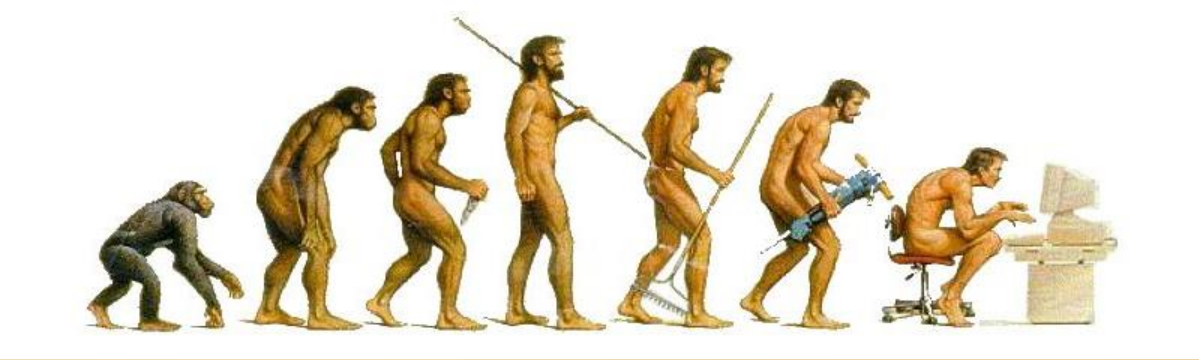
Lecture 1-1

Introduction to Computer Science:
Course Overview



Course objectives

The course will give you …

- Basic exposure to Computer Science
- Basic programming skills

In addition, you will …

- Sharpen your analytic skills
- Appreciate clarity and elegance
- Develop a taste for beauty in science and engineering
- Learn how to learn and develop.

Course requirements

Attend:

- Two weekly lectures (שיעור)
- One weekly recitation (תרגיל)
- One weekly workshop (סדנה)

Submit:

- A weekly homework assignment (שיעורי בית)

Course Website (“Moodle”)

Contents

- Lectures / recitations slides + code
- HW documents + code
- Course announcements

Important:

The only formal and abiding channel of communications between the course team and the students is **the course web site**

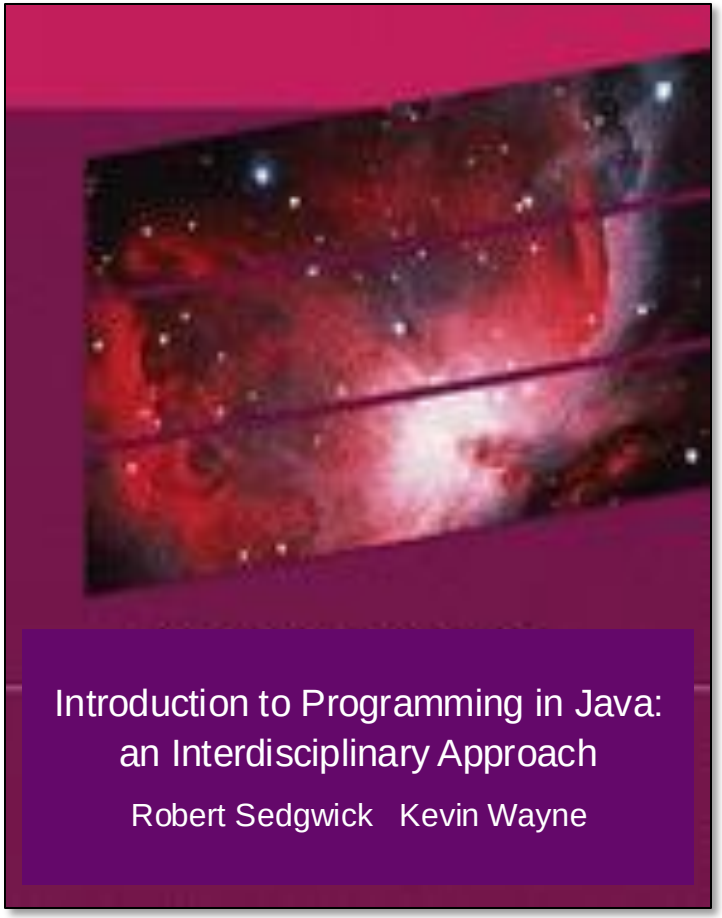
What is said orally by the course staff about HW, exams, requirements, grading, deadlines – is **tentative**. What is written in the web site is **formal**.

Q&A Forums

How to ask questions

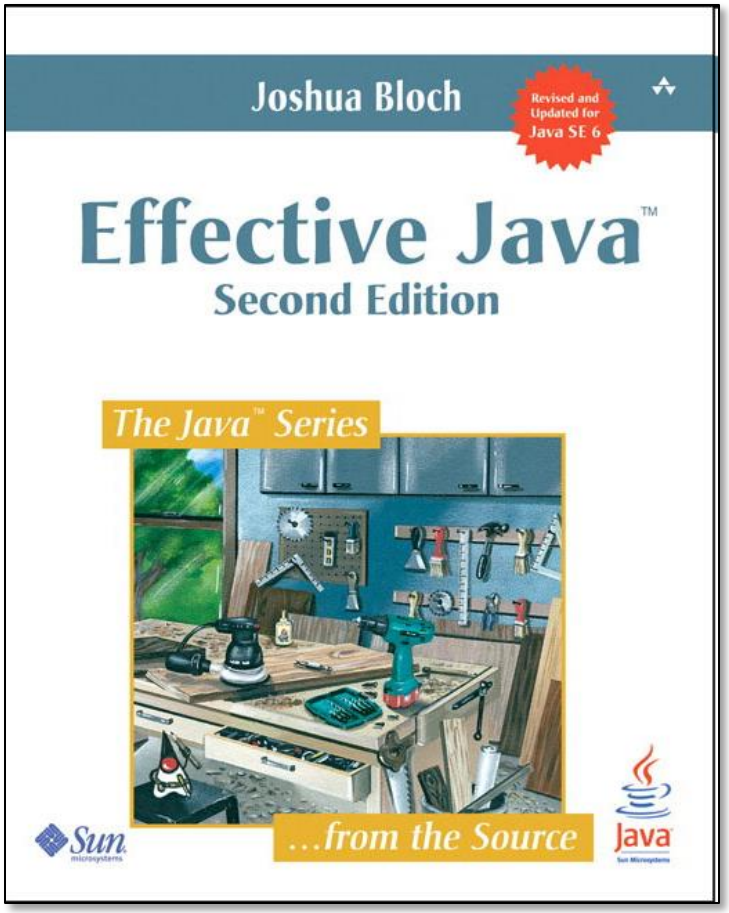
- Find the relevant forum, by week
- Read existing posts (questions, answers)
- If needed, post a question
- You'll get an answer within a few hours, from a TA or from another student
- Feel free to answer questions, if you think that your answer will be useful
- Avoid clutter, keep the channel clean
- Use English (whatever is your language level – let's practice!)

Textbooks



Recommended textbook

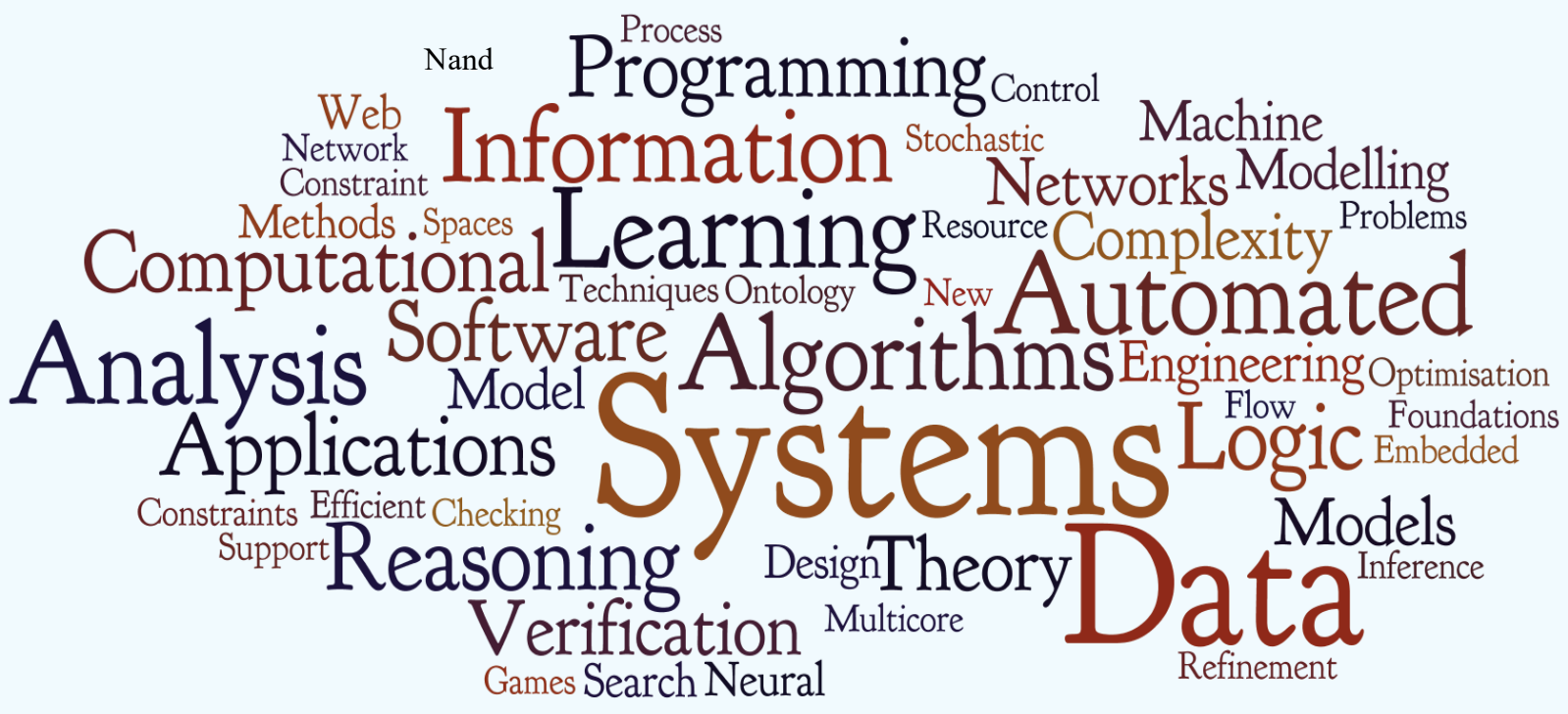
(any edition is fine)



For students with Java experience

(a classic)

Computer science is about …



Computer science is about …

Theory

- Algorithms
- Data structures
- Complexity
- Artificial intelligece

. . .

Systems

- Operating Systems
- Compilers
- Networks
- Security

. . .

Applications

- Image processing
- Molecular biology
- Robotics
- Finance

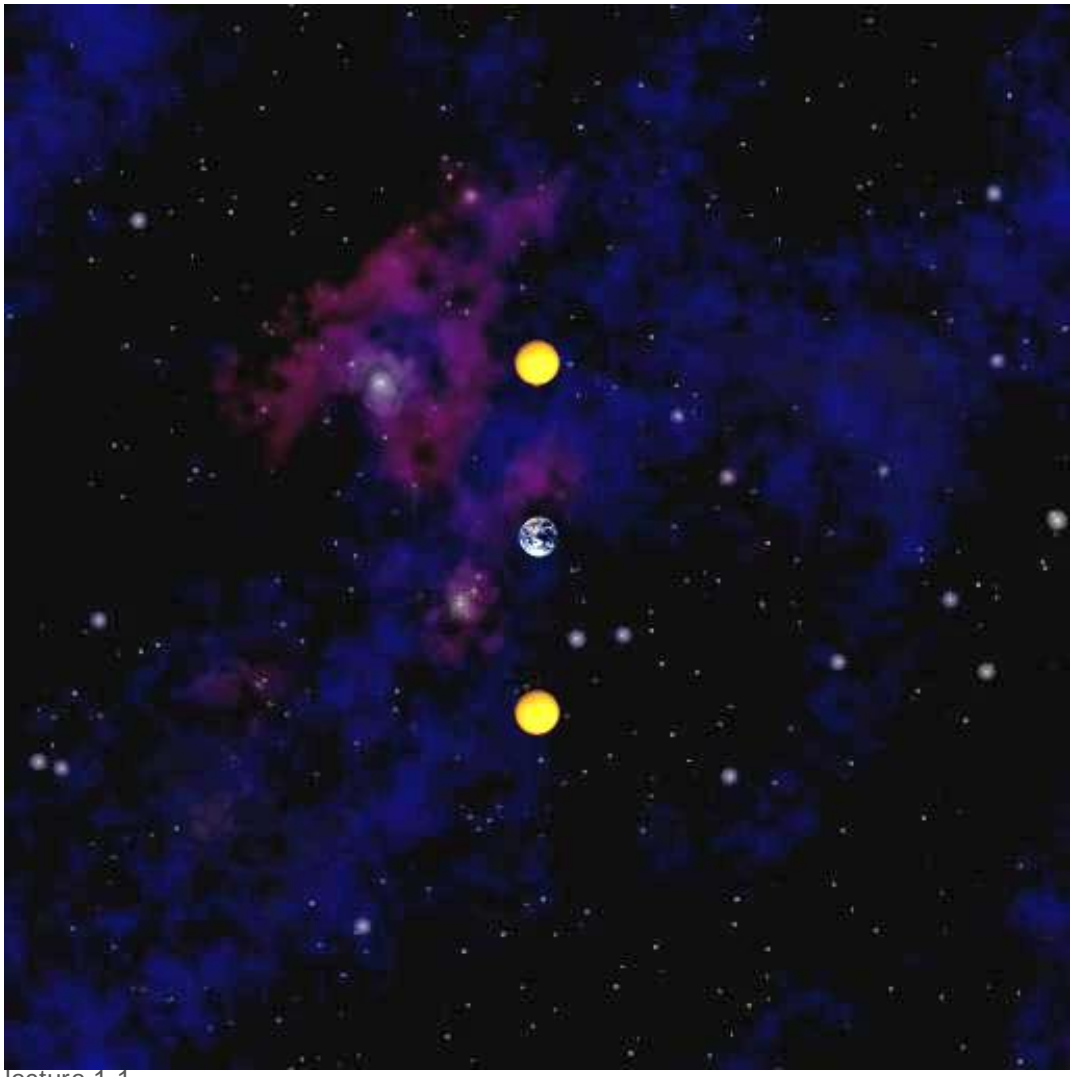
. . .

Common themes

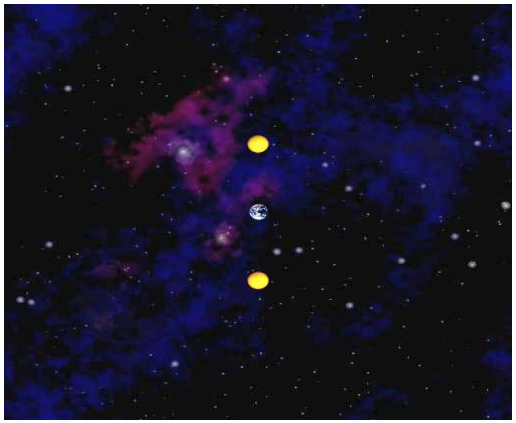
- Programming
- Efficiency
- Beauty (acquired taste)

A taste of programming

Simulate the motion of *N* heavenly bodies,
subject to Newton's laws of motion and gravity



Programming is about . . .



Problem solving

Simulation, imaging, medical systems, e-commerce, social networks, chatbots, ...

Packaged software

Apps, games, tools, ...

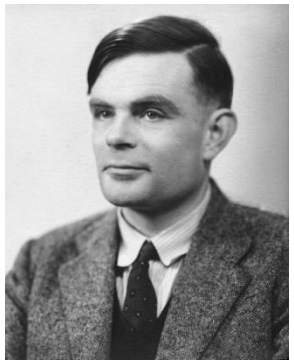
System software

Operating systems, compilers, networks, cloud, security, ...

Can programming help solve any possible problem / need?

- No, computers and programming have inherent limitations
- Stay tuned.

Ada Lovelace



Alan Turing

Programming is about . . .

Writing code that is:

- Correct
- Efficient
- Readable

And is:

- Easy to understand
- Easy to test
- Easy to maintain
- Easy to extend
- Pleasure to work with

"Instead of imagining that our main task as programmers is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do." – Donald Knuth

Programming is about . . .

Writing code that is:

- Correct
- Efficient
- Readable

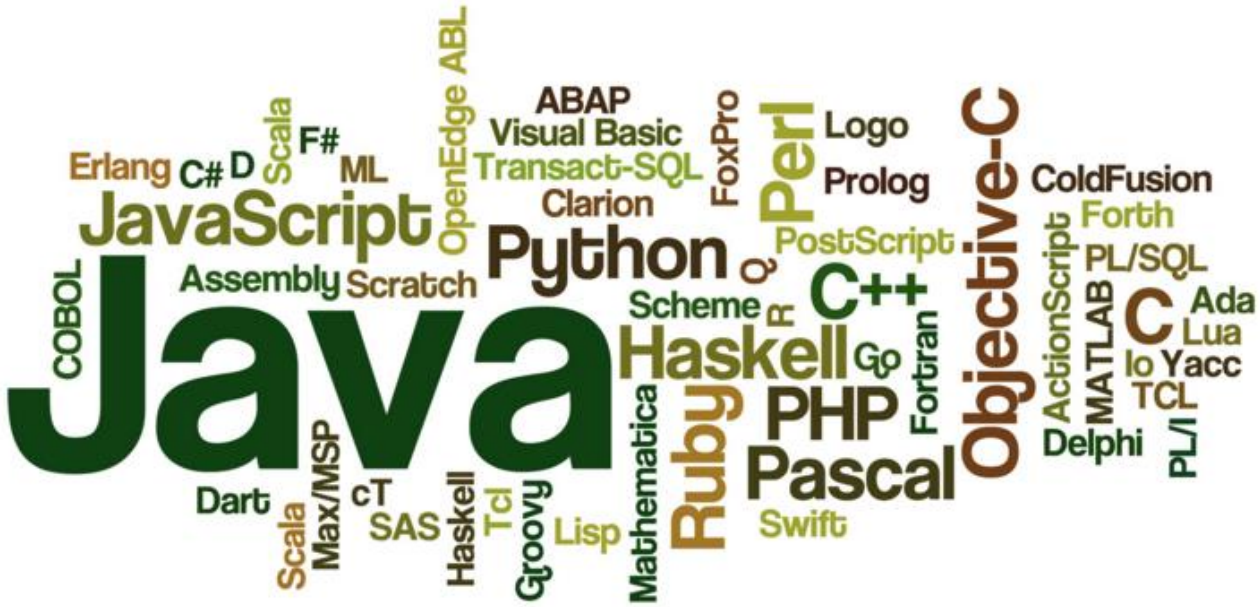
And is:

- Easy to understand
- Easy to test
- Easy to maintain
- Easy to extend
- Pleasure to work with

"Instead of imagining that our main task as programmers is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do." – Donald Knuth

How to make a peanut butter sandwich...

Programming languages



Which language to learn?

- Java
- Python
- C
- Haskell

…

Programming languages

Why Java?

- Widely used
- Widely available
- Powerful, elegant, multi-platform
- Addresses numerous needs
- Excellent software development tools
- Our school legacy

Java Applications

- Android
- Google docs
- Netflix
- Spotify
- LinkedIn
- Amazon

. . .

Programming languages

Facts of life

- There is no perfect programming language
- We need to choose *some* language

Our approach

- Teach a **subset** of Java
- Develop **general programming skills** applicable to any software development task
- Build a foundation that allows learning any other language quickly

It's not about the language!

"There are two kinds of programming languages: those that people always complain about, and those that nobody uses."

– Bjarne Stroustrup (father of C++)

Java program example

Task: Print the numbers 0 to 5

Algorithm

```
i = 0
while (i < 6)
   print i
   i = i + 1
```

Pseudocode

Java implementation

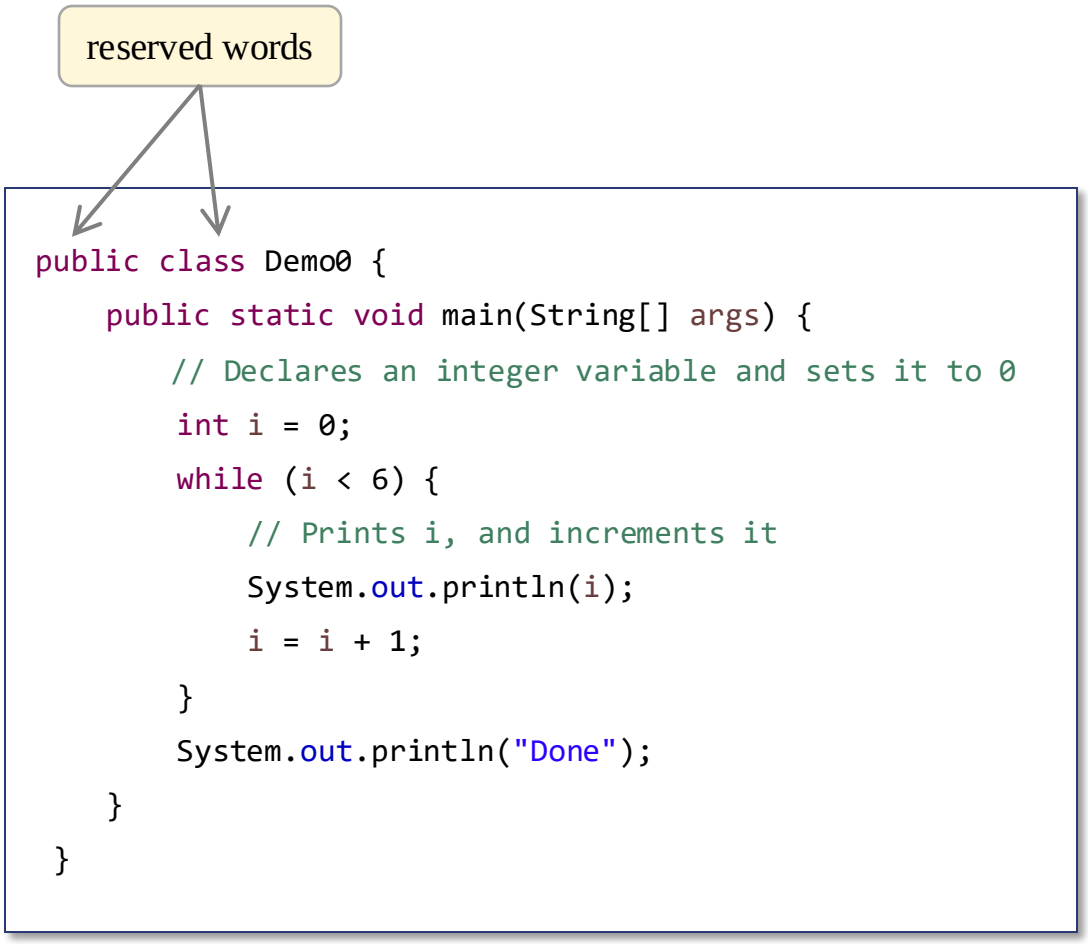
```
public class Demo0 {
    public static void main(String[] args) {
        // Declares an integer variable and sets it to 0
        int i = 0;
        while (i < 6) {
            // Prints i, and increments it
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
 }
```

Java code

Java syntax elements (first approximation)

"Words":

- reserved words
- identifiers (user-defined)



Java syntax elements (first approximation)

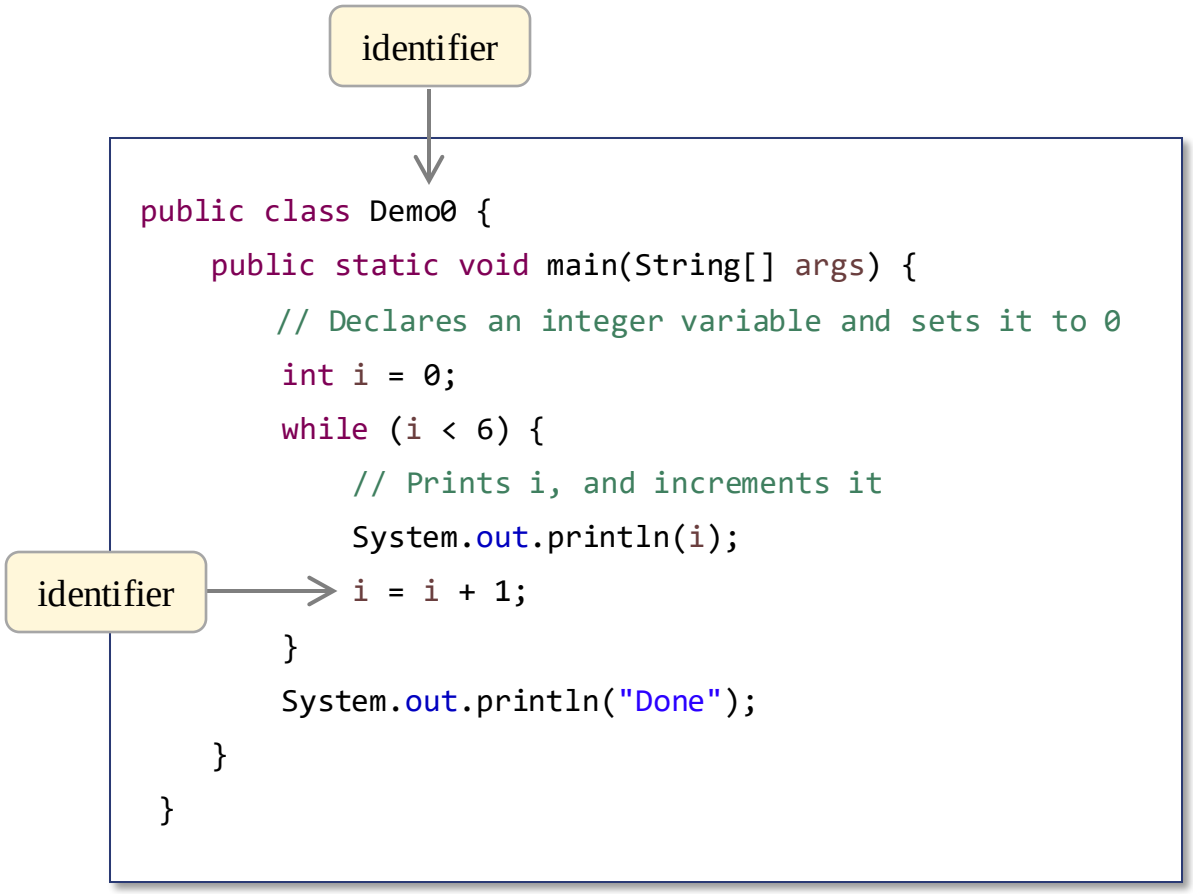
Java reserved words:

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strict	volatile
const	float	native	super	while

Java syntax elements (first approximation)

"Words":

- reserved words
- identifiers (user-defined)



Java syntax elements (first approximation)

"Words":

- reserved words
- identifiers (user-defined)

Literals (constants):

- numbers
- strings

```
public class Demo0 {
    public static void main(String[] args) {
        // Declares an integer variable and sets it to 0
        int i = 0;
        while (i < 6) {
            // Prints i, and increments it
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

numeric literals (like 0, 6)

String literal

Java syntax elements (first approximation)

"Words":

- reserved words
- identifiers (user-defined)

Literals (constants):

- numbers
- strings

Symbols:

( ) [ ] { } , . ; + - * / …

```
public class Demo0 {
    public static void main(String[] args) {
        // Declares an integer variable and sets it to 0
        int i = 0;
        while (i < 6) {
            // Prints i, and increments it
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
}
```

symbol

Java syntax elements (first approximation)

"Words":

- reserved words
- identifiers (user-defined)

Literals (constants):

- numbers
- strings

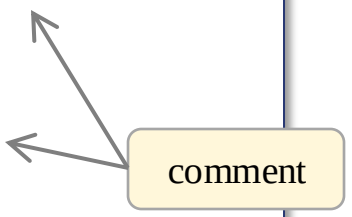
Symbols:

( ) [ ] { } , . ; + - * / …

White space

- comments
- indentation
- colors.

```
public class Demo0 {
    public static void main(String[] args) {
        // Declares an integer variable and sets it to 0
        int i = 0;
        while (i < 6) {
            // Prints i, and increments it
            System.out.println(i);
            i = i + 1;
        }
        System.out.println("Done");
    }
 }
```

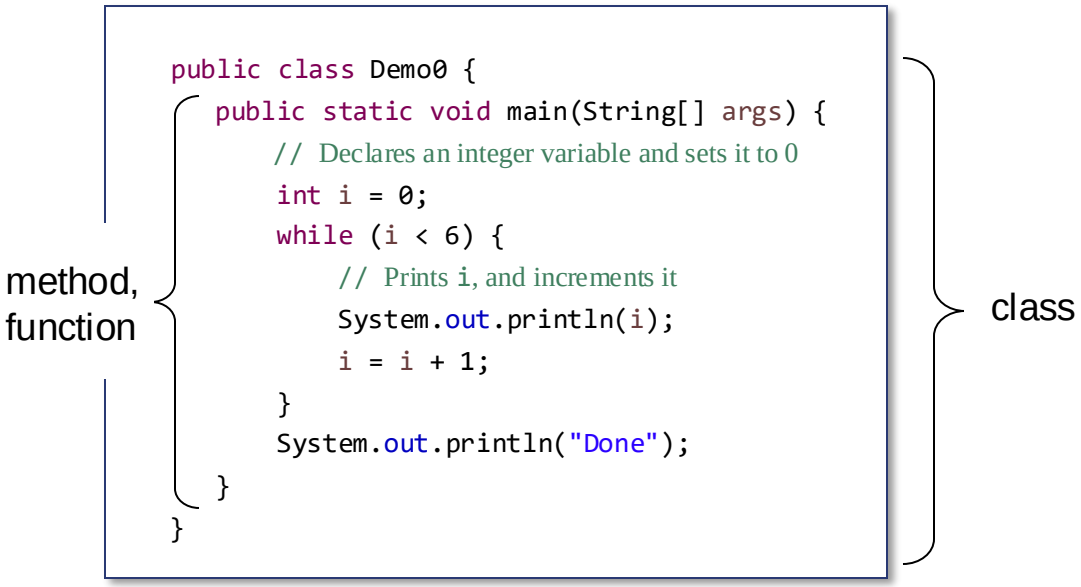


A note about the comments in this particular program:

First comment: Not necessary

Second comment: Useful, although the code is self-explanatory

Java program structure



Program (loosely defined): a collection of one or more classes

Class: a collection of one or more methods / functions

Method / function: a sequence of one or more statements.

In this course:

- We'll start writing programs that consist of one class and one method ("`main`")
- Later we will write classes that consist of several methods
- Later we will write programs that consist of several classes.

Java program structure

```
public class Demo0 {
   public static void main(String[] args) {
      // Declares an integer variable and sets it to 0
      int i = 0;
      while (i < 6) {
         // Prints i, and increments it
         System.out.println(i);
         i = i + 1;
      }
      System.out.println("Done");
   }
}
```

Same functionality

```
public class Demo0 {public static void main(String[] args){int i=0;while
(i<6){System.out.println(i);i=i+1;}System.out.println("Done");}}
```

White space

Comments, indentation, colors (ignored by the compiler)

Purpose

Used to make programs readable

Program readability and clarity are as important as program correctness (maybe more)!

"Any fool can write code that a computer understands. Good programmers write code that humans understand" – Martin Fowler

Java program structure

```
public class Demo0 {
   public static void main(String[] args) {
       // Declares an integer variable and sets it to 0
       int i = 0;
       while (i < 6) {
           // Prints i, and increments it
           System.out.println(i);
           i = i + 1;
       }
       System.out.println("Done");
   }
}
```

Aspects of any language

Semantics (what you want to say / do):
meaning / intention

Syntax (how to say it):
The rules of the language:
vocabulary and grammar

Style (how *well* you say it):
Critically important

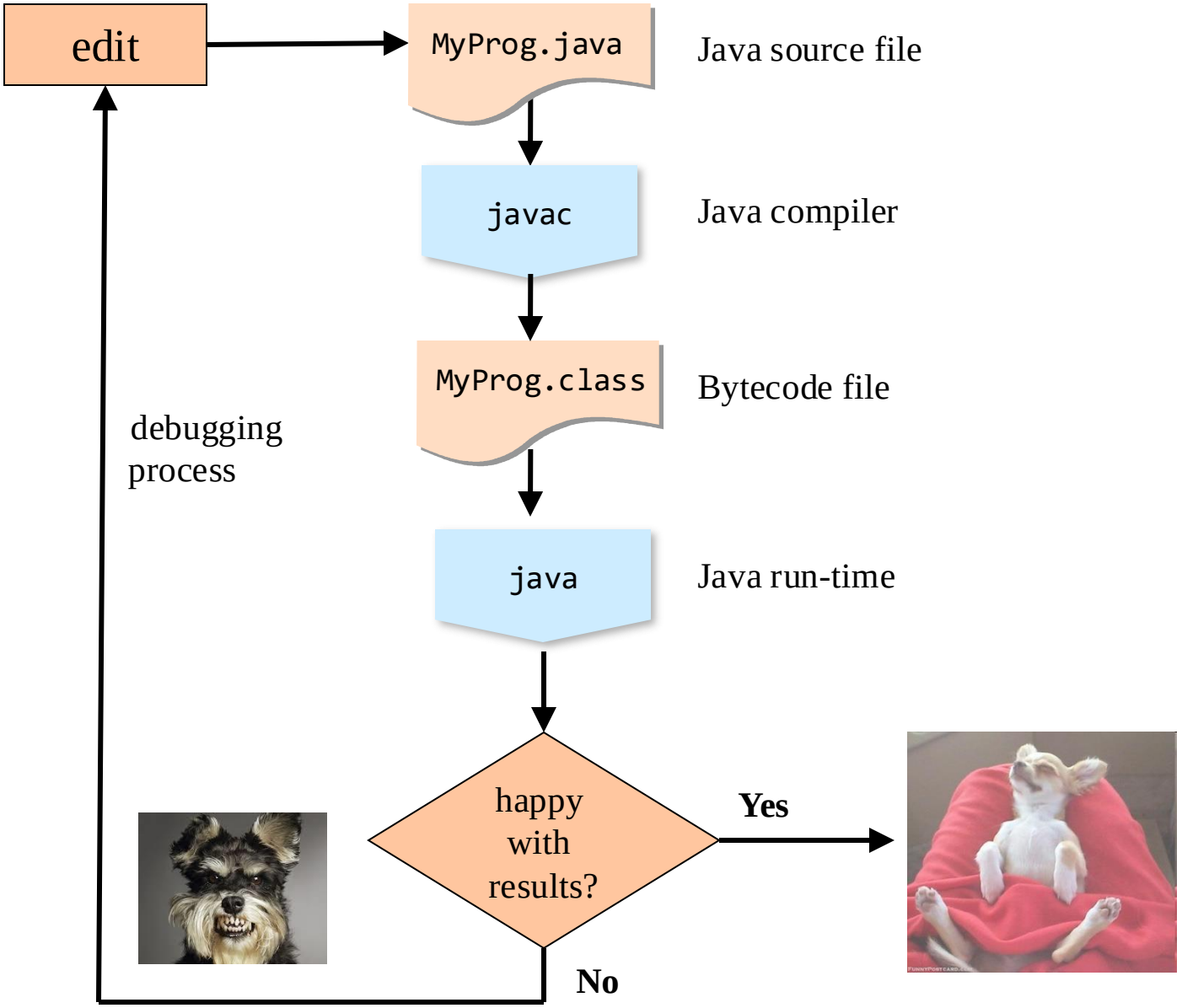
Natural languages (Hebrew, English, ...)

- Allow breaking syntax rules
- Occasionally there is more than one meaning to a sentence

Programming languages

- Syntax is sacred
- Only one semantic interpretation: No ambiguity.

Program development



Program development: “Command” / “Terminal” level

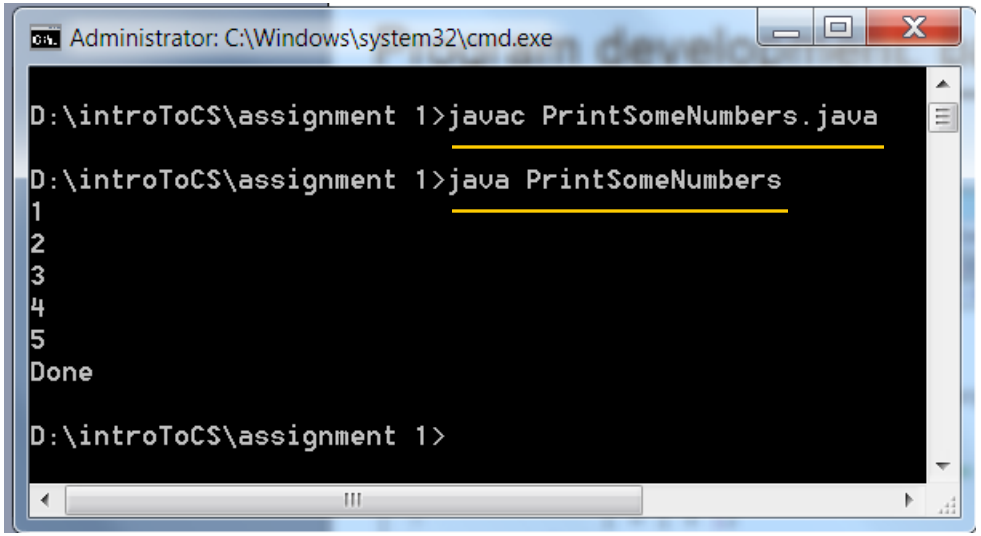
Edit (in this example: the file `Demo0.java`):

EditPlus - [D:\introToCS\assignment 1\PrintSomeNumbers.java]

File Edit View Search Document Project Tools Browser Zen Coding Windo

```
1  public class PrintSomeNumbers {
2
3      // Prints the numbers 1 to 5
4      public static void main(String[] args) {
5          int i = 1;
6          while (i < 6) {
7              // Prints the current value of i
8              System.out.println(i);
9              i = i + 1;
10         }
11         System.out.println("Done");
12     }
13 }
```

Compile and execute:



Debugging

0. Run / execute the program
1. Observe the program’s execution
2. Figure out what’s wrong
3. Use the editor to fix the code
4. Goto step 0.

Program development: Integrated Development Environments

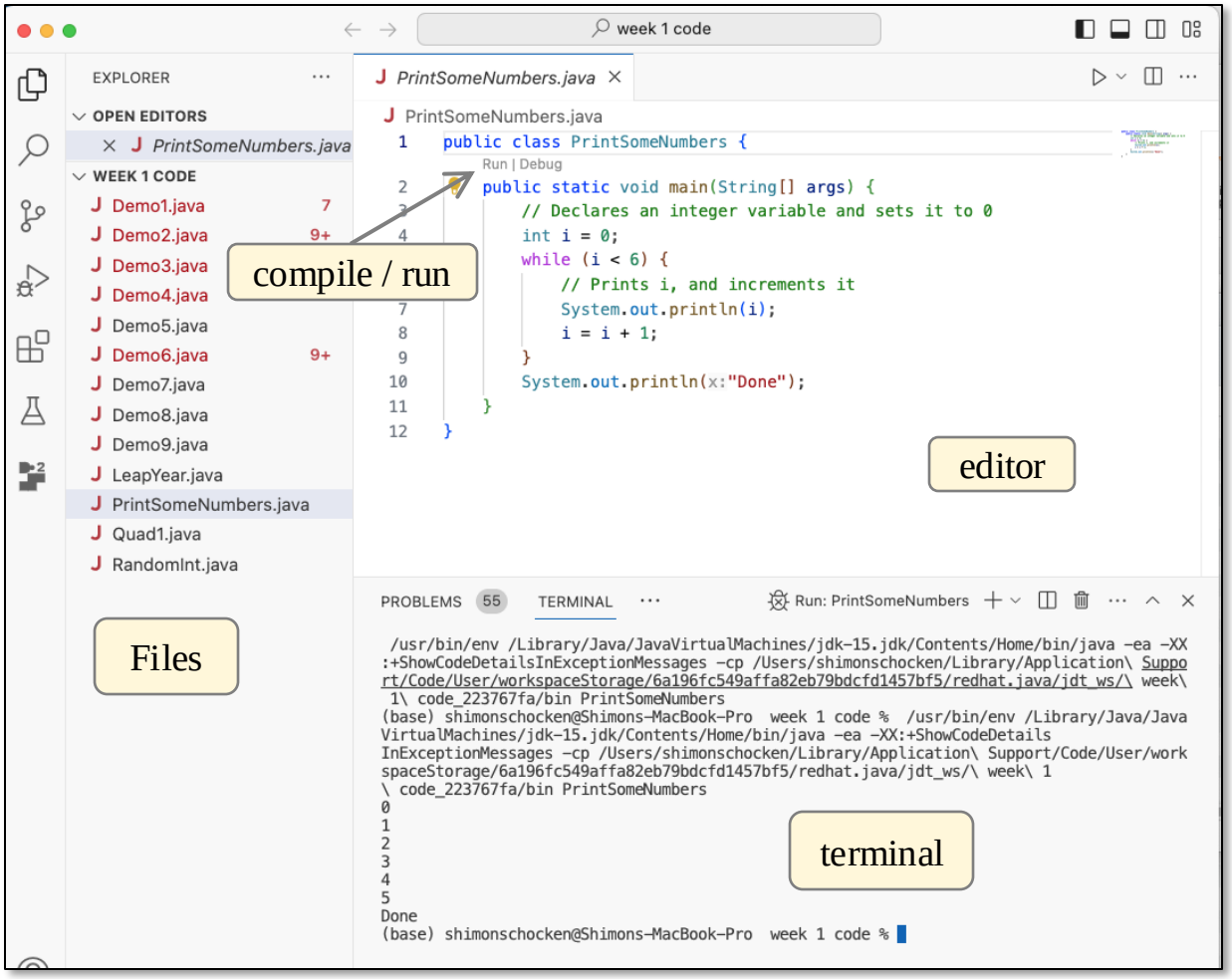
IDE: a software package featuring:

- editor (language-specific)
- compiler
- debugger
- . . .
- various dev goodies

Popular IDEs:

- VS Code
- InteliJ
- Eclipse
- NetBeans

. . .



Debugging

That's what you'll do most of the semester

Error types

- **Compile-time errors:** mostly syntax violations; detected by the compiler
- **Run-time errors:** the program passes compilation, runs, but crashes
- **Logical errors:**
  - The program runs, doing something unexpected
  - The program runs, but should be improved

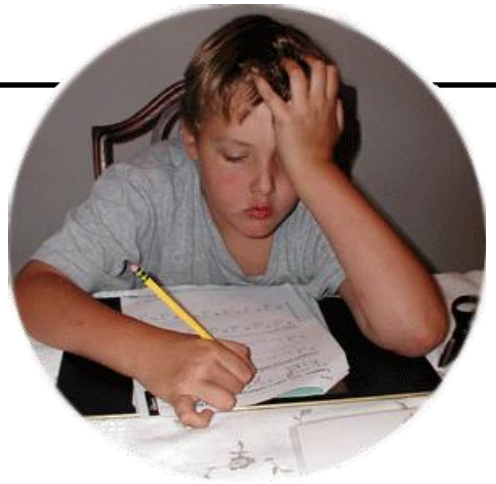
Mistakes are the portal of discovery

(James Joyce)

Anything that can possibly go wrong, will

(Murphy's Law)

Things to do



Getting started

- Visit the course website
- Install Java + VS Code on your computer (*From Zero to Code tutorial*)

Homework 1

- Play with an existing Java program
- Experience debugging
- Write a few simple programs
- Further instructions: see the course website.