the end of each chapter in order for students to practice the concepts learned, and a final section contains an overall exam which allows them to evaluate how well they have assimilated the material covered in the book.
Computer Science
ISBN 978-1-84882-338-9
springer.com
UTiCS
Nielsen
A Concise and Practical Introduction to Programming Algorithms in Java
Frank Nielsen
UNDERGRADUATE TOPICS
in COMPUTER SCIENCE
A Concise and Practical Introduction to Programming Algorithms in Java
Springer
UTiCS