Universidad
Carlos III de Madrid
www.uc3m.es

Lesson 1

Introduction

Programming

Grade in Computer Science

Outline

1. **What is *programming*?**
2. **Components of a program: data and algorithms**
3. **Creating and running programs**
4. **Programming paradigms**
5. **Introduction to the Java programming language**

Outline

1. **What is *programming*?**
2. **Components of a program: data and algorithms**
3. **Creating and running programs**
4. **Programming paradigms**
5. **Introduction to the Java programming language**

1. What is *programming*?

Definition

According to RAE:

5. tr. *Inform*. Develop programs to solve problems with computers

An informal but more elaborate definition:

Provide a **computer** with a set of **instructions** and a set of **data** on what should be done with the data for the resolution of a given problem

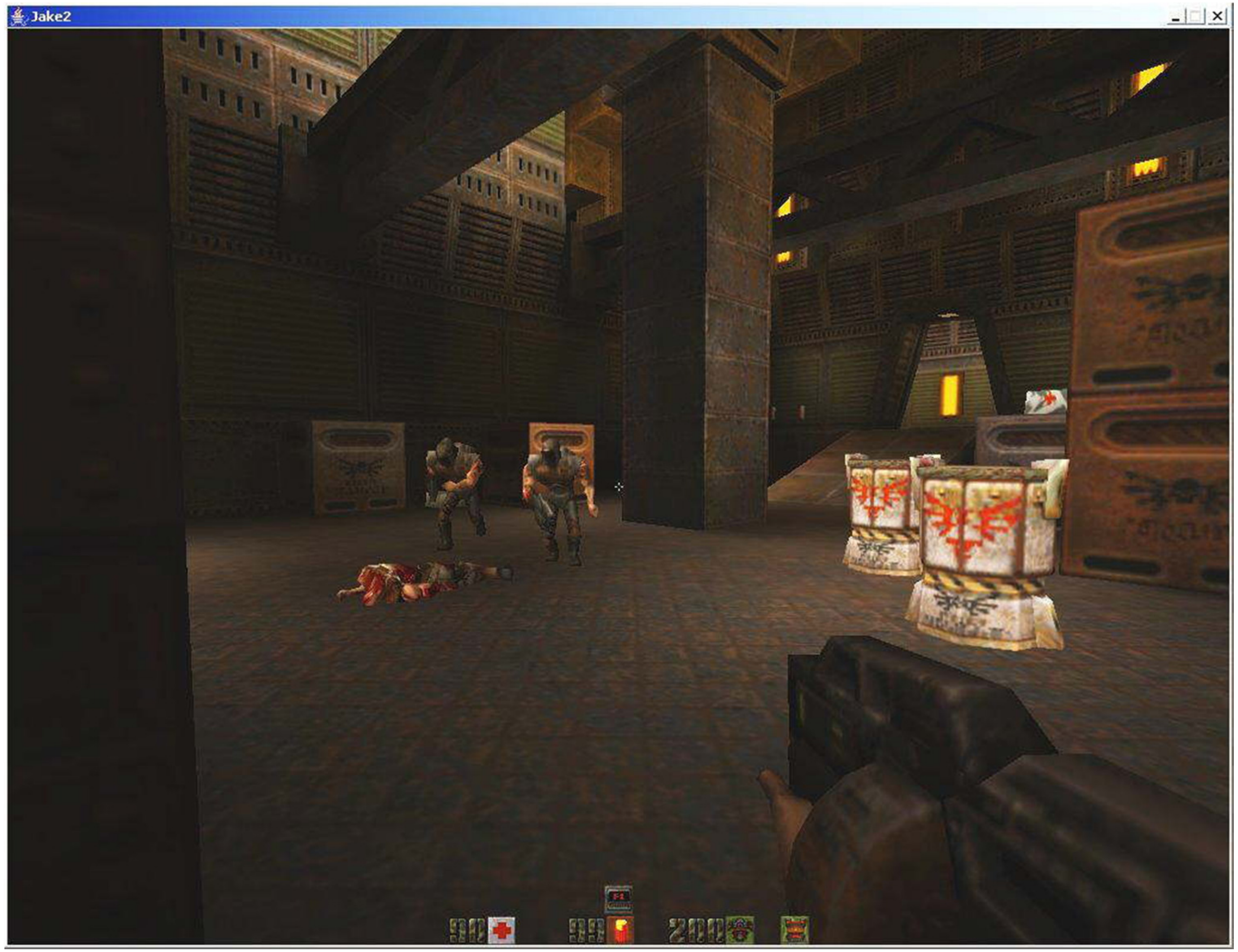
Programming encompasses several activities aimed to develop a computer program

> or *to implement* a computer program

software design
coding
compilation
running
debugging
deploying
etc.

Jake2

[link]



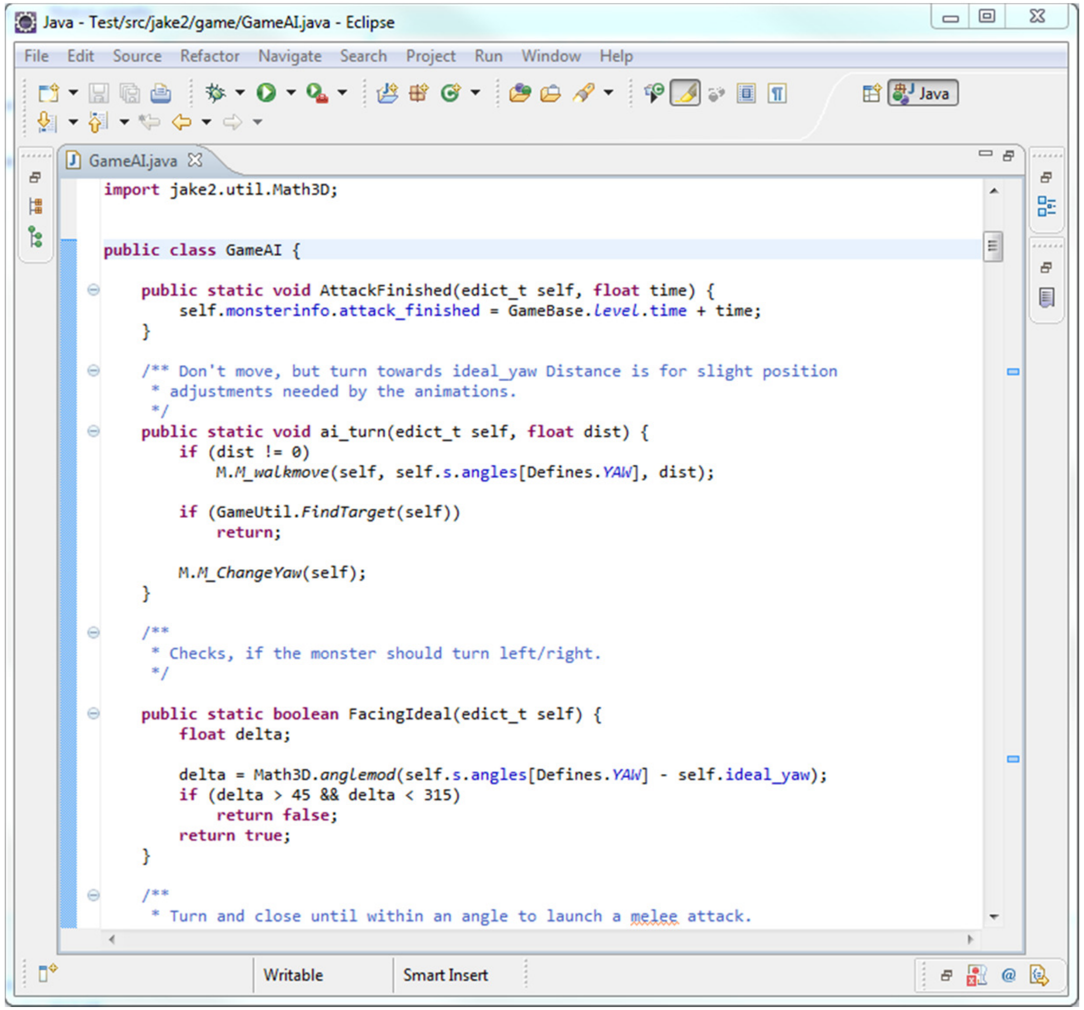
All your history are belong to us (id Software) [link]

1. What is *programming*?

Examples

Jake2

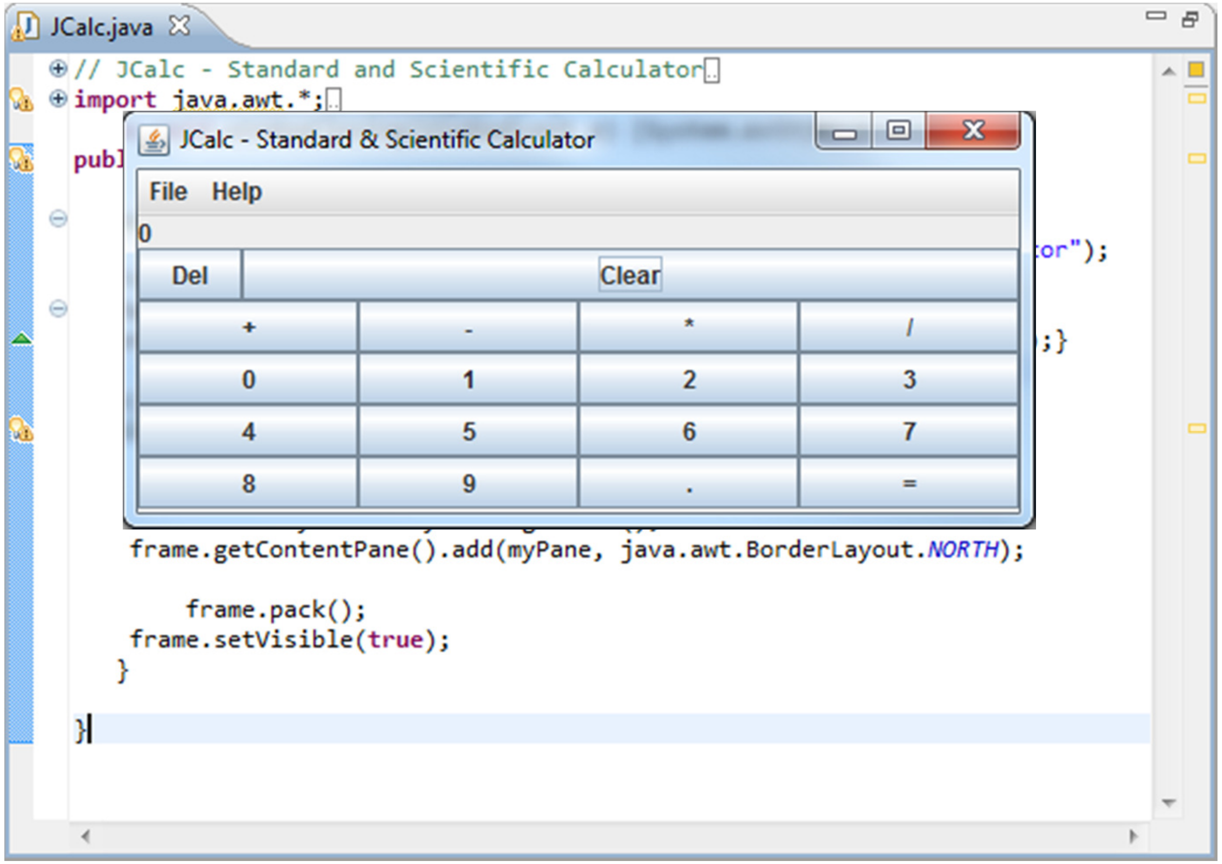
[link]


Java - Test/src/jake2/game/GameAI.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
GameAI.java
Writable
Smart Insert

```
import jake2.util.Math3D;

public class GameAI {

    public static void AttackFinished(edict_t self, float time) {
        self.monsterinfo.attack_finished = GameBase.level.time + time;
    }

    /** Don't move, but turn towards ideal_yaw Distance is for slight position
     * adjustments needed by the animations.
     */
    public static void ai_turn(edict_t self, float dist) {
        if (dist != 0)
            M.M_walkmove(self, self.s.angles[Defines.YAW], dist);

        if (GameUtil.FindTarget(self))
            return;

        M.M_ChangeYaw(self);
    }

    /**
     * Checks, if the monster should turn left/right.
     */

    public static boolean FacingIdeal(edict_t self) {
        float delta;

        delta = Math3D.anglemod(self.s.angles[Defines.YAW] - self.ideal_yaw);
        if (delta > 45 && delta < 315)
            return false;
        return true;
    }

    /**
     * Turn and close until within an angle to launch a melee attack.
```

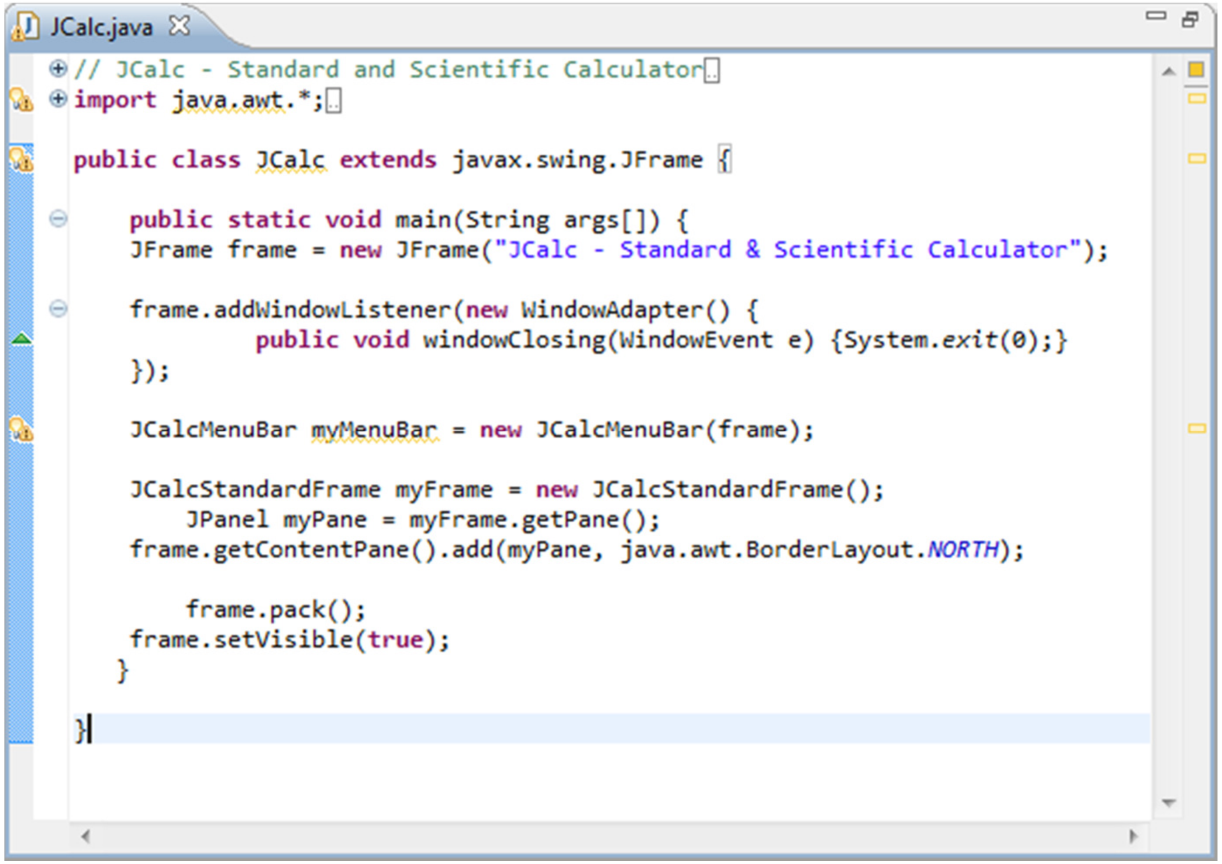
All your history are belong to us (id Software) [link]

jCalculator [link]



```
// JCalc - Standard and Scientific Calculator
import java.awt.*;

publ

                                                                 or");

                                                                 ;}

        frame.getContentPane().add(myPane, java.awt.BorderLayout.NORTH);

            frame.pack();
        frame.setVisible(true);
      }

}
```

jCalculator [link]



```
// JCalc - Standard and Scientific Calculator
import java.awt.*;

public class JCalc extends javax.swing.JFrame {

    public static void main(String args[]) {
    JFrame frame = new JFrame("JCalc - Standard & Scientific Calculator");

    frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {System.exit(0);}
    });

    JCalcMenuBar myMenuBar = new JCalcMenuBar(frame);

    JCalcStandardFrame myFrame = new JCalcStandardFrame();
        JPanel myPane = myFrame.getPane();
    frame.getContentPane().add(myPane, java.awt.BorderLayout.NORTH);

        frame.pack();
    frame.setVisible(true);
   }

}
```

The kind of programs that we will develop…

> Calculate the factorial of a number introduced by the user

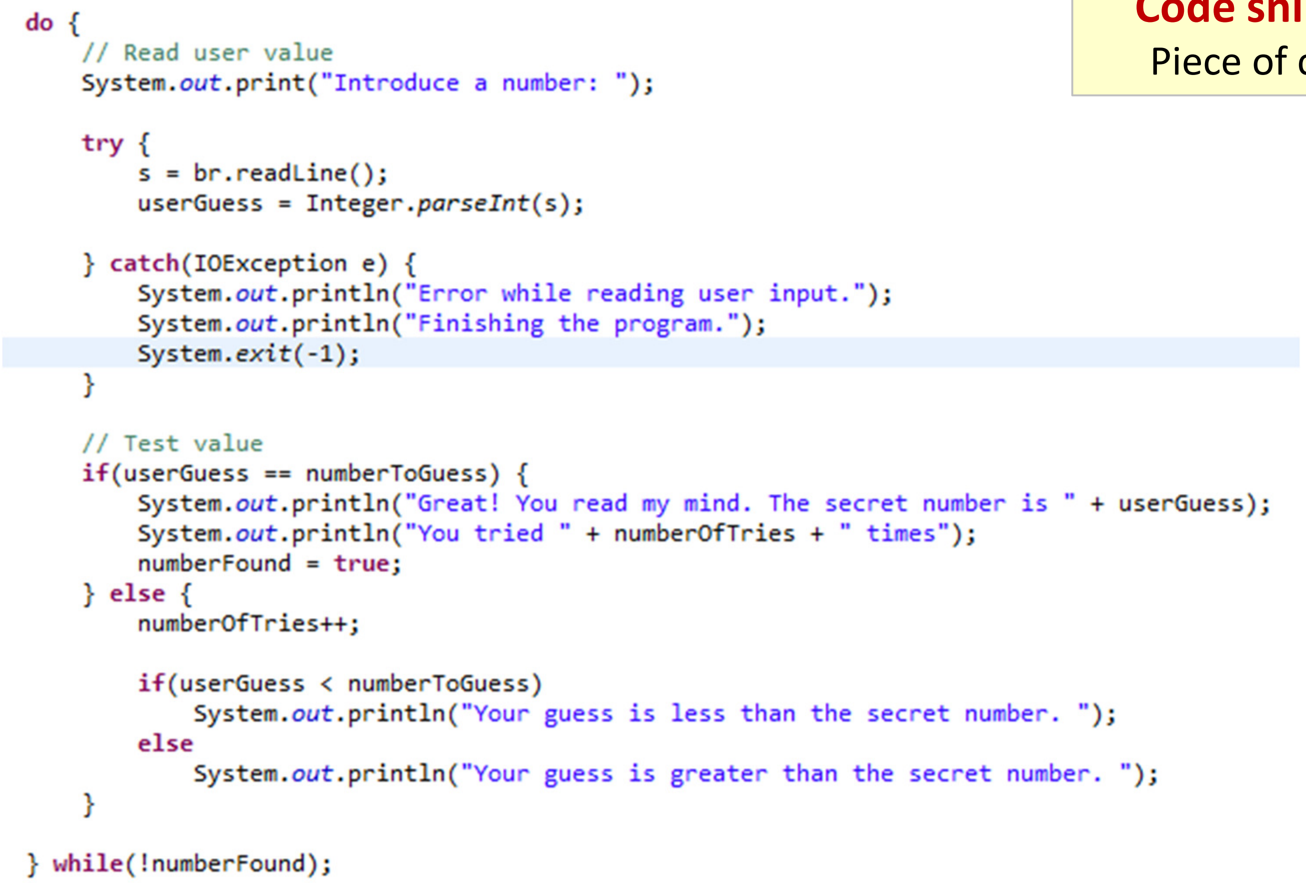
> The computer randomly choses a number. The user make guesses to find out this number. The computer gives back clues to the user

> Simplified version of the Mastermind game

1. What is *programming*?

Examples

Code snippet
Piece of code



```
do {
    // Read user value
    System.out.print("Introduce a number: ");

    try {
        s = br.readLine();
        userGuess = Integer.parseInt(s);

    } catch(IOException e) {
        System.out.println("Error while reading user input.");
        System.out.println("Finishing the program.");
        System.exit(-1);
    }

    // Test value
    if(userGuess == numberToGuess) {
        System.out.println("Great! You read my mind. The secret number is " + userGuess);
        System.out.println("You tried " + numberOfTries + " times");
        numberFound = true;
    } else {
        numberOfTries++;

        if(userGuess < numberToGuess)
            System.out.println("Your guess is less than the secret number. ");
        else
            System.out.println("Your guess is greater than the secret number. ");
    }

} while(!numberFound);
```

Outline

1. What is programming?
2. **Components of a program: data and algorithms**
3. Creating and running programs
4. Programming paradigms
5. Introduction to the Java programming language

Programming

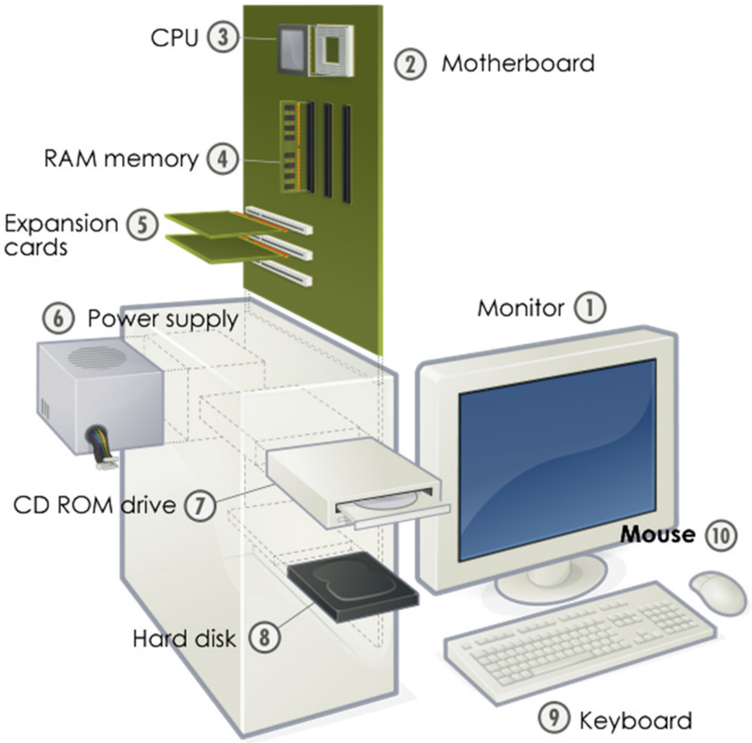
Provide a **computer** with a set of **instructions** and a set of **data** on what should be done with the data for the resolution of a given problem

Hardware

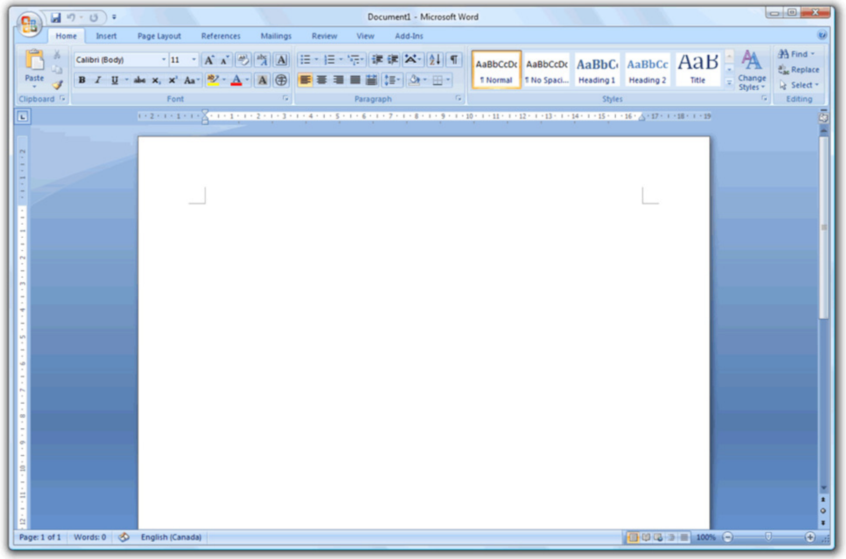
Physical components of a computer (the machine)

Software

Logical instructions, data, and documentation (the programs)



Source: Wikipedia Commons



Source: Wikipedia Commons

2. Data and algorithms

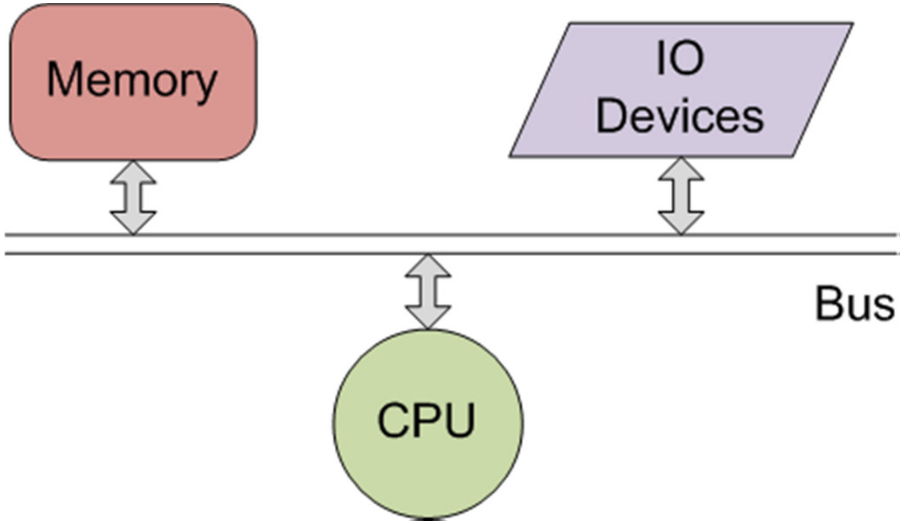
Hardware

99% of computers (including all Personal Computers) have an architecture composed by:

- CPU
- Memory
- I/O Devices

Data and instructions are stored in the memory

(This architecture is called von Neumann architecture, although it was originally proposed by Eckert and Mauchly)



Central processing unit (CPU)

It executes the instructions and coordinates the rest of the elements

Memory

Stores the data, instructions and results

Volatile memory

Devices for input/output

For providing data and instructions and receiving results

Hard disk is usually considered an output device

Data Bus

For sharing information among the previous components

2. Data and algorithms

Hardware

Source: www.hp.com (HP Pavilion p6103uk)

Technical Features

Operating system installed	Genuine Windows Vista® Home Premium 32-bit
Processor type	Intel® Pentium® processor E5200
Chipset	Intel® G31 express chipset
Standard memory	3 GB
Memory	DDR2-SDRAM
Memory slots	2 DIMM sockets
Internal drives	1 TB
Optical drive type	DVD writer SATA DVD RAM and Double Layer supporting LightScribe technology
Network interface	Ethernet 10/100BT integrated network interface
External I/O ports	6 USB 2.0 ports (2 in front)
Video RAM	512 MB dedicated memory, up to 1791 MB total available graphics memory as allocated by Windows Vista®
Video adapter, bus	1 PCI-Express 16x

System Software (Operating System):

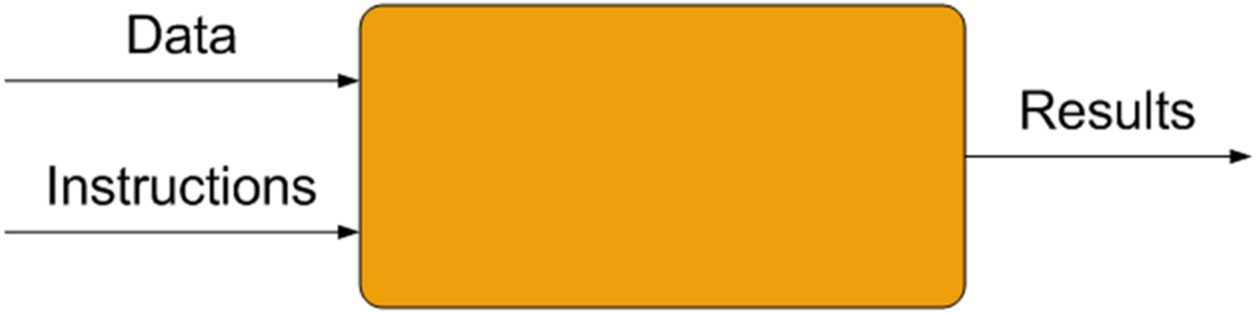
- Provides control over the hardware and underlies applications

Application software

- Programs for specific purposes, solving a specific problem or family of problems
  - Office (word processors, spreadsheets…)
  - Accounting
  - Control
  - Games
  - …

2. Data and algorithms

Abstract representation of a computer



The algorithmical machine

Objective:

Solve a problem (by using a computer)

How?

Use an algorithm (and *implement* it)

An algorithm is:

A set of instructions that allow for the resolution of a problem step by step

A **well-defined, ordered, and finite list of operations** that is able to find a solution for a problem

Instructions to create a paper plane

Fold a sheet of paper exactly in half long-ways, and re-open it so you have a crease separating the two halves

On one end of the paper, fold each corner in towards the center to the point where the inside edges are even with the centerline crease

Starting at the very tip of the point, fold the paper down on each side so the inside edges line up with the center crease

Turn the paper airplane over and fold it in half along the centerline

Fold the first wing with the line of the fold running nearly parallel to the centerline of the plane. Make this fold from 1/2 to 1 inch from the center. Step 6 shows this fold more clearly

Fold the second wing exactly as you did the first

Source: 10paperairplanes.com [link]

Ordered and finite
…but well-defined?

2. Data and algorithms

Algorithms

The previous example is written in **natural language**: is a form easily readable by people

Computers

Do not understand natural language

Offer a restricted collection of instructions

Do not admit imprecision: *one end of the paper*, *nearly parallel,* etc.

How do we instruct a computer what to do: Translate the algorithm into a program written in a **programming language** suitable for the implementation of that algorithm

There exist many languages for programming computers (e.g. C++, **Java**, etc.)

Outline

1. What is programming?
2. Components of a program: data and algorithms
3. **Creating and running programs**
4. Programming paradigms
5. Introduction to the Java programming language

Creating and running a program with Eclipse IDE

1. Run Eclipse IDE
2. Select workspace folder
3. Create project (File > New > Java Project, set name *Test*)
4. Create program (File > New > Class, set name *HelloWorld*)
5. Type the code (see next slide)
6. Run the program (Run > Run)

Create/develop/ write/implement
Write the program in a programming language

Run/execute
Put the program into functioning

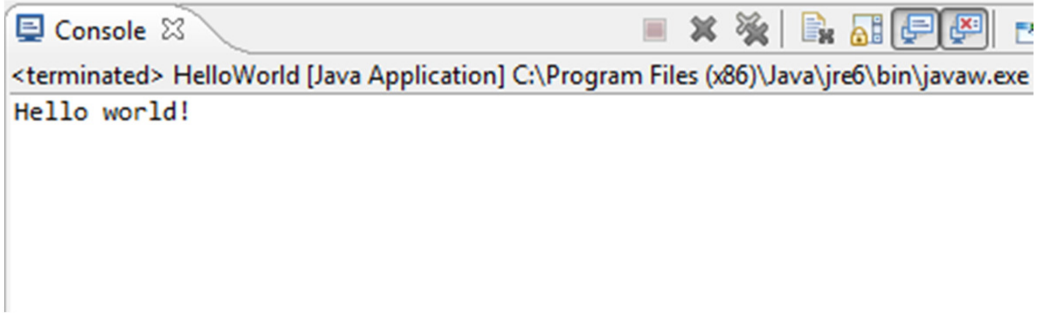
At home:

a. Download JDK [link]
b. Download Eclipse IDE for Java Developers [link]
c. Unzip folder
d. Double click *eclipse* file to run Eclipse IDE

3. Creating and running programs

Our first program!

```
/* My first Java program! */

public class HelloWorld {

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

}
```



```
<terminated> HelloWorld [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe
Hello world!
```

Binary language (machine code)

0s and 1s

Low level languages

Very basic operations (move registers, add, etc.)

High level languages

Closer to natural language

…but no so much

3. Creating and running programs

Binary language

Binary language (or machine code) is the language that the computer can directly understand

- Data and instructions are encoded using sets of 0 and 1
- The fastest: talking to the computer on its own idiom
- Very error prone, very complicated

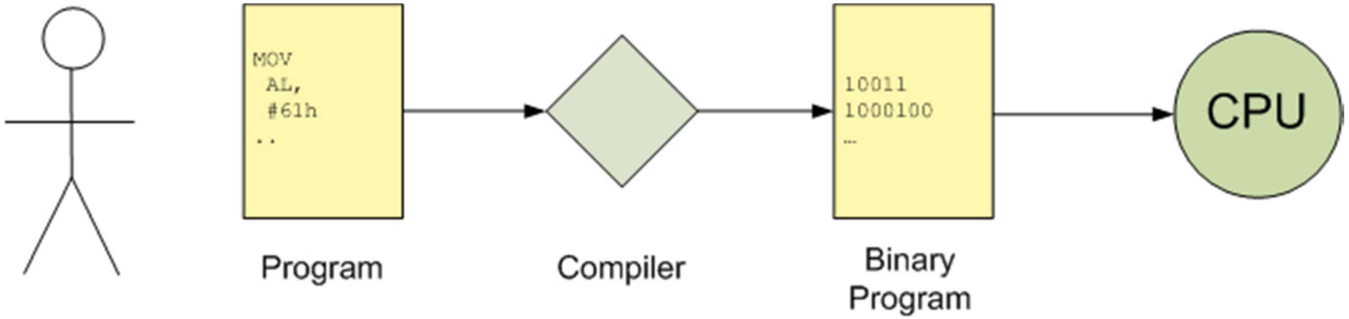
E.g.: Adding the registers 1 and 2 and placing the result in register 6 (MIPS architecture)

type	Op 1	Op 2	Res	Shift	Function
0	*1*	*2*	*6*	*0*	*32*
000000	**00001**	**00010**	**00110**	**00000**	**100000**

Low level instructions expressed as text

- Not very intuitive
- Processor-dependent: a specific set of instruction for each processor type

Compiler: Program that translates assembly code into a binary program



```
.model small
.stack
.data
String1 DB 'HelloWorld.$'
.code
program:
     mov ax, @data
     mov ds, ax
     mov dx, offset String1
     mov ah, 9
     int 21h
     mov ah,4ch
     int 21h
end program
```

3. Creating and running programs

High-level languages

High-level languages are intended to bring programming languages closer to human language

A program encoded in a high-level programming language is translated into binary code

- *Compilation*
- *Interpretation*

There exist over 300 (over 2400 with dialects) [link] [link]

The pioneers included concepts such as:

- Variables –it is not necessary to directly manage data in memory
- Complex data structures
- New instructions, other than those provided by the computer

3. Creating and running programs

High-level languages

Structured languages

Group data and instructions in blocks of code (no GOTO)

Modular languages

The program is divided into separate modules (C, Pascal)

Object-oriented languages

Data and operations are conceptually grouped into objects (C++, Java)

Component-oriented languages

Programs are constructed by gluing together sets of pieces (.NET platform)

Web-oriented languages

Specially suited to develop web applications (JavaScript, Ruby)

…

3. Creating and running programs

Compilation and interpretation

The translation from a program written in a programming language into binary code can be done in two ways:

All at once: **compilation**

- An executable program is generated (plus intermediate object files)
- Faster

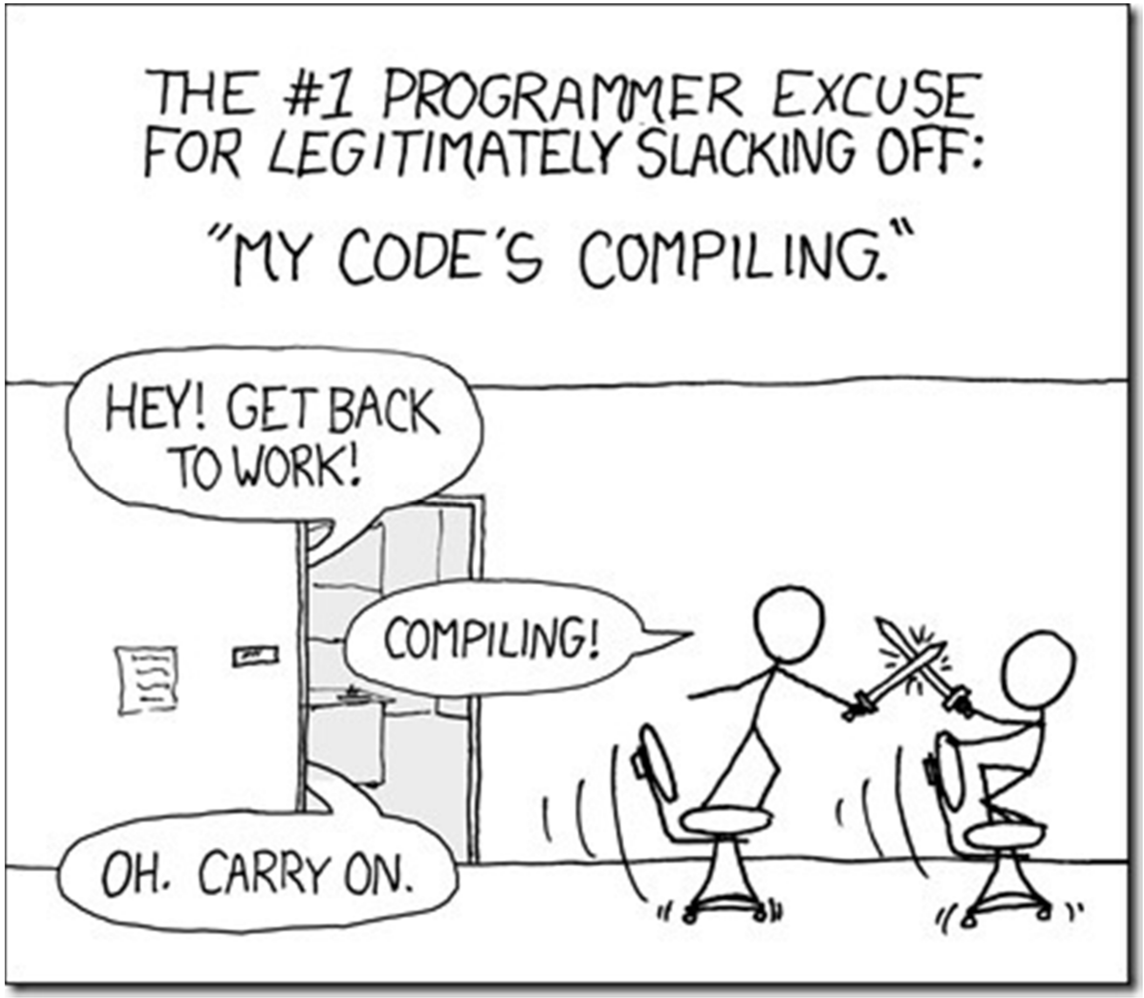
One instruction at a time: **interpretation**

- Run even if there are errors in the program (as long as the current instruction is correct)
- More flexible

Java has a hybrid schema

- Pre-compilation to bytecode
- Interpretation by means of a Java Virtual Machine

Compilation time!

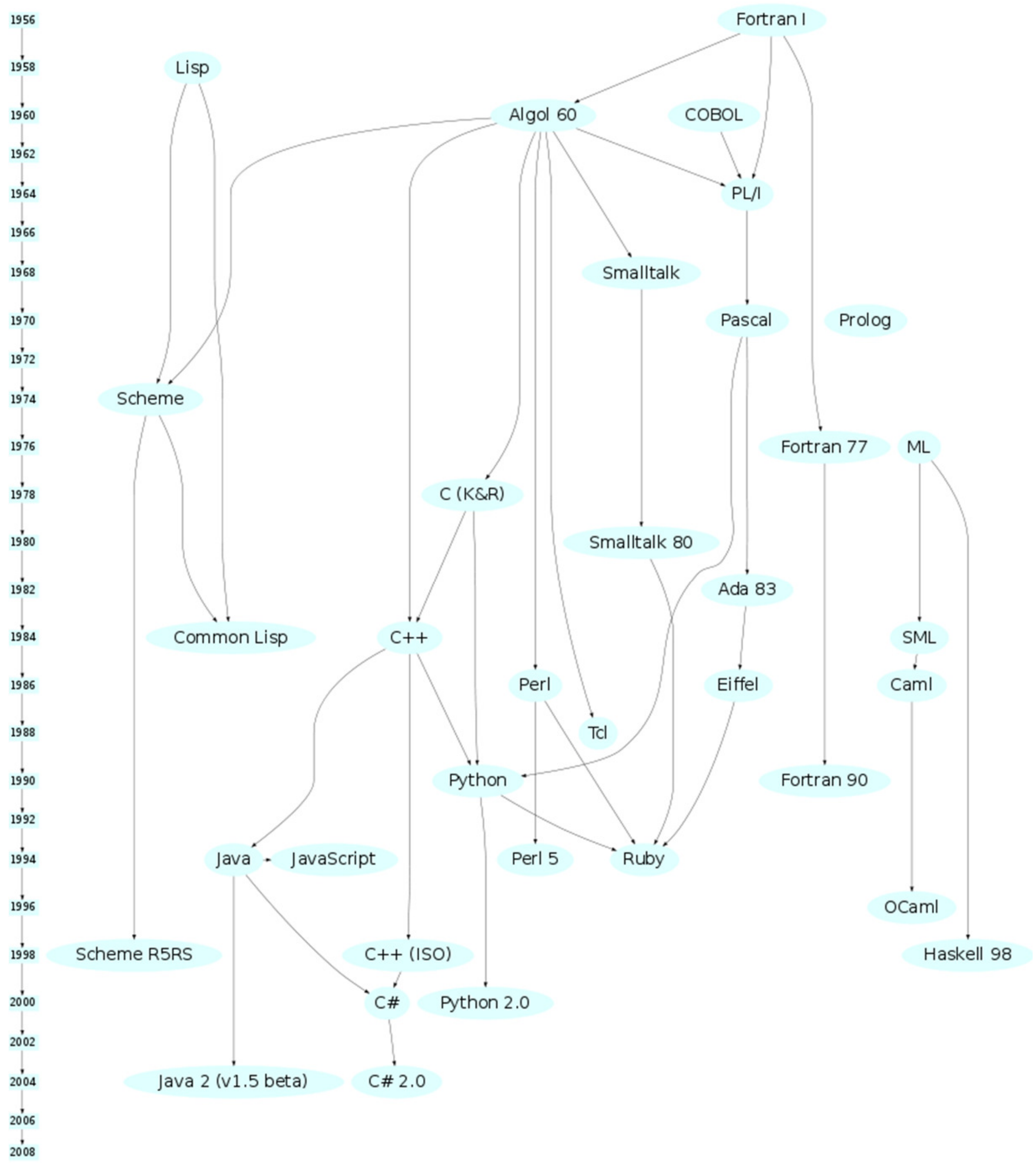


Source: XKCD (http://xkcd.com/303/)

Compilation time
Code development

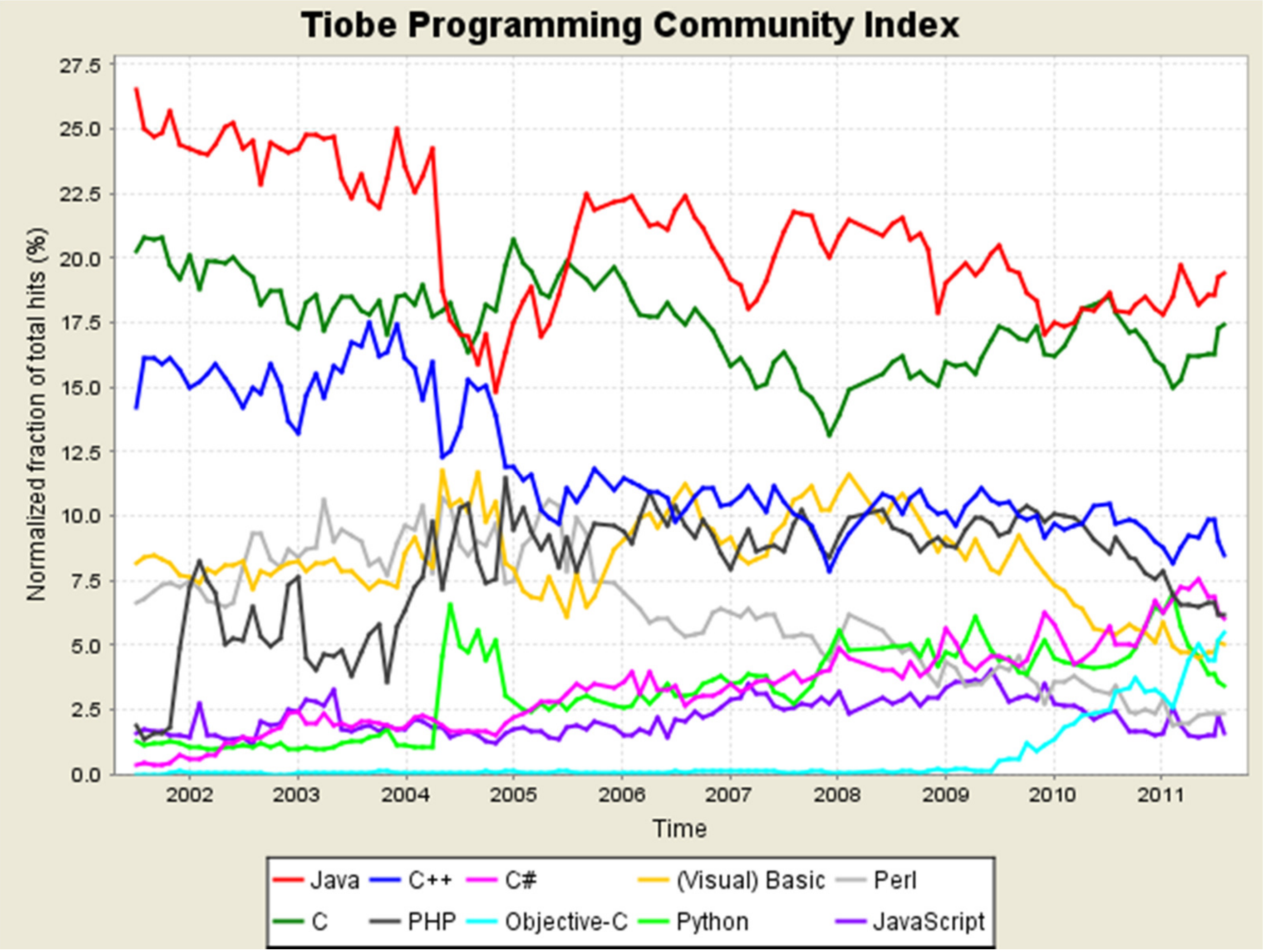
Runtime
Program execution

Evolution of programming languages



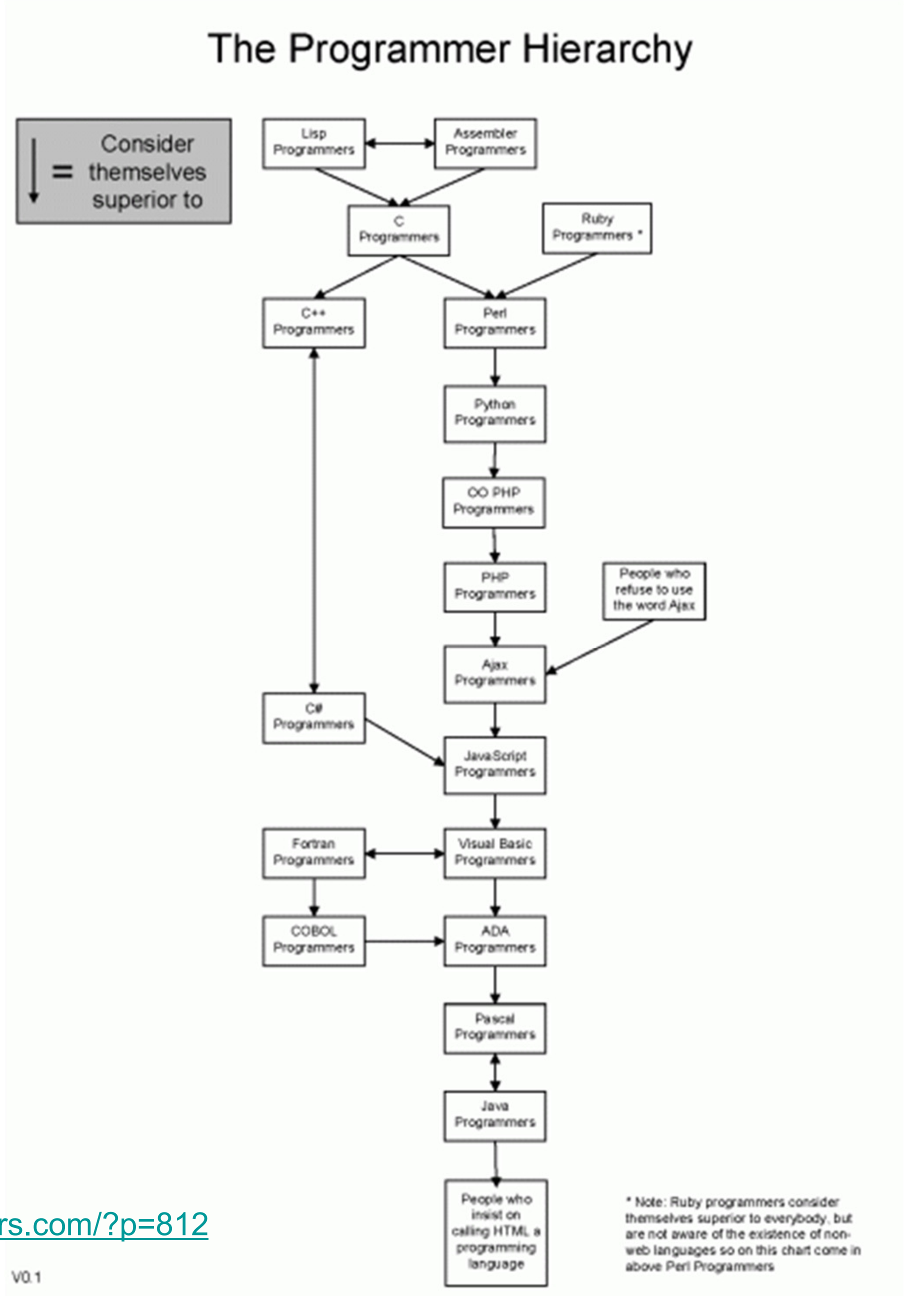
Source: http://people.mandriva.com/~prigaux/language-study/diagram-light.png

Usage of some languages



Source: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Which programming language is the best?



* Note: Ruby programmers consider themselves superior to everybody, but are not aware of the existence of non-web languages so on this chart come in above Perl Programmers

Source: http://www.thesmokesellers.com/?p=812

Outline

1. What is programming?
2. Components of a program: data and algorithms
3. Creating and running programs
4. **Programming paradigms**
5. Introduction to the Java programming language

4. Programming paradigms

A **programming paradigm** is a *philosophy* to solve a problem with a computer program

Imperative programming (Java, C++, Python, Perl)

A program describes the necessary steps that need to be taken to solve the problem

Functional programming (Lisp, Erlang, Haskell, F#)

Program instructions are given as mathematical expressions

Logic programming (Prolog)

Program does not include instructions, but logical formulas

The problem is solved through logical inference.

None is better than the other

Many languages are mixed

4. Programming paradigms

Java – *Factorial.java*

```java
public class Factorial {
  public static double factorial(int n) {
    int f = 1;
    for(int i=2; i<=n; i++)
      f *= i;
    return f;
  }

  public static void main(String [] args) {
    factorial(42);
  }
}
```

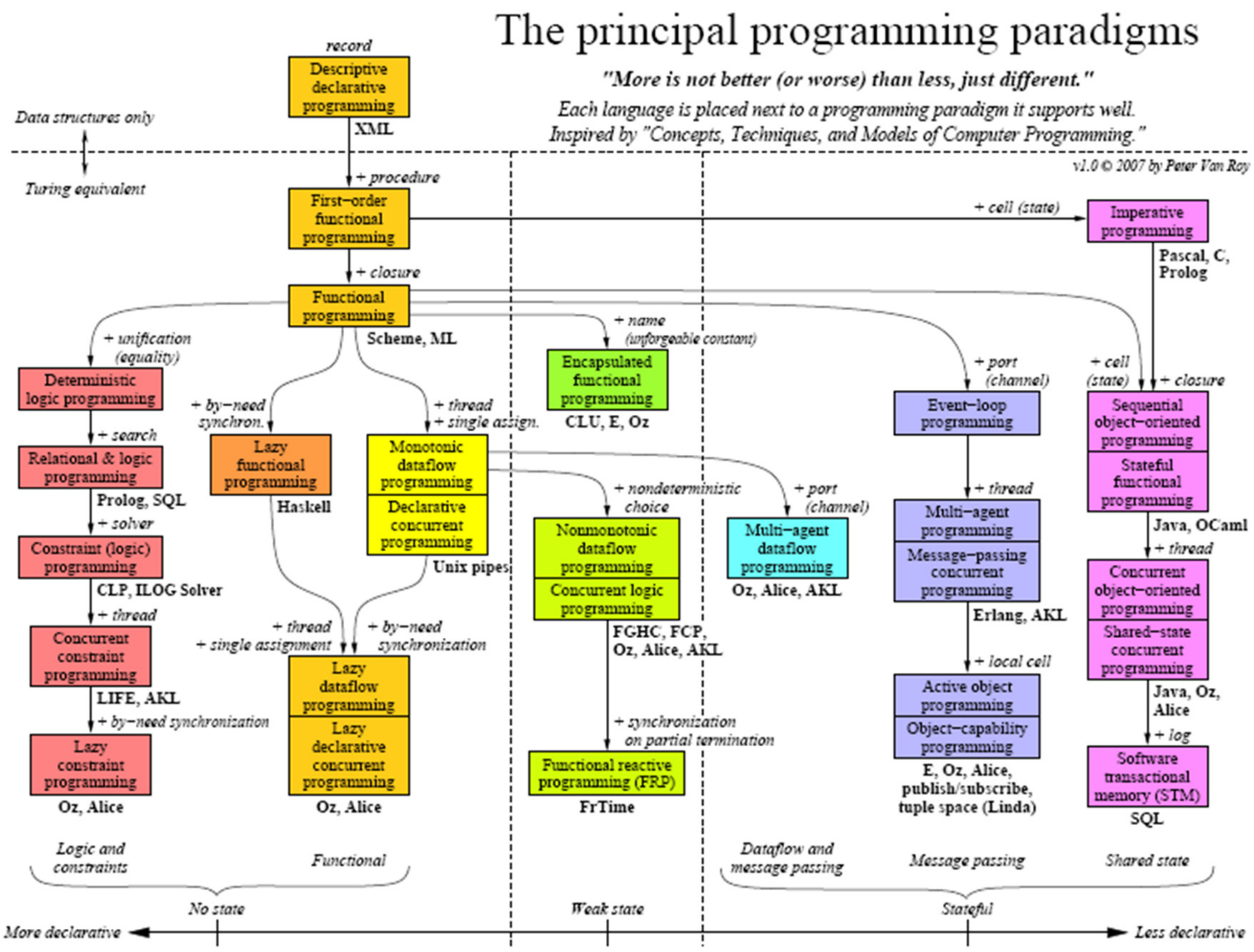
Haskell – *fac.hs*

```haskell
fac 0 = 1 fac n = n * fac (n-1)
main = print (fac 42)
```

Prolog - *factorial.hs*

```prolog
factorial(0,1).

factorial(N,F) :-
    N>0,
    N1 is N-1,
    factorial(N1,F1),
    F is N * F1.

?- factorial(42,X).
```

4. Programming paradigms


The principal programming paradigms
"More is not better (or worse) than less, just different."
Each language is placed next to a programming paradigm it supports well.
Inspired by "Concepts, Techniques, and Models of Computer Programming."
v1.0 © 2007 by Peter Van Roy
Data structures only
Turing equivalent
record
Descriptive declarative programming
XML
+ procedure
First–order functional programming
+ cell (state)
Imperative programming
Pascal, C, Prolog
+ closure
Functional programming
Scheme, ML
+ unification (equality)
Deterministic logic programming
+ search
Relational & logic programming
Prolog, SQL
+ solver
Constraint (logic) programming
CLP, ILOG Solver
+ thread
Concurrent constraint programming
LIFE, AKL
+ by–need synchronization
Lazy constraint programming
Oz, Alice
+ by–need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Unix pipes
+ thread
+ single assignment
+ by–need synchronization
Lazy dataflow programming
Lazy declarative concurrent programming
Oz, Alice
+ name (unforgeable constant)
Encapsulated functional programming
CLU, E, Oz
+ nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
FGHC, FCP, Oz, Alice, AKL
+ synchronization on partial termination
Functional reactive programming (FRP)
FrTime
+ port (channel)
Multi–agent dataflow programming
Oz, Alice, AKL
+ port (channel)
Event–loop programming
+ thread
Multi–agent programming
Message–passing concurrent programming
Erlang, AKL
+ local cell
Active object programming
Object–capability programming
E, Oz, Alice, publish/subscribe, tuple space (Linda)
+ cell (state)
+ closure
Sequential object–oriented programming
Stateful functional programming
Java, OCaml
+ thread
Concurrent object–oriented programming
Shared–state concurrent programming
Java, Oz, Alice
+ log
Software transactional memory (STM)
SQL
Logic and constraints
Functional
Dataflow and message passing
Message passing
Shared state
No state
Weak state
Stateful
More declarative
Less declarative

Source: http://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng.pdf

Outline

1. What is programming?
2. Components of a program: data and algorithms
3. Creating and running programs
4. Programming paradigms
5. **Introduction to the Java programming language**

***…but before*:**

Additional lectures (1-4)

Introduction to programming

Recommended lecture:

H. M. Deitel, P. J. Deitel. *Java: How to Program. Prentice Hall, 2007 (7th Edition)*, Chapter 1 [link]

Additional lectures on "Introduction to Programming"

http://en.wikipedia.org/wiki/Programming_language
http://nayar.uan.mx/~iavalos/introprog.htm
http://mosaic.uoc.edu/recursos/Introduccion_a_la_Programacion.pdf
http://elvex.ugr.es/decsai/java/pdf/2B-Java.pdf
http://www.landofcode.com/programming-intro/
http://www.bfoit.org/itp/
http://chortle.ccsu.edu/java5/index.html

http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf (Spanish)

Outline

1. What is programming?
2. Components of a program: data and algorithms
3. Creating and running programs
4. Programming paradigms
5. **Introduction to the Java programming language**

A high-level object oriented language

Sun Microsystems (1991) designs a language for embedded systems (set-top-boxes, electrical appliances)

Requirements for the new language:

- Object oriented
- Multiplatform
- No company shows interest in the language

Language simple, small, neutral

5. Introduction to Java

Features

Object Oriented

Absolutely Portable

Interpreted Language

Bytecode is machine independent

Java Virtual Machine (JVM)

Automatic management of dynamic memory

Garbage collector

Case sensitive

Distributed

Robust

Secure

Efficient (JIT compilation)

Clean?

5. Introduction to Java

History

1995: Java is introduced on the Internet, very complete language

Netscape 2.0 introduces the first JVM (Java Virtual Machine) in a web browser

Java Philosophy: "*Write once, run everywhere*"

1997: Appears Java 1.1. Many improvements with respect to 1.0

1998: Java 1.2 (Java 2). Very mature platform Supported by large companies: IBM, Oracle, Inprise, Hewtlett-Packard, Netscape, Sun

1999: Java Enterprise Edition. Revolutionizes server side programming

2006: Java SE 6 is launched

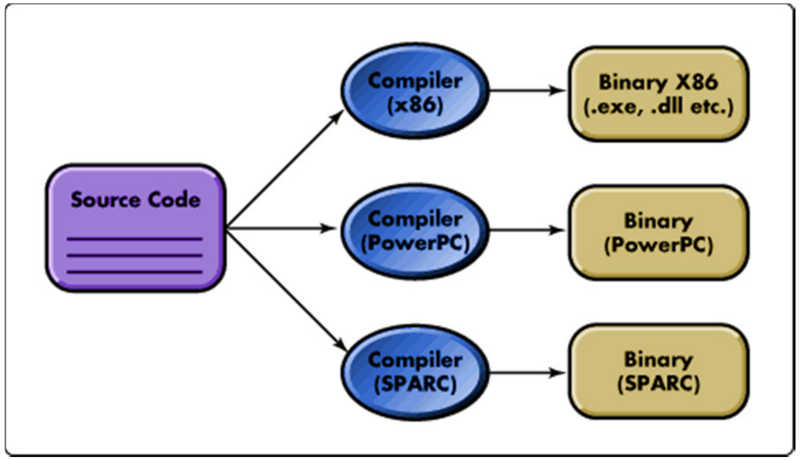
2007: Sun publishes Java core as open-source software (GPL)

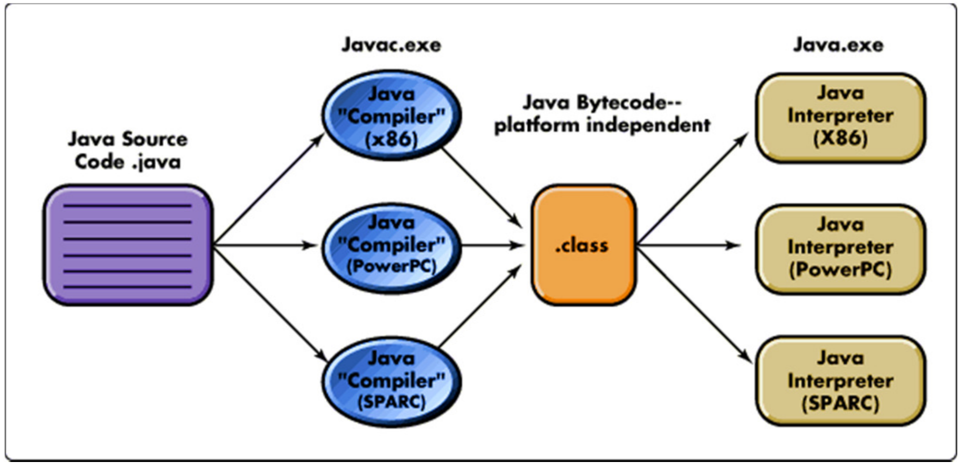
2009: Oracle acquires Sun

2011: Oracle launches Java SE 7

Java uses a *virtual machine*

Two steps are needed, but platform independence is accomplished





Source: http://support.novell.com/techcenter/articles/ana19970701.html

Just-in-time (JIT) compilation [link]

Multiple specifications

- J2ME (Java 2 Mobile Edition)
- **J2SE** (Java 2 Standard Edition)
- J2EE (Java 2 Enterprise Edition)

Multiple technologies

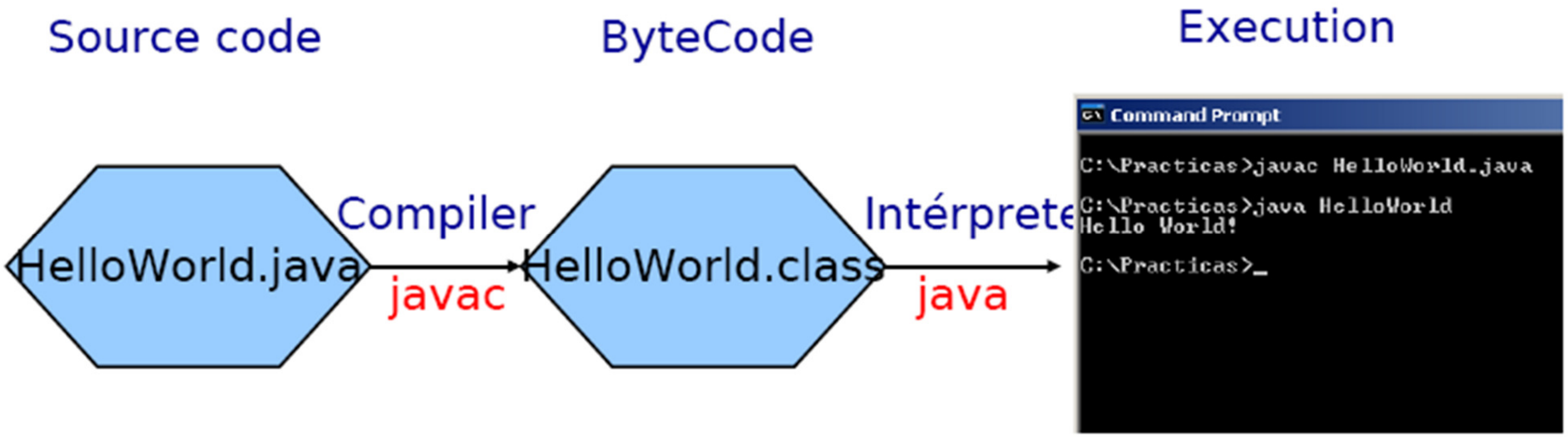
- Programming: java.*, JNI, Java Beans
- UI Programming: AWT, Swing
- Graphics programming: Java 2D, Java 3D
- www: Applets
- Server: JSP, Servlets
- Distributed programming: RMI, Corba, EJB
- Databases: JDBC
- *Third-party tools!*

Java SDK (Java Software Development Kit)

- Includes compiler and other development tools
  - `javac`
- Includes JRE interpreter to run Java bytecodes
  - `java`

Command-line tools!

javac and java



```
C:\Practicas>javac HelloWorld.java

C:\Practicas>java HelloWorld
Hello World!

C:\Practicas>_
```

Software that **supports** program development, debugging and running

- Project management
- Syntax highlight
- Productivity
- Visual modeling
- Debugging
- Rapid development

Examples

- **Eclipse**
- Netbeans
- JBuilder
- Oracle Jdeveloper
- BlueJ

Summary

Introduction to programming

What is programming?

- Solve problems using computer
- An algorithm is used for the resolution of problems
- An algorithm is written in a programming language

Basic computer architecture

- CPU
- Memory
- Devices for I/O
- Abstract algorithmical machine

Programming languages

- Machine code
- Low level languages
- High level languages

First steps

- Compilation vs. interpretation
- Program execution

Programming paradigms

- Imperative programming
- Functional programming
- Logic programming

Summary

Java programming language

Java is an object-oriented programming language

Java has a hybrid compilation process

- Compilation to bytecode
- Interpretation with JVM

Java SDK includes java core libraries and tools (compiler, execution, etc.)

IDEs (e.g. Eclipse) support program development

Programming is easy and fun!

Recommended lectures:

H. M. Deitel, P. J. Deitel. *Java: How to Program. Prentice Hall, 2011 (9th Edition)*, **Chapters 1** [link], **2** [link]

K. Sierra, B. Bates. *Head First Java.* O'Reilly Media, 2005 (2nd Edition), **Chapter 1** [link]

Programming – Grado en Ingeniería Informática	
Authors	
Of this version: Juan Gómez Romero *Based on the work by:* Ángel García Olaya Manuel Pereira González Silvia de Castro García Gustavo Fernández-Baillo Cañas	Universidad Carlos III de Madrid