

AP Computer Science A

Java Programming Essentials [Ver.4.0]

Unit 1: Using Objects and Methods

CHAPTER 21: RECURSION

DR. ERIC CHOU
IEEE SENIOR MEMBER



AP Curriculum Goals

- Introduction to Algorithms, Programming, and Compilers (T 1.1)
- Documentation with Comments (T 1.8)

Objectives

- What is Computer?
- What is Computer Science?
- What Are Programming Languages?
- First Java Program
- Java Language
- Interpretation Levels
- Java Knowledge



What is Computer?

Lecture 1

What is computer? (from Wikipedia)

com·put·er

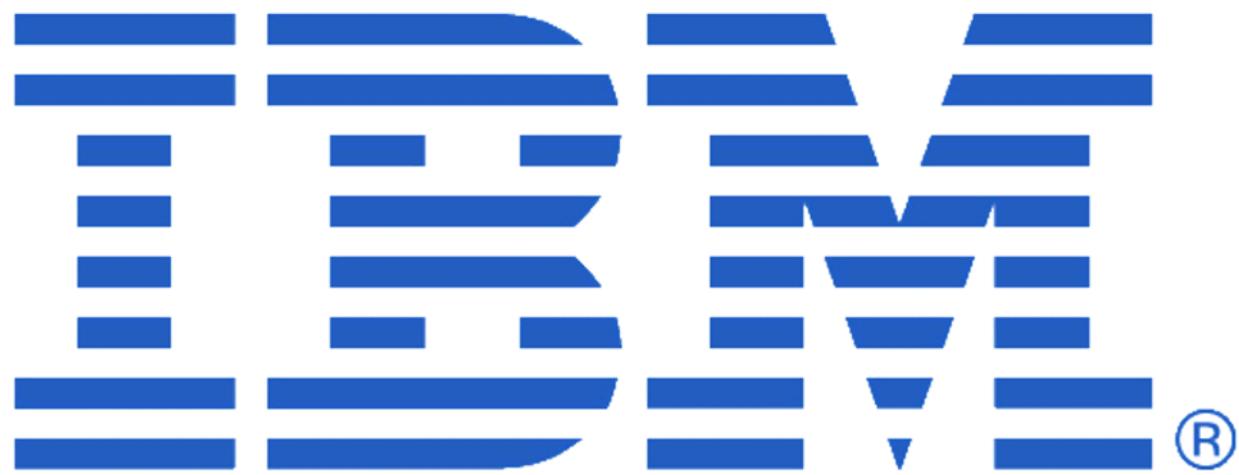
/kəm'pyoodər/

noun

an electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.

synonyms: personal computer, PC, laptop, netbook, ultraportable, desktop, terminal;
[More](#)

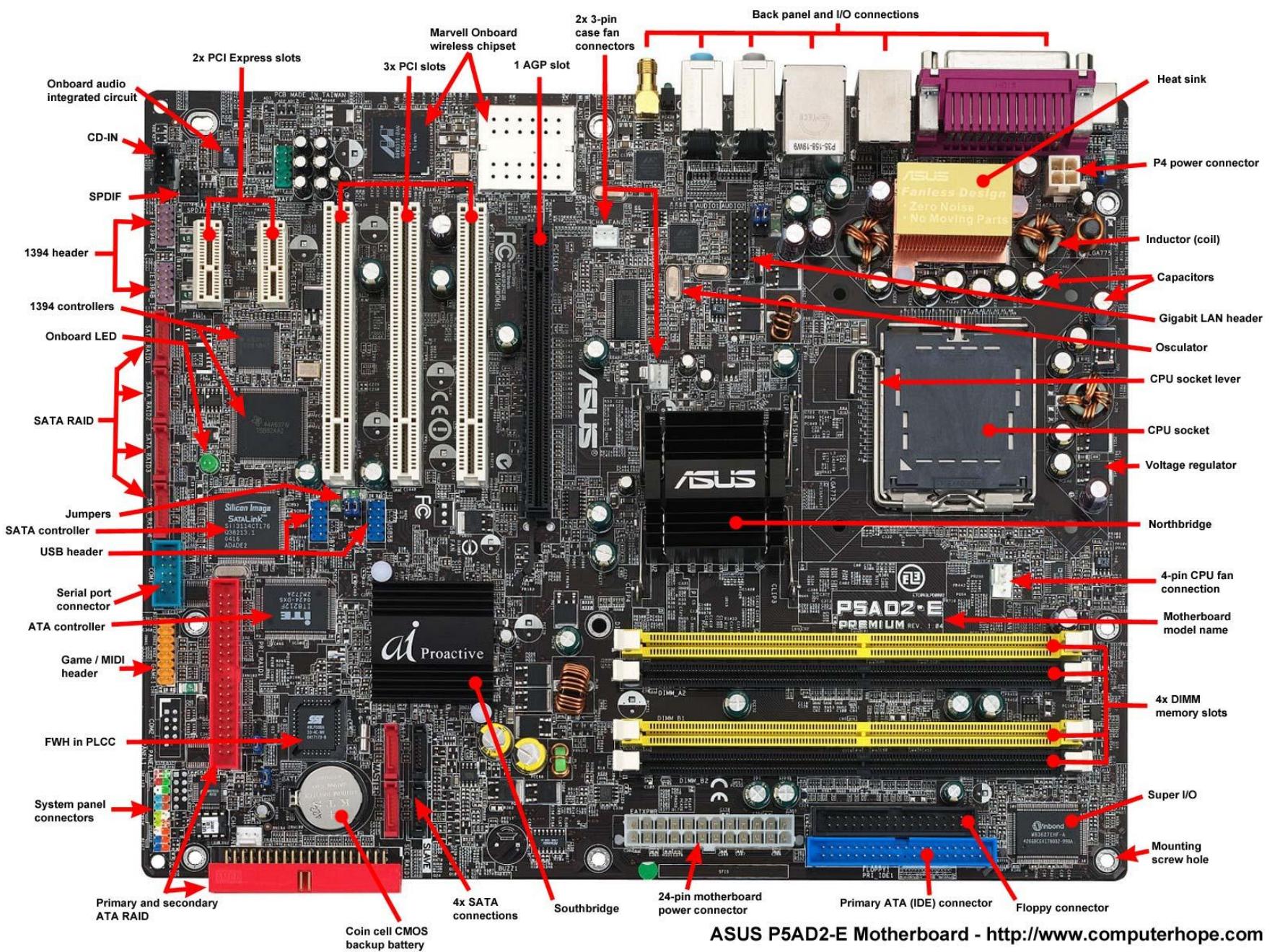
- a person who makes calculations, especially with a calculating machine.







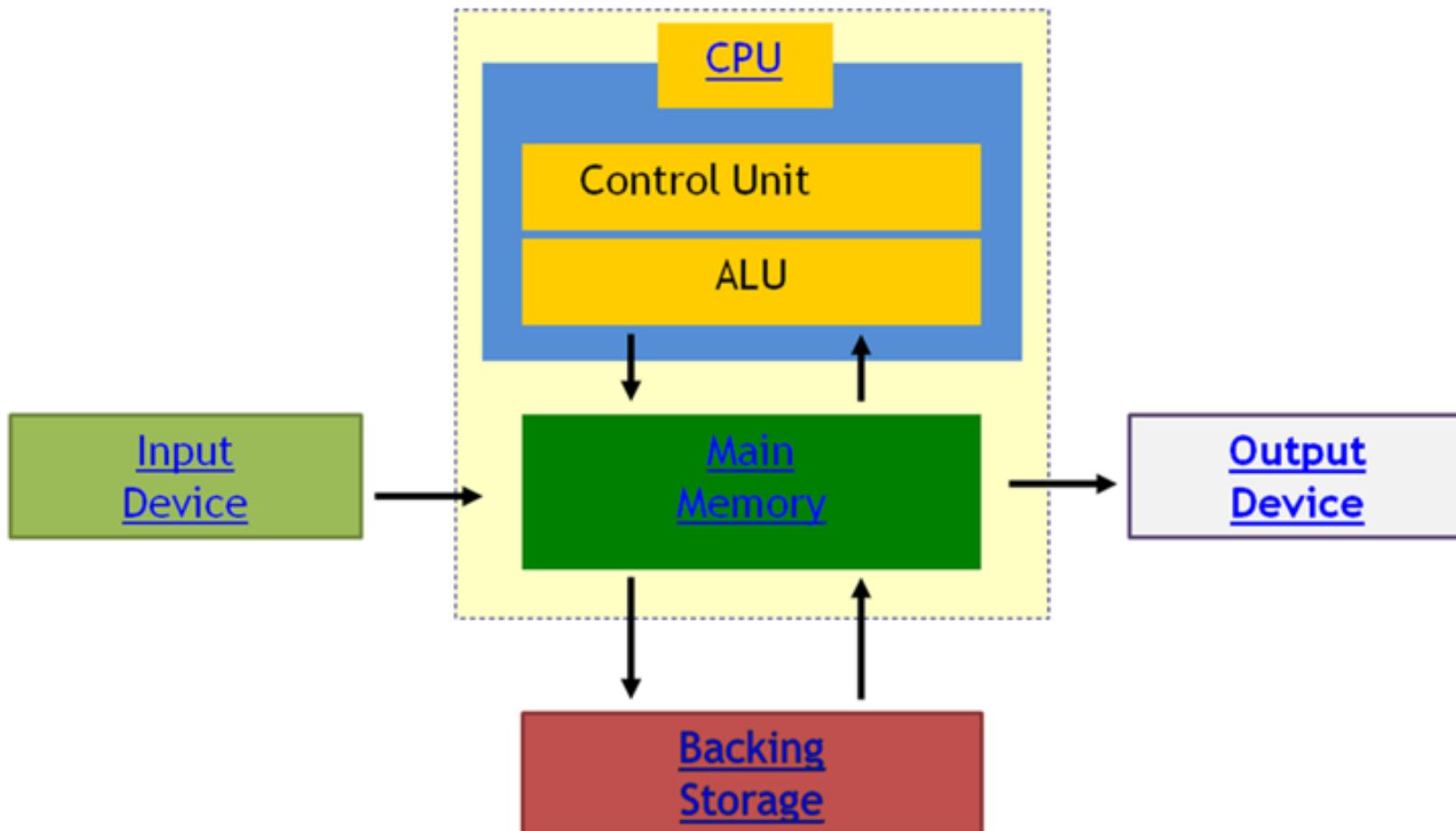




ASUS P5AD2-E Motherboard - <http://www.computerhope.com>

Computer System Diagram

Hardware View



Input Devices of Computer



Touch screen



Camera



Scanner



Mouse



Keyboard



Joystick



Web cam



Microphone



Track ball

SPEAKER



MONITOR



HEADPHONE



Output Devices of Computer

PLOTTER

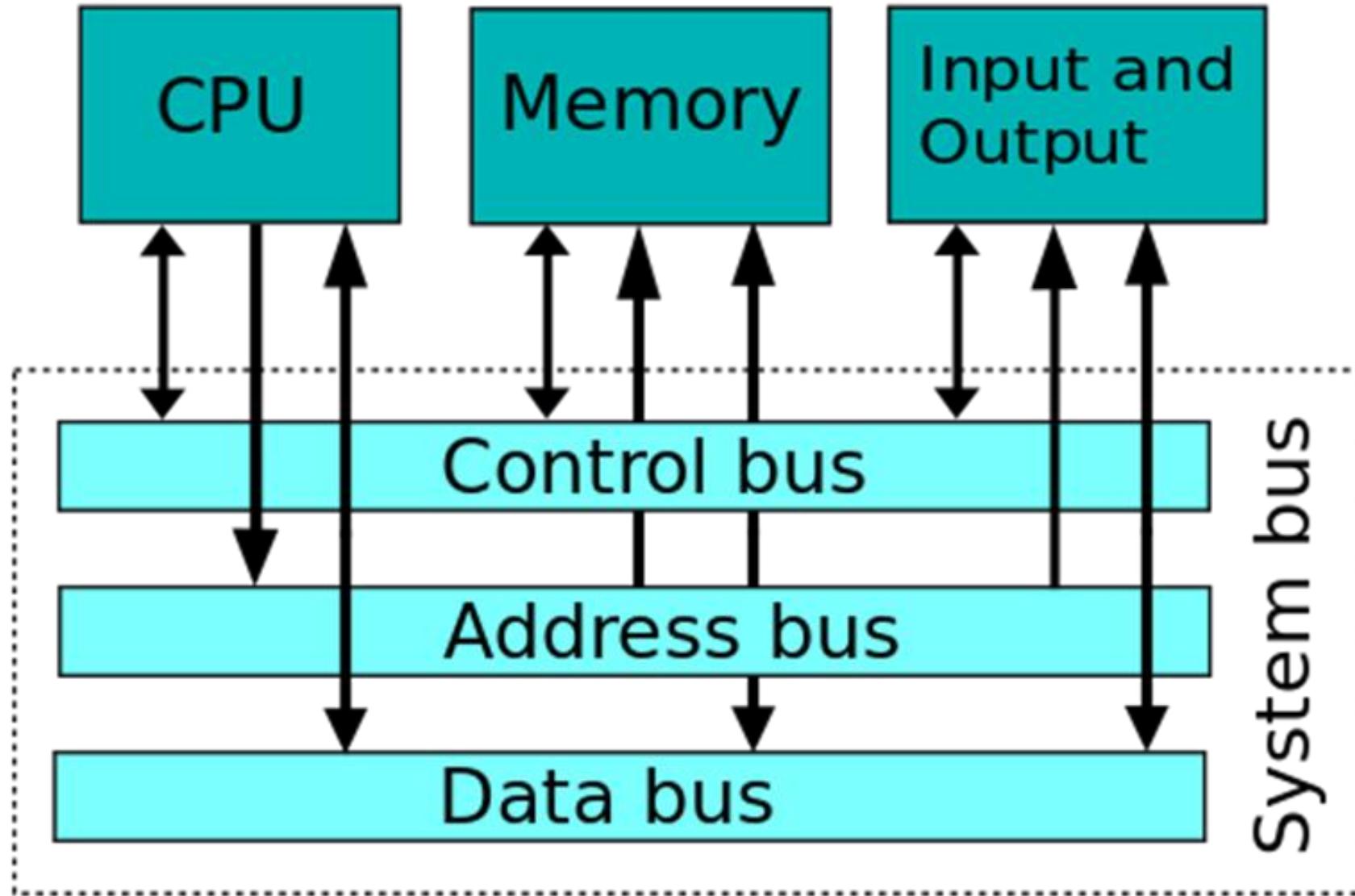


PROJECTOR



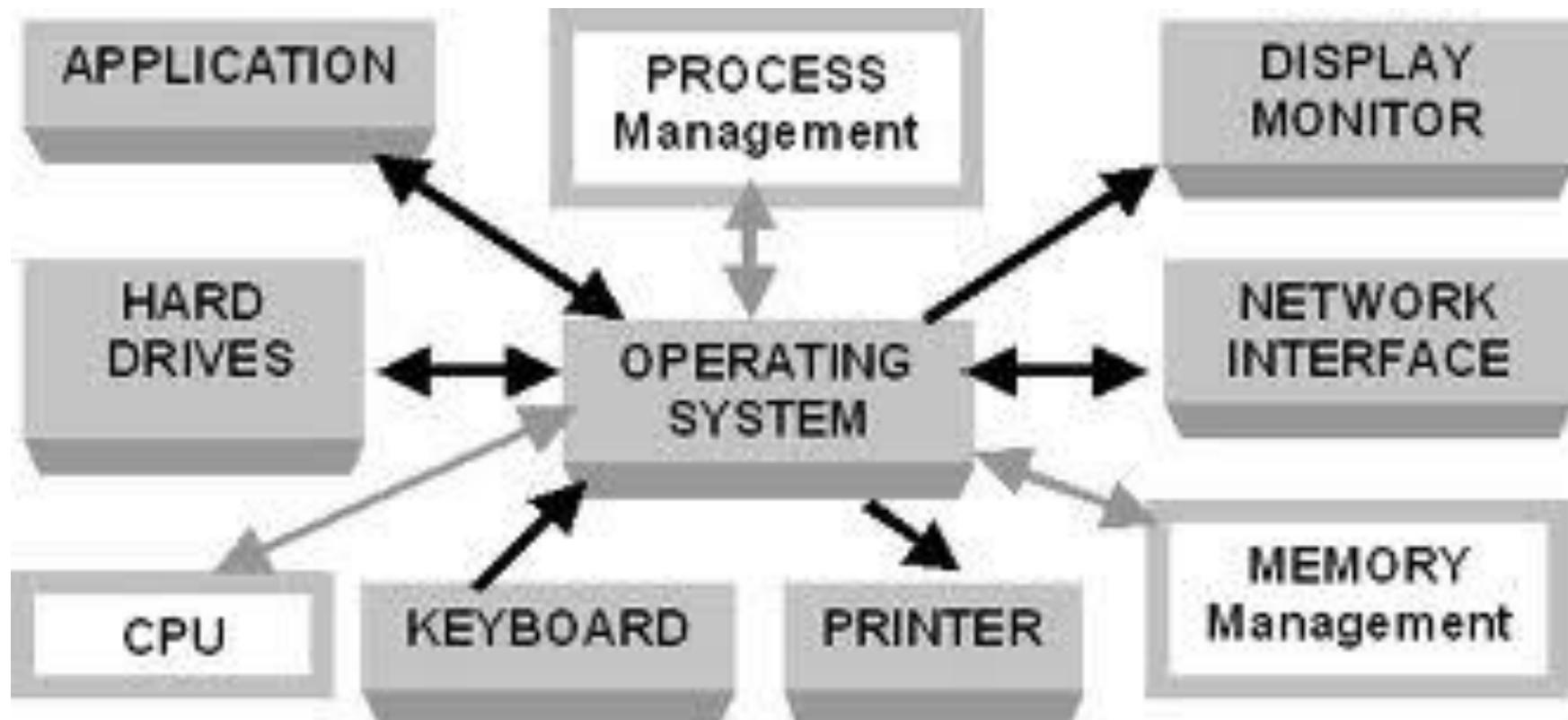
PRINTER





Computer System Diagram

(Operation System/Software View)





What is Computer Science?

Lecture 2

The Definition of Computer Science

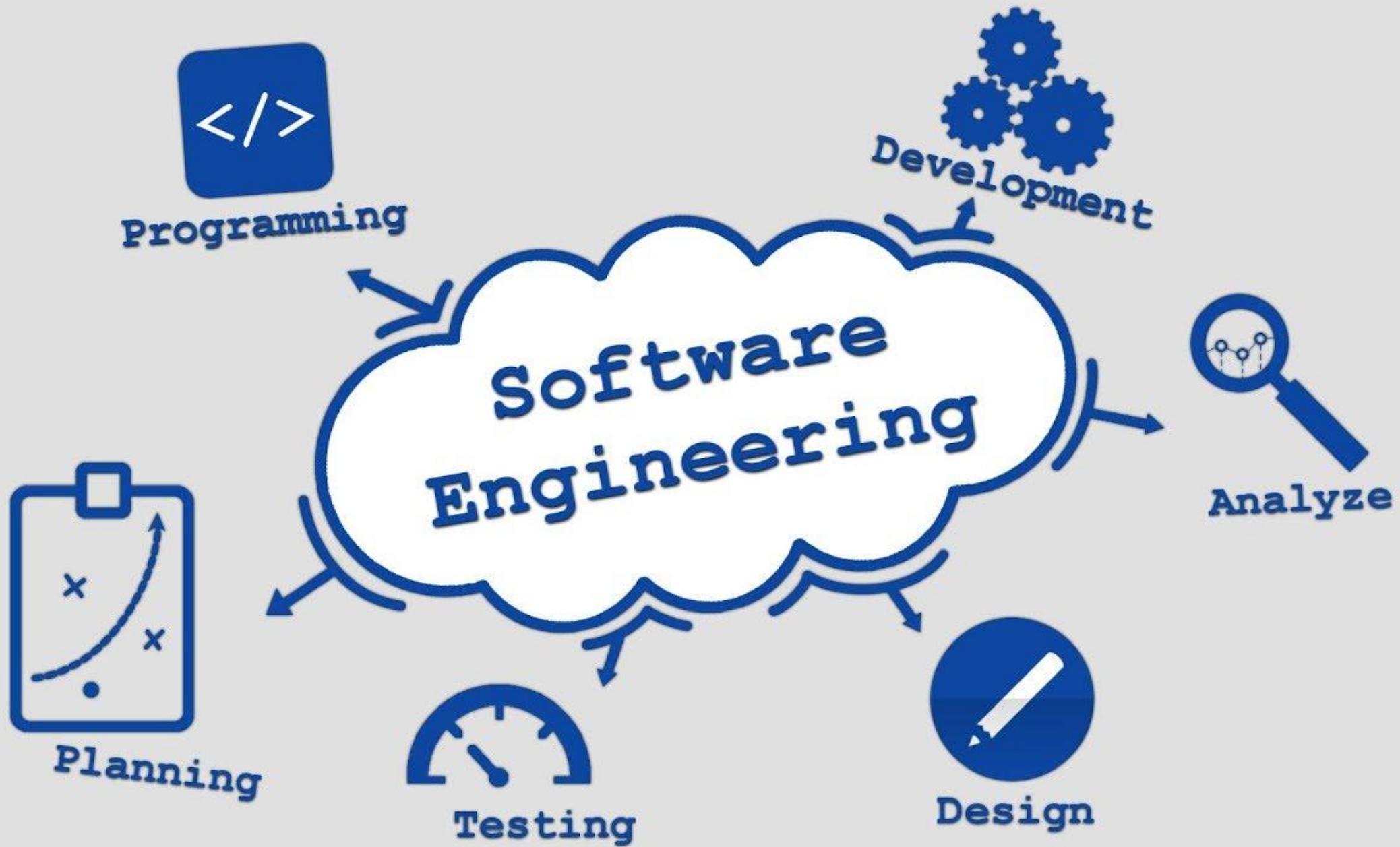
- Gibbs and Tucker definition of computer science
 - The study of algorithms, including their:
 - Formal and mathematical properties
 - Hardware realizations
 - Linguistic realizations
 - Applications

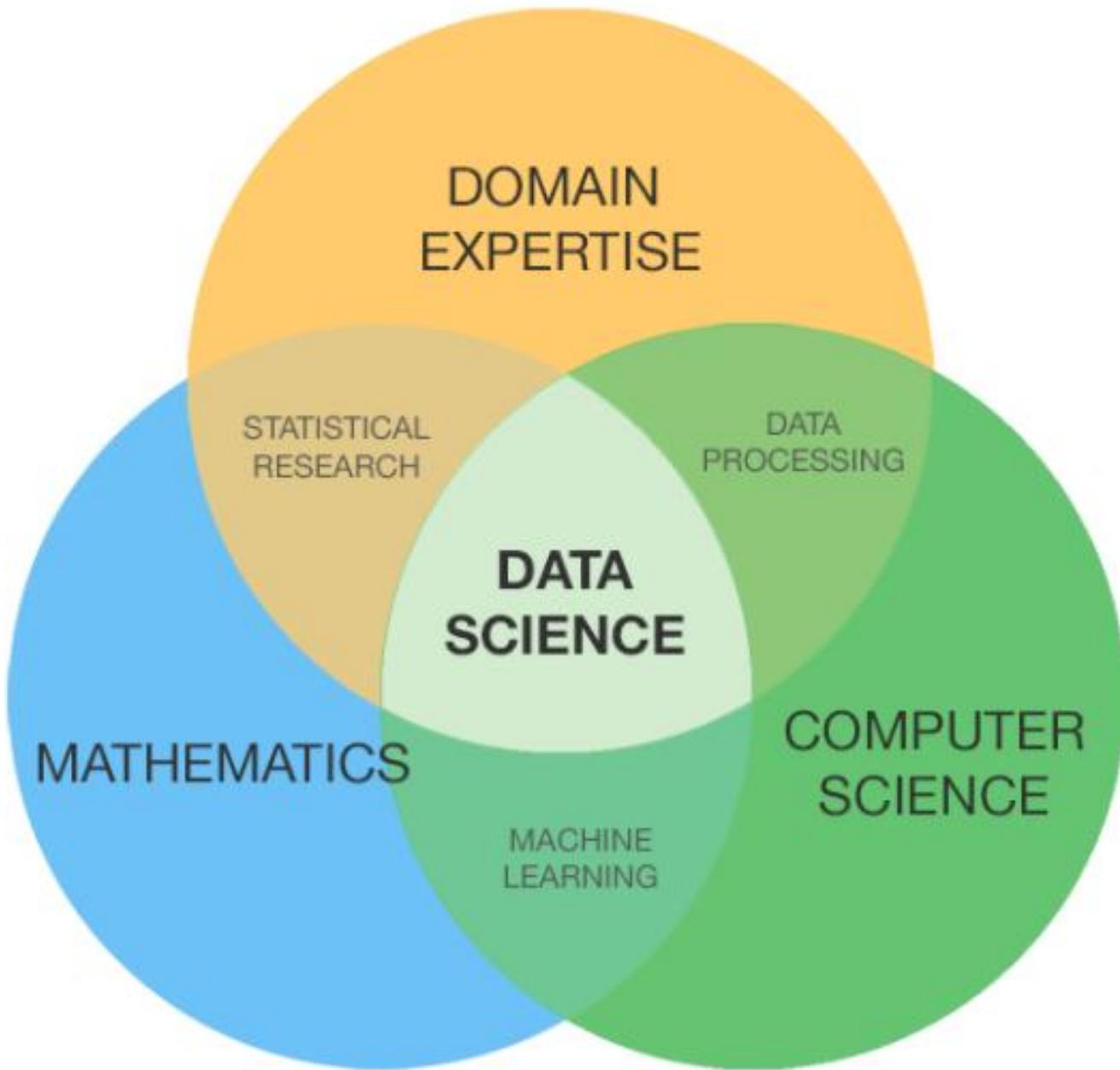
Computer Science

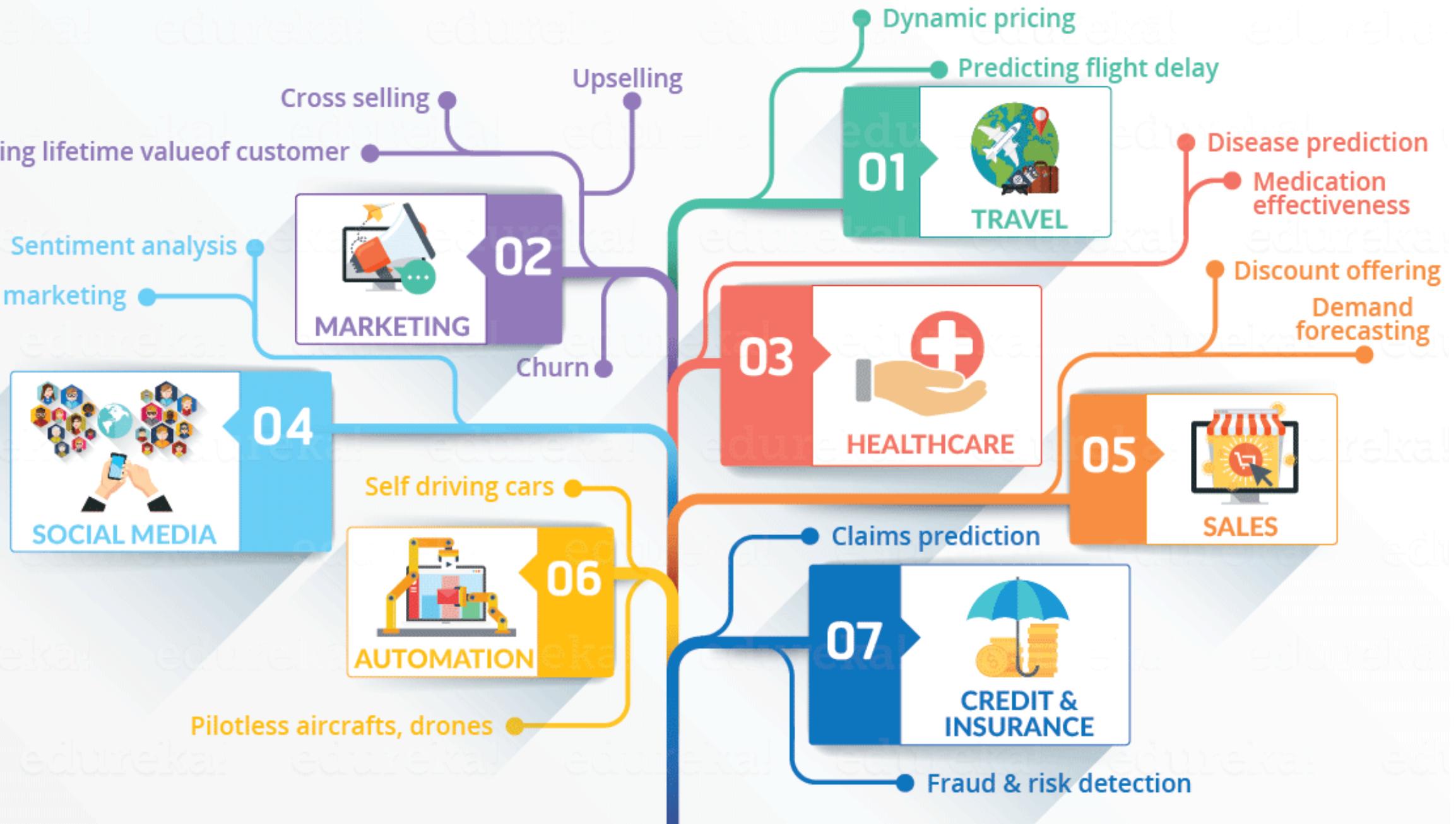
- The definition of computer science is a branch of **engineering science** that studies the technology and the principles of computers
- Computer science deals with the theoretical foundations of information and computation, together with practical techniques for the implementation and application of these foundations

THE COMPUTER ENGINEERING MAJOR











Java and Programming Languages

Lecture 3



24

What's your favorite IDE for Java Development?



Eclipse



IntelliJ IDEA



NetBeans



BlueJ



JDeveloper



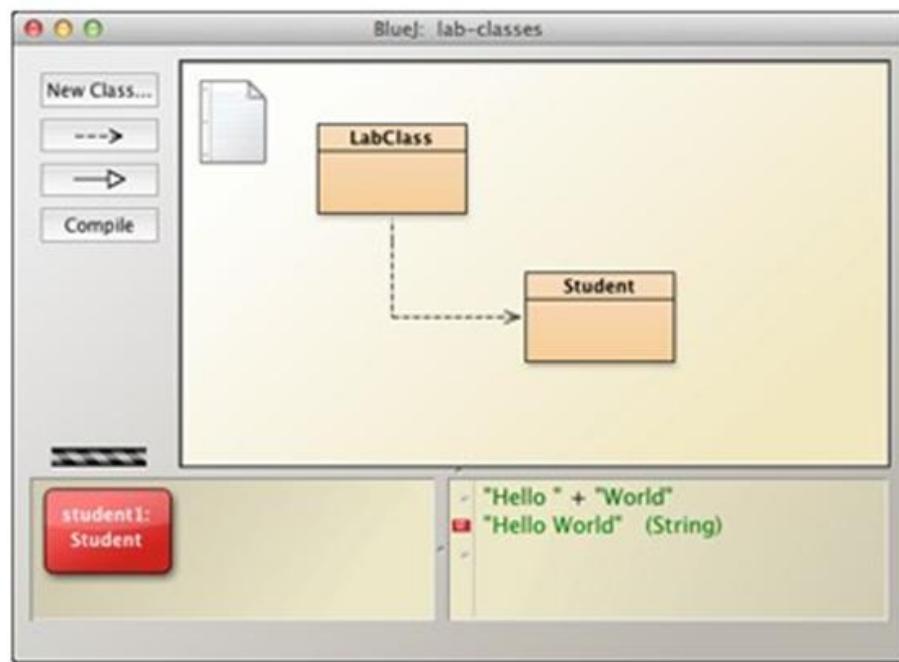
DrJava



Android Studio

Other





```
/**  
 * Add a student to this LabClass.  
 */  
public void enrollStudent(Student newStudent)  
{  
    if(students.size() == capacity) {  
        System.out.println("The class is full.");  
    }  
    else {  
        students.add(newStudent);  
    }  
}
```

BlueJ is Light-weighted Java IDE

bluej.org



1. Easy to install and easy to use. **Java JDK included.(Optional)**
2. BlueJ is as light-weighted as **Notepad** for text editing.
3. Java files created by BlueJ can all be edited by notepad.
4. You may copy or compress (zip) the java files from BlueJ and copy them to any location or put them on the web-site for download just like text file. No special import/export required.
5. **BlueJ files can go to Eclipse, Netbeans, IntelliJ and many other IDE tools.** But not the other way around. You cannot copy the Eclipse (or Netbeans, or IntelliJ) files to any other IDE without much struggling.
6. You can just copy to source code files (from Eclipse or any other tools) to a BlueJ project directory and continue to work without any trouble.
7. BlueJ has less overheads. It runs faster.
8. BlueJ has Class Diagram views.
9. BlueJ has object execution environment.
- 10.BlueJ can link the library much faster.



First Program

Hello World?

Lecture 4



In-Class Demonstration Program

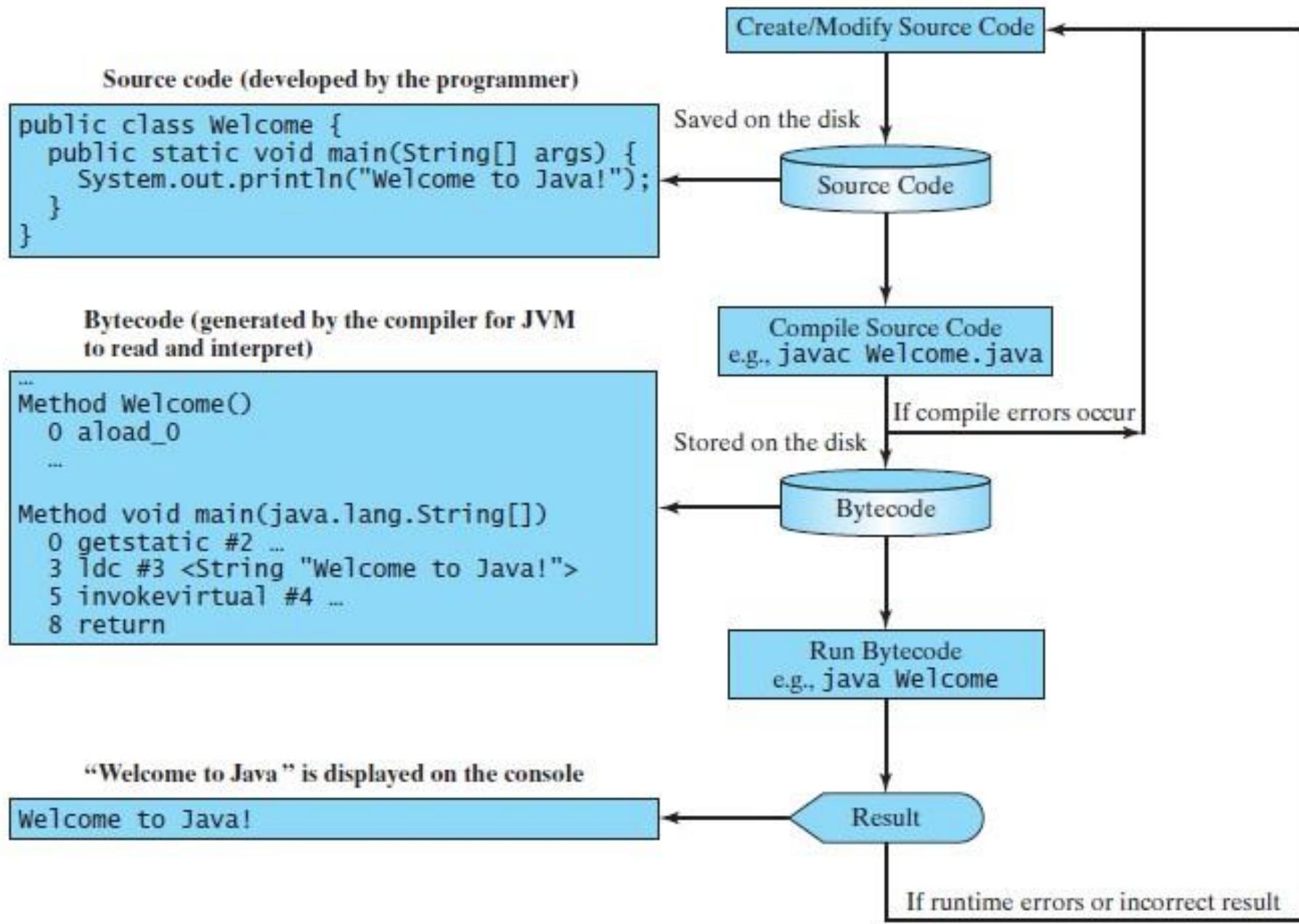
HelloWorld.java

Compile Undo Cut Copy Paste Find... Close

Source Code

```
public class HelloWorld
{
    static void hello() {
        System.out.println("Hello World!");
    }
} //===== end class HelloWorld =====
```

Class compiled – no syntax errors



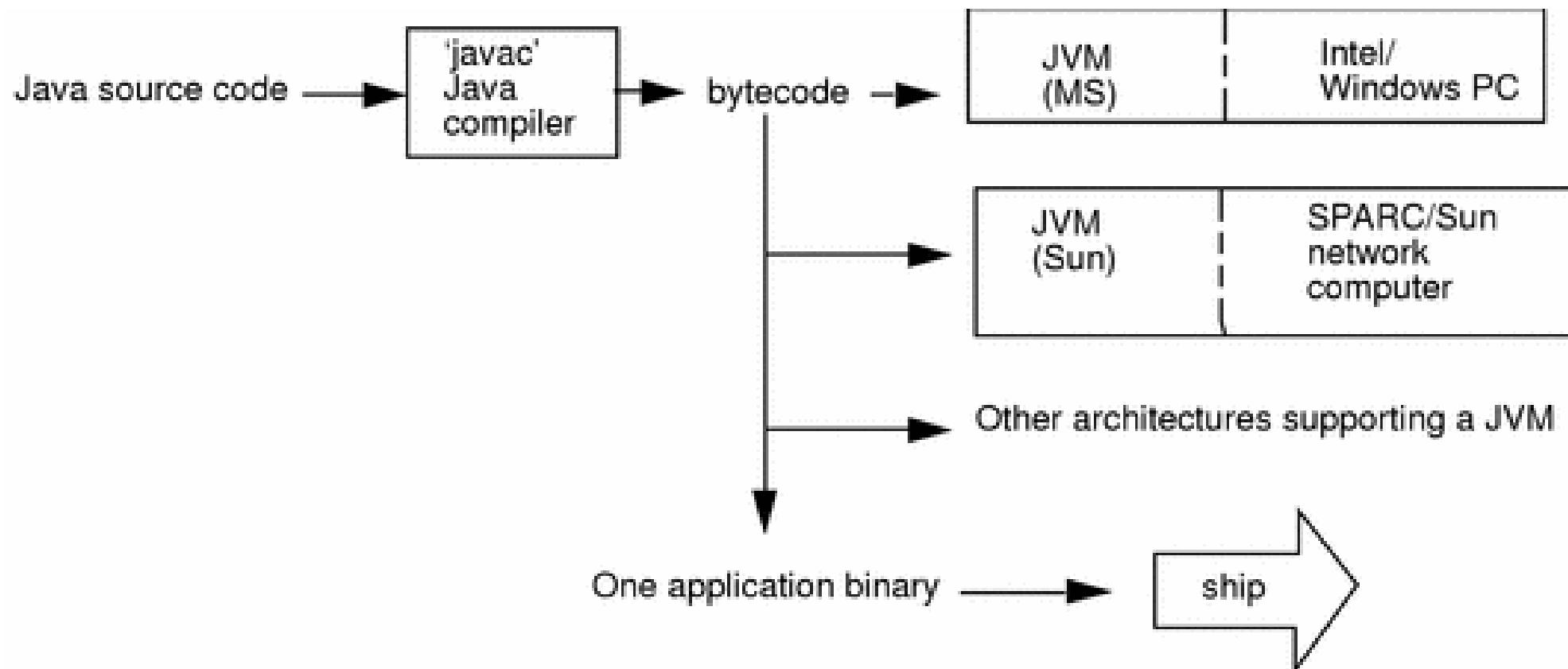
i Command Prompt

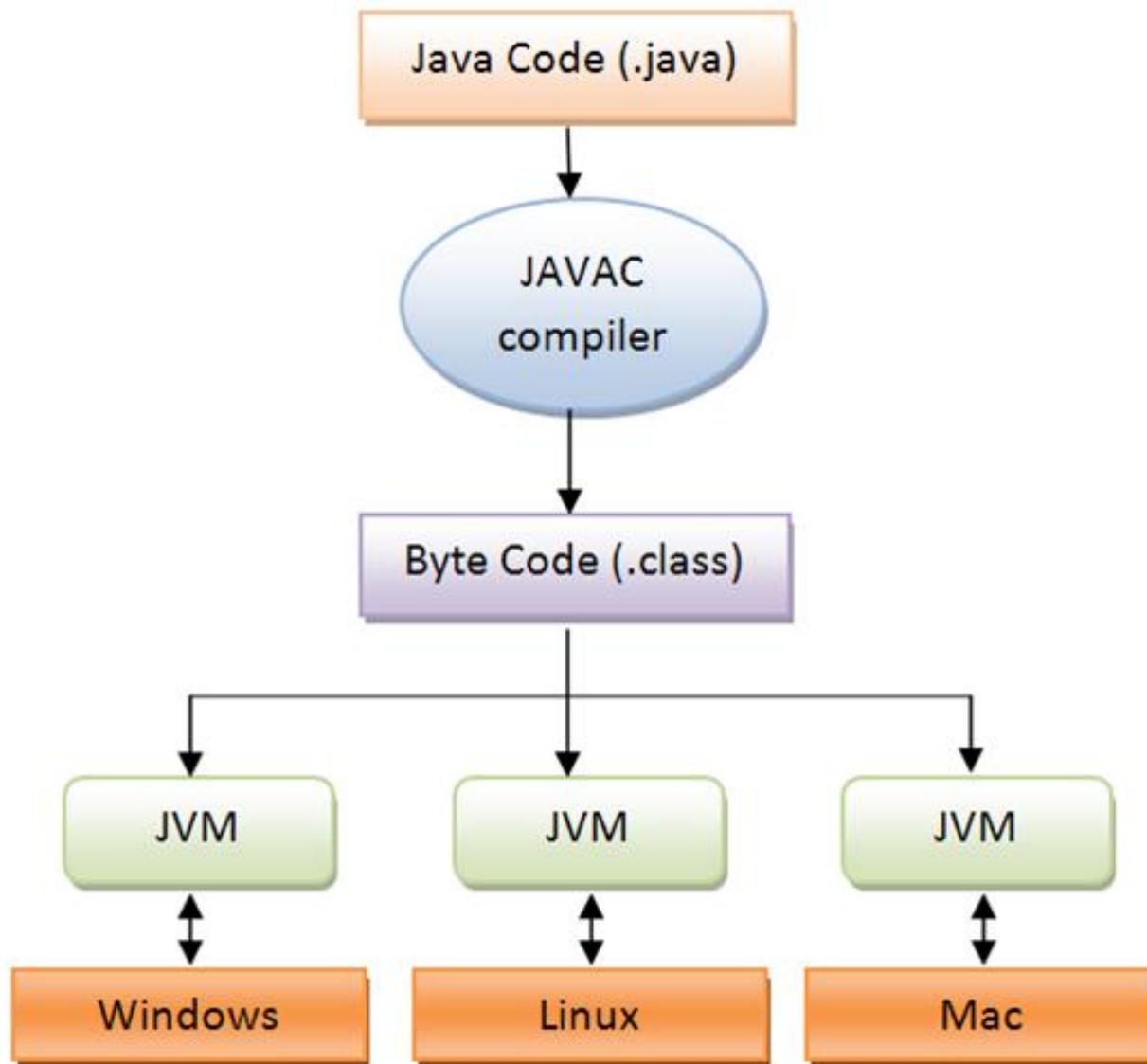
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\student>set Path=C:\Program Files\Java\jdk1.6.0_21\bin
C:\Users\student>set HomePath= C:\Program Files\Java\ jdk1.6.0_21
C:\Users\student>javac helloworld.java
C:\Users\student>java helloworld
Hello World!

C:\Users\student>
```

Cross-platform by JVM and bytecode







Java Language

Lecture 5

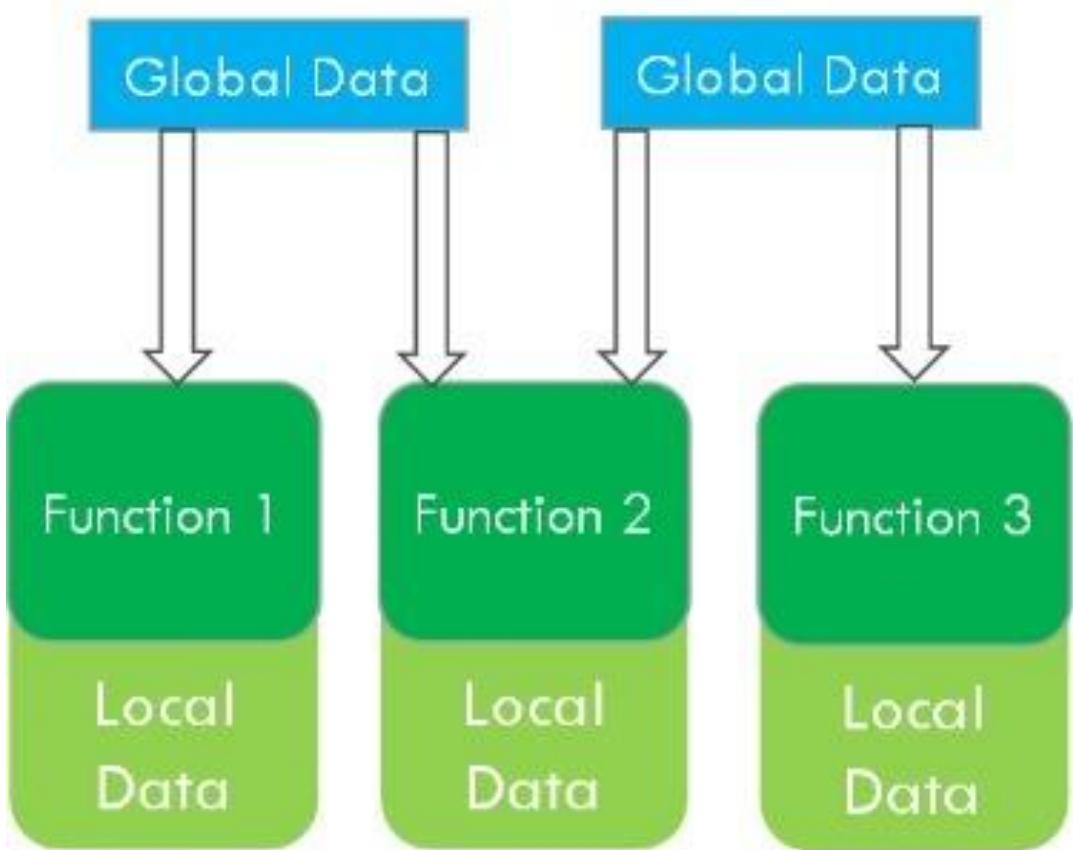
JAVA Programming Language

- Object-oriented programming is at the core of Java. In fact, **all Java programs are object-oriented**, this isn't an option the way that it is in C++, for example.
- OOP is so integral to Java that you must understand its basic principles before you can write even simple Java programs.

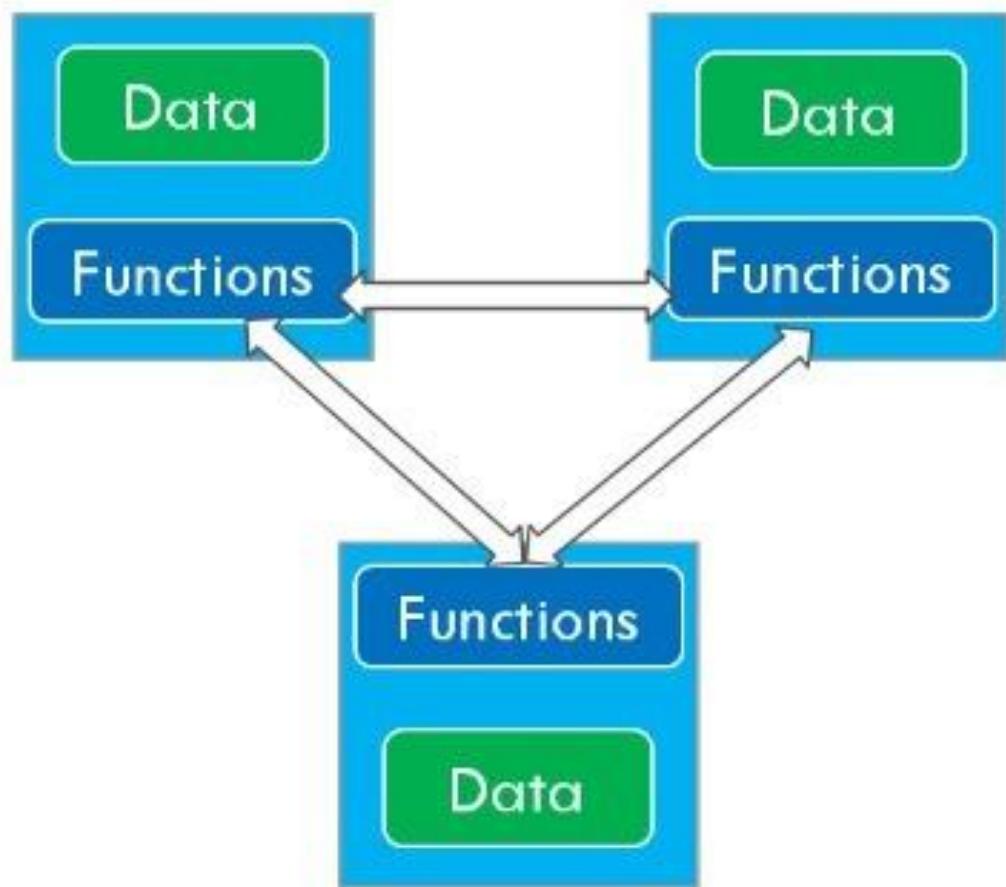
Java Programming

- Structured Programming (Procedural Programming)
- Object-Oriented Programming
- Event-Driven Programming

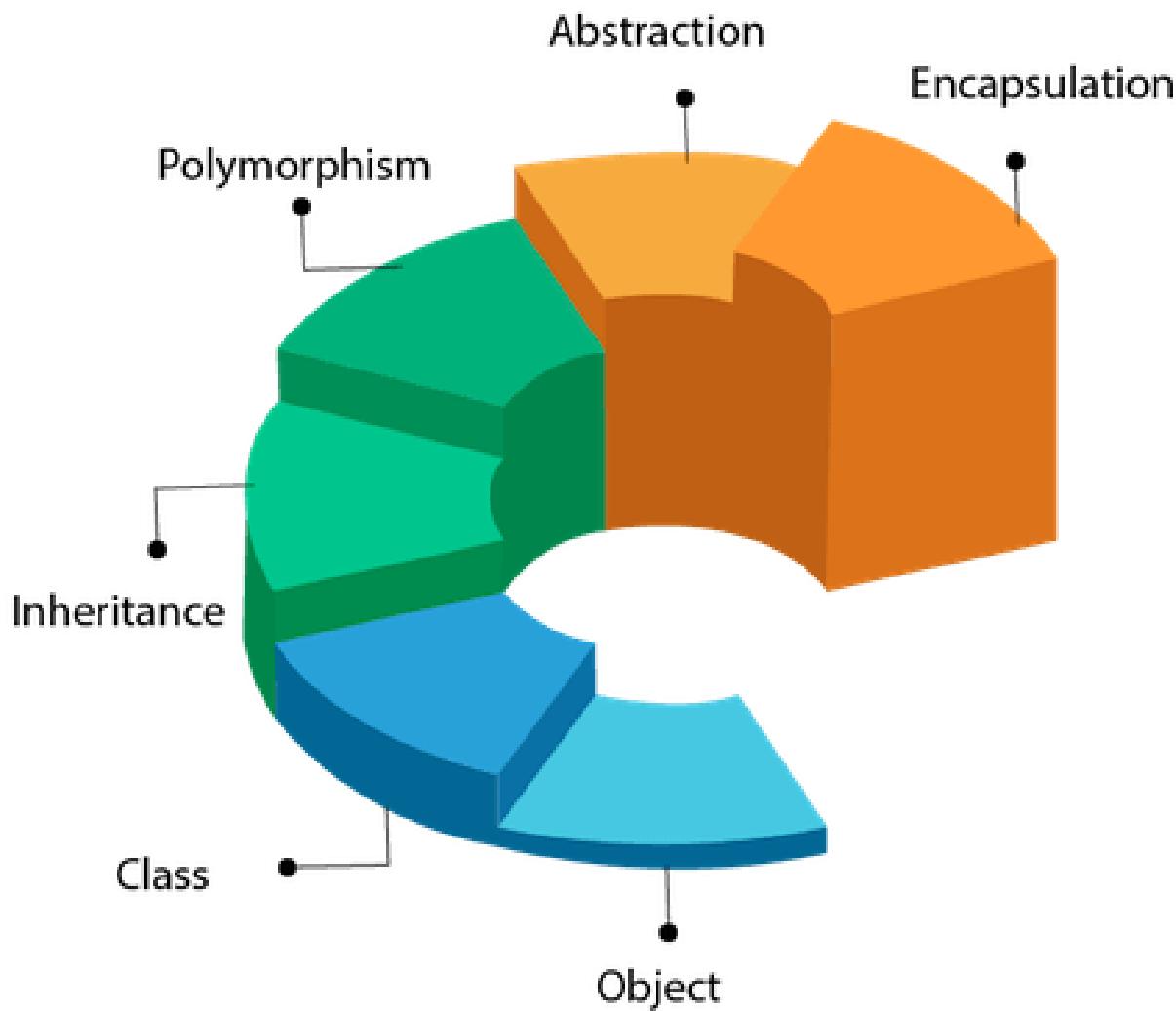
Procedural Oriented Programming

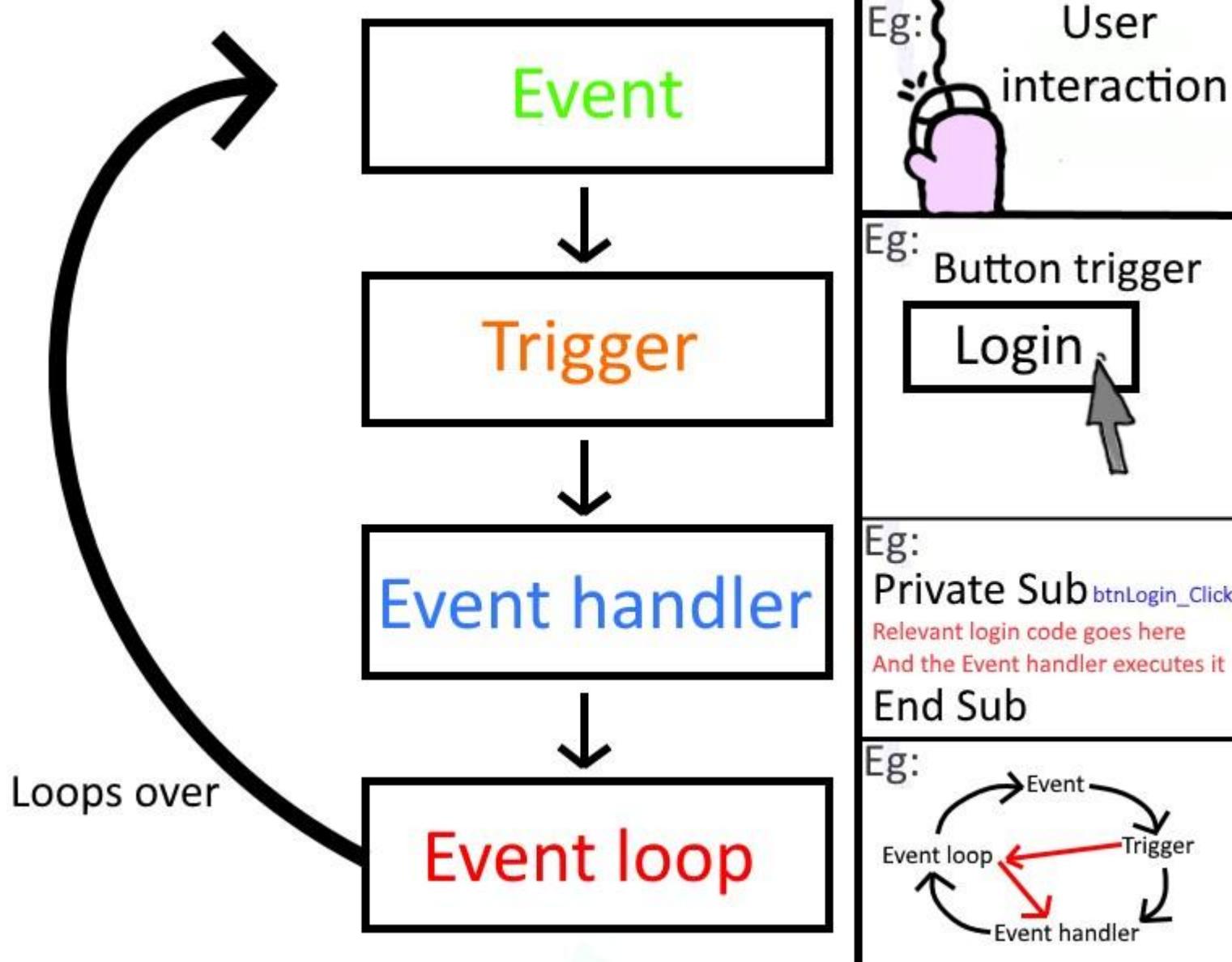


Object Oriented Programming



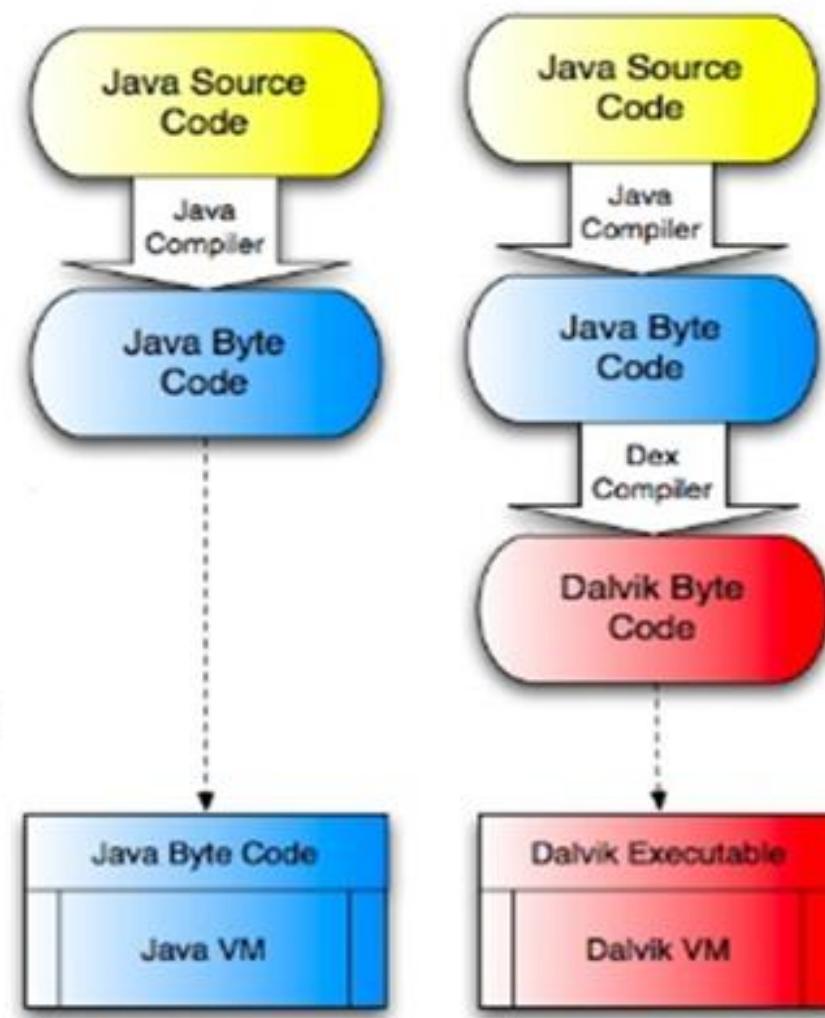
OOPs (Object-Oriented Programming System)



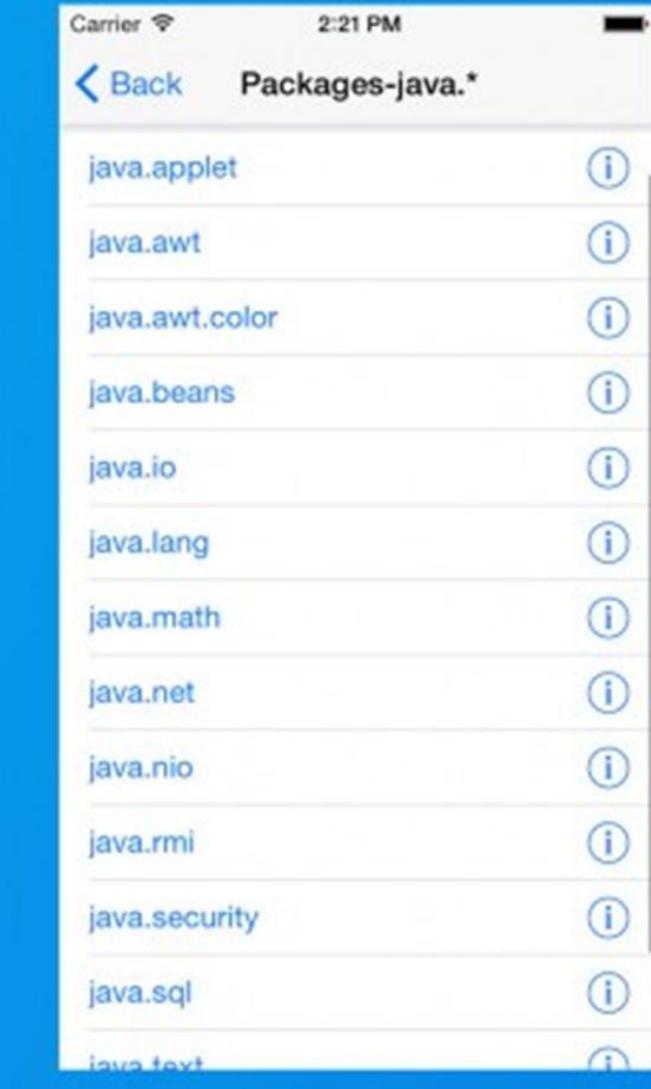
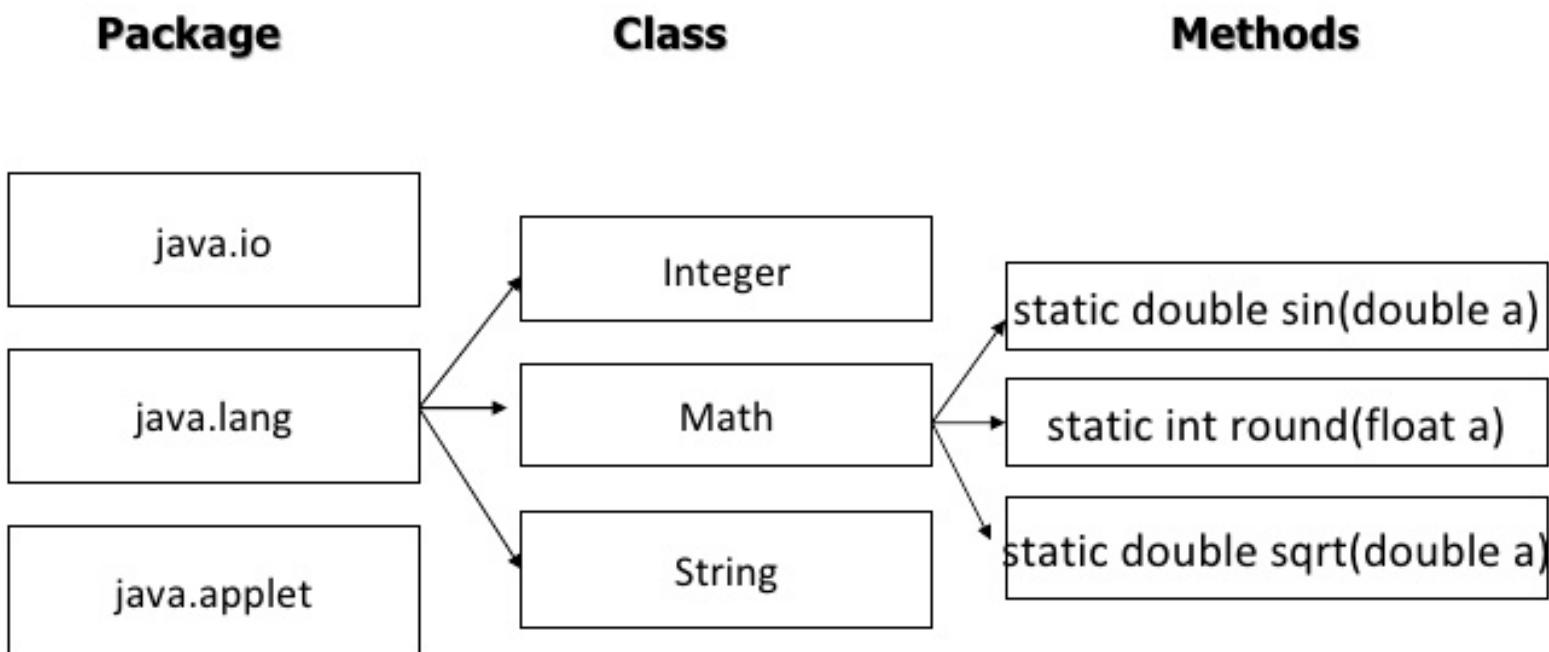


Android and Java

Android Java =
Java SE –
AWT/Swing +
Android API



Structure of the Java API

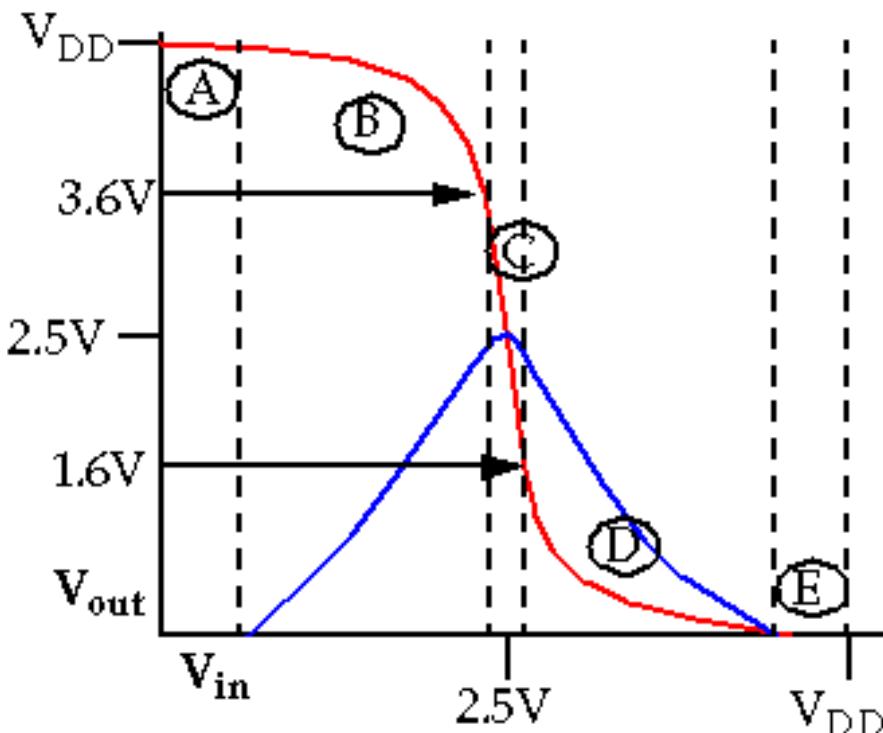
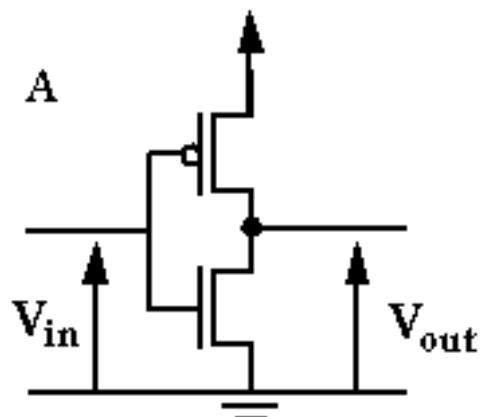




Interpretation Levels

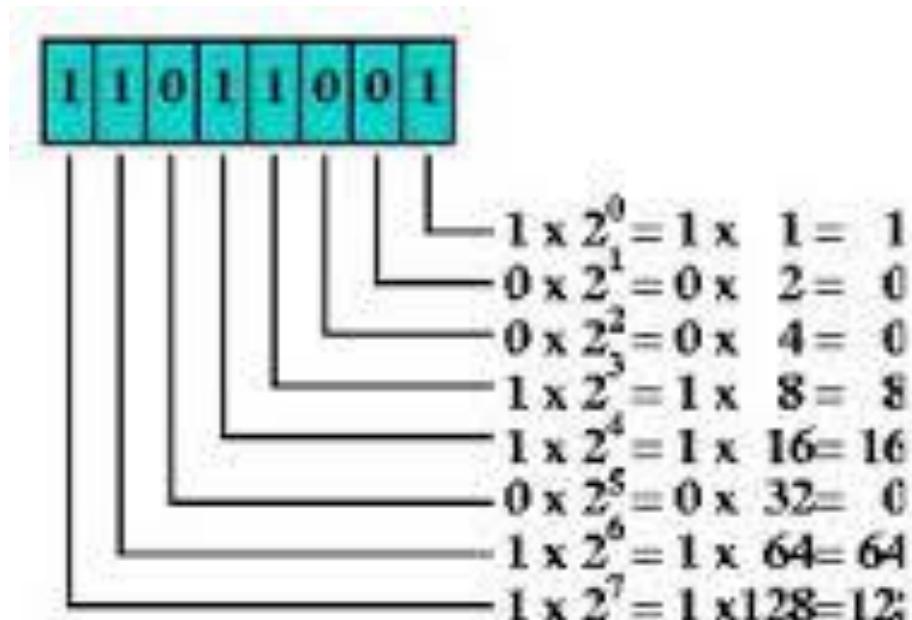
Lecture 6

Binary Digit (Bit)	Electronic Charge	Electronic System
1		ON
0		OFF



- (A) $0 \leq V_{in} \leq V_{tn}$;n-device is cut off ($I_{dsn}=0$), p-device in linear.
- (B) $V_{tn} < V_{in} < V_{DD}/2 - \Delta$;n-device is in sat., p-device in linear.
- (C) $V_{DD}/2 - \Delta \leq V_{in} \leq V_{DD}/2 + \Delta$;n-device is in sat., p-device in sat.
- (D) $V_{DD}/2 + \Delta < V_{in} < V_{DD} + V_{tp}$;n-device is in linear, p-device in sat.
- (E) $V_{DD} + V_{tp} \leq V_{in} \leq V_{DD}$;n-device is in linear, p-device in cut off ($I_{dsp}=0$).

Binary Number System



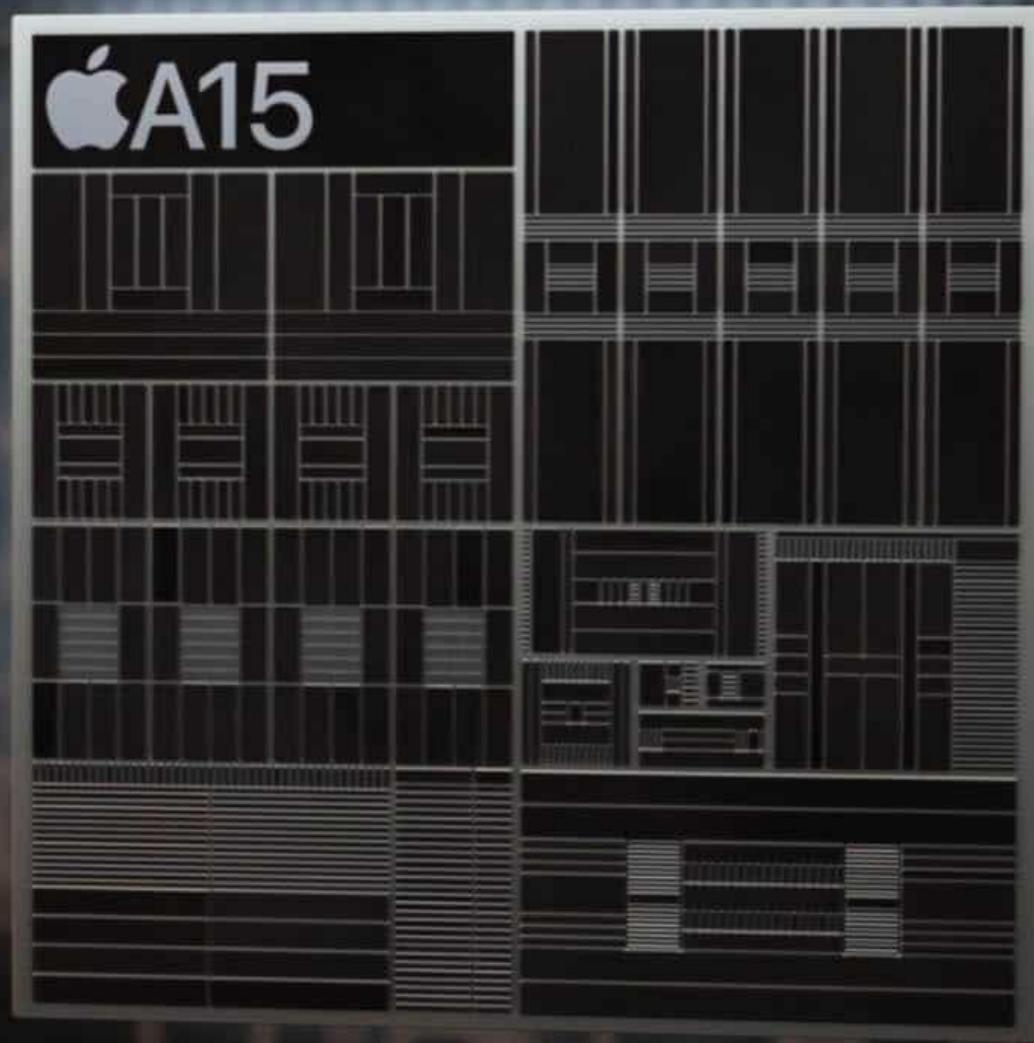
$$1 + 8 + 16 + 64 + 128 = 217$$

FASTER CPU

NEW 5-CORE GPU

NEW DISPLAY ENGINE

FASTER NEURAL ENGINE



ProRes

encode and
decode



Thunderbolt 4



Secure Enclave



Support for two external displays

33.7 billion
Transistors

16-core

**Neural
Engine**

11 trillion operations per second

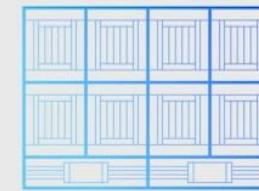
Industry-leading
performance per watt

5 nm process

Up to

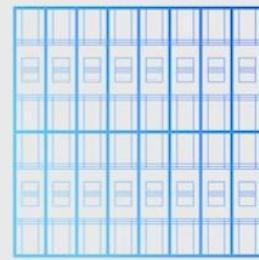
32GB

Unified memory



Up to

10-core
CPU



Up to

16-core
GPU

200GB/s
Memory bandwidth

ProRes

encode and
decode



Thunderbolt 4



Secure Enclave



Support for four external displays

64GB

Unified memory

57 billion

Transistors

16-core

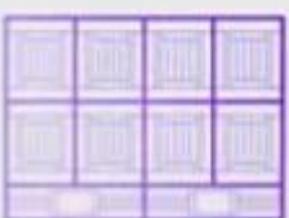
**Neural
Engine**

11 trillion operations per second

Industry-leading
performance per watt



5 nm process



10-core
CPU



32-core
GPU

400GB/s

Memory bandwidth

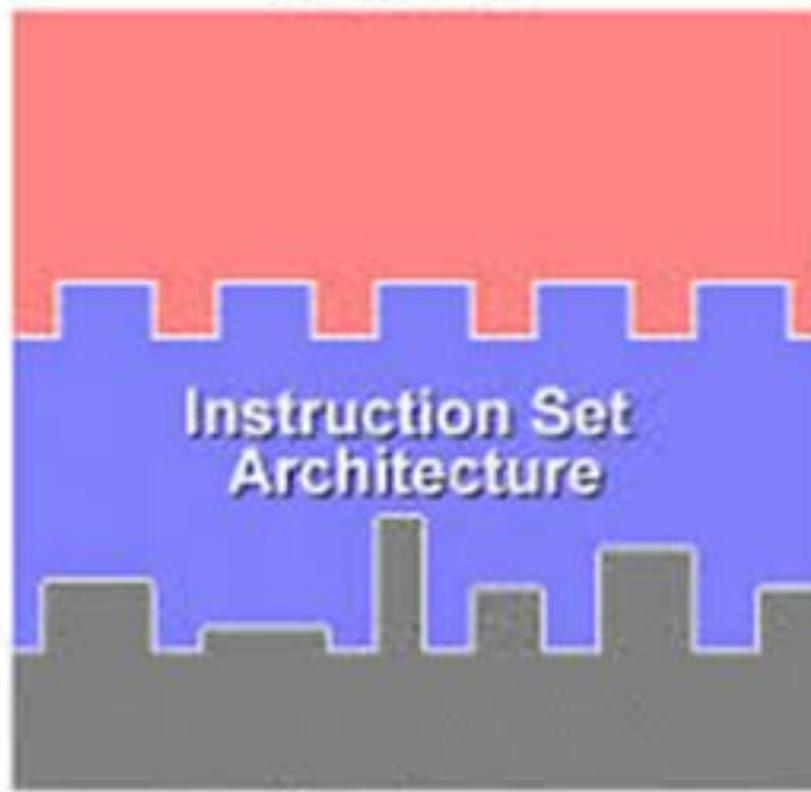
Programming Languages

Machine Language Assembly Language High-Level Language

Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Program with native machine language is a tedious process. Moreover the programs are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:

1101101010011010

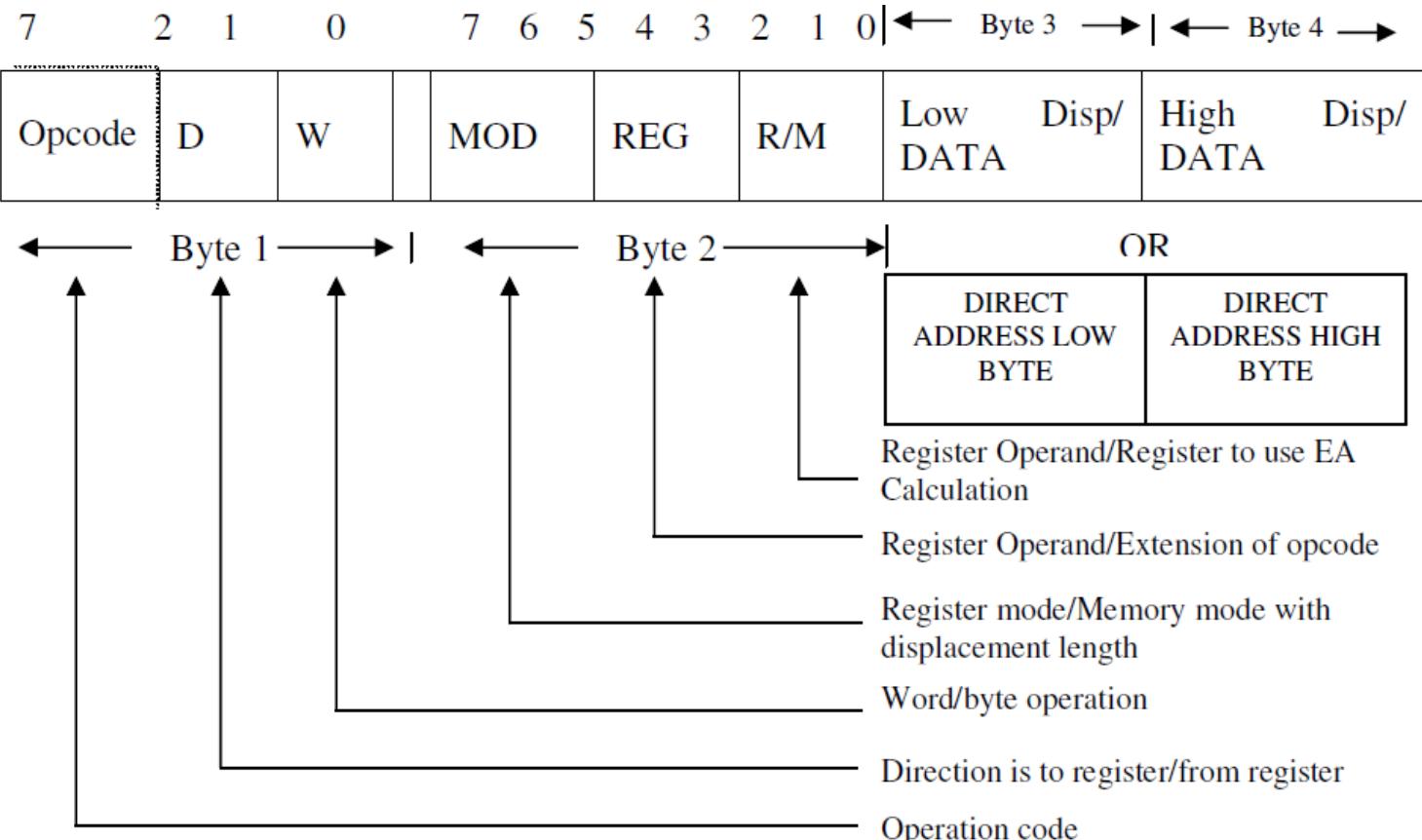
Software



Hardware

Machine Code (Instructions)

The whole legal collection of instructions is called instruction set



Instruction

Instruction

Operand

Instruction

Operand

Operand

mov

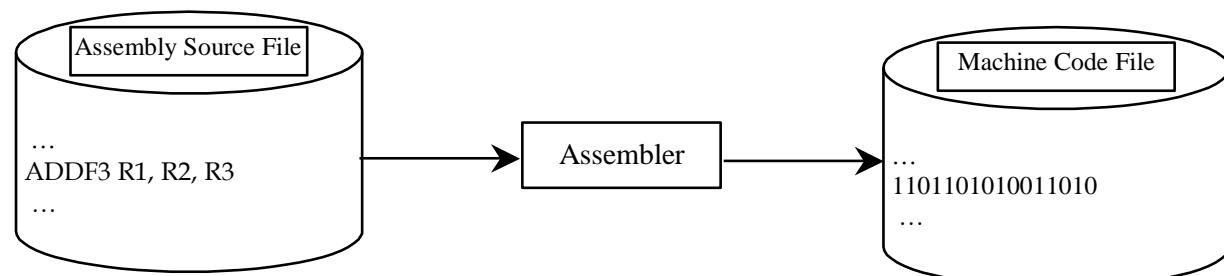
destination, source

Programming Languages

Machine Language **Assembly Language** High-Level Language

Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3





8086 Emulator & Assembler

File Edit Assemble Devices



New



Open



Save



Assemble Emulate



Calculator



Convertor

Options

```
03 ; COM file is loaded at CS:0100h
04 ORG 100h
05
06 LEA BX, message
07 MOV CX, 27d ; Length of message
08 MOV AX, 0h ; Ensure top and bottom of
09 ; ax empty
10
11 spit:
12 MOV AL, [BX] ; Put char into al
13 OUT 130d, AL ; push char out port
14 ; (ie. into printer)
15
16 INC BX ; inc pointer
17
18 wait1: ; Loop to ensure the printer
19 IN AL, 130d ; is ready, it clears
20 OR AL, 0 ; the port when this is true.
21 JNZ wait1
22
22 loop wait ; Go back and repeat if more
```

8086 Assembly language.

- The language that can be directly translated to machine code.
- LEA is an opcode. BX is a register name. Message is a pointer to a string.
- MOV is another opcode. CX is another register name. 27d is decimal 27.
- Assembly code is emulated by an emulator. And, there is debugger to help remove coding errors.
- Assembly code is used for **device driver or O.S. Kernels**.

Assembler and Debugger

The screenshot shows the DOSBox interface with the title bar "DOSBox 0.74, Cpu speed: 4000 cycles, Frameskip 0, Program: INSIGHT". The menu bar includes File, Edit, Run, Breakpoints, Options, and Window. The main window displays assembly code and memory dump.

Assembly Code:

```
0198:00133496 add    [Dsp+01+8634].ah
0199:00133497 xor    ax,ax
019A:00133498 mov    ah,00
019B:00133499 mov    ax,[2800]
019C:0013349A C3      ret
019D:0013349B mov    es,[0658]
019E:0013349C mov    si,[0644]
019F:0013349D mov    al,es:[si]
01A0:0013349E mov    ah,41
01A1:0013349F and    al,27
01A2:001334A0 cmp    al,26
01A3:001334A1 jne    01A2
01A4:001334A2 49
01A5:001334A3 inc    si
01A6:001334A4 mov    ax,es:[si]
01A7:001334A5 cmp    al,09
01A8:001334A6 jb    01A7
01A9:001334A7 cmp    al,8C
01A9:001334A8 33:0000-0016 = 14
```

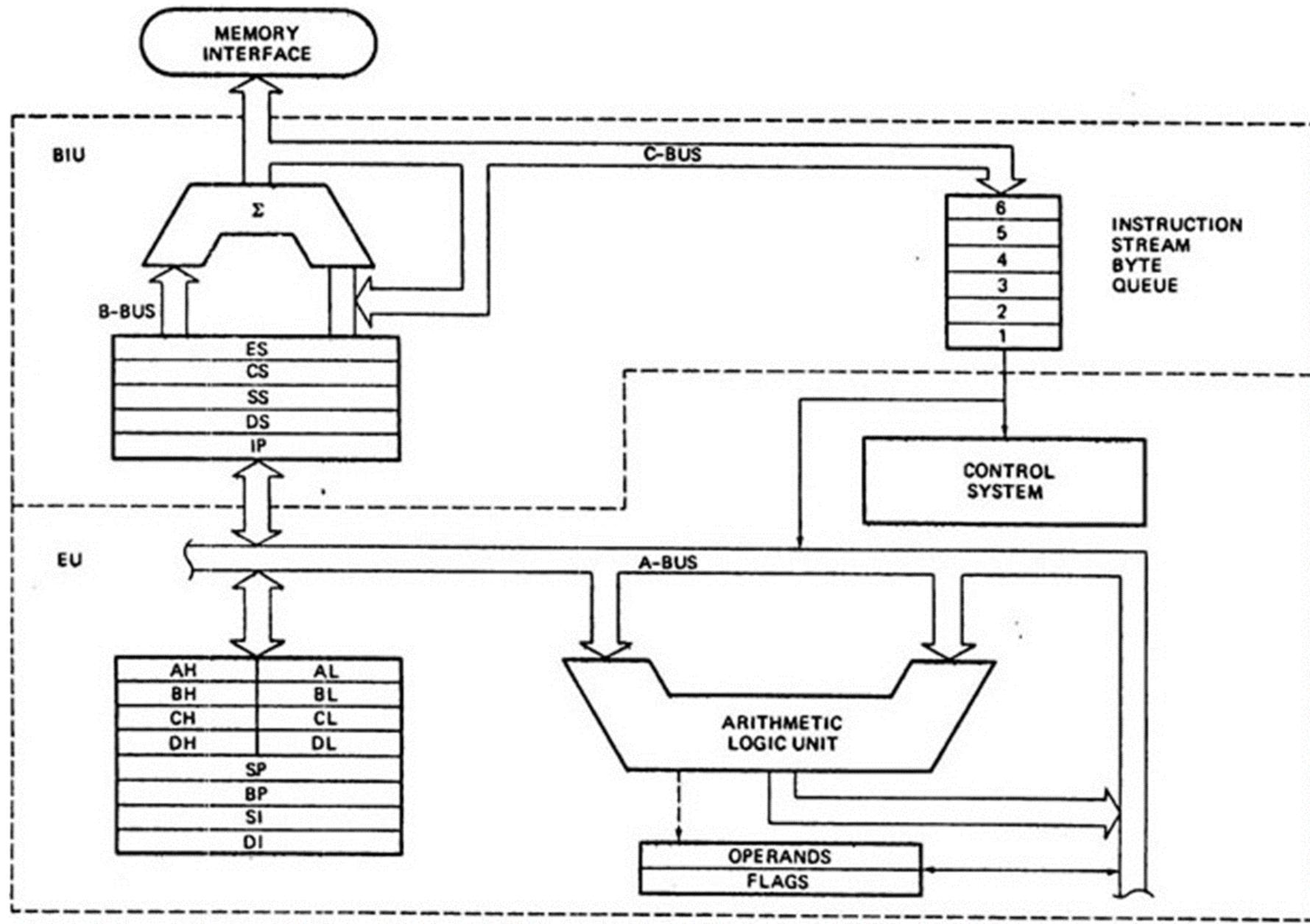
Registers:

00:0000	01:FFFF	15:0016					
CD:00FF	17:001C	15:0016					
00:0016	3F:FFFF	33:0016					
IP:01A9 Flags=7282							
OF	DF	IF	SF	ZF	NF	PF	CF
0	0	1	0	0	0	0	0

Memory Dump:

0016:0000	CD 29 FF 3F 00 EA FF FF AD DC A1 00 93 4E 00 00	- f.9 41 10 8c ..
0016:0010	18 01 10 01 18 01 93 01 01 01 01 00 02 FF FF FF	10-0202000000 0
0016:0020	FF 18 00 01 87	>J%g
0016:0030	93 01 14 00 18 00 16 00 FF FF FF FF 00 00 00 00	0201 1 -f ..
0016:0040	05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	+

- **Assembler** is the software to convert Assembly Language into Machine code.
- **Emulator** is to emulate the assembly code on real hardware to know about what will be the outcome.
- **Debugger** is a software to show the error and contents for each registers and memory.



8086 Instruction Set Architecture

- Machine code is the real code for machine. It is used to control the Arithmetic and Logic Unit (ALU) and the register file and the memory.
- All programs are eventually executed in machine code.
- No one programs on Machine code.

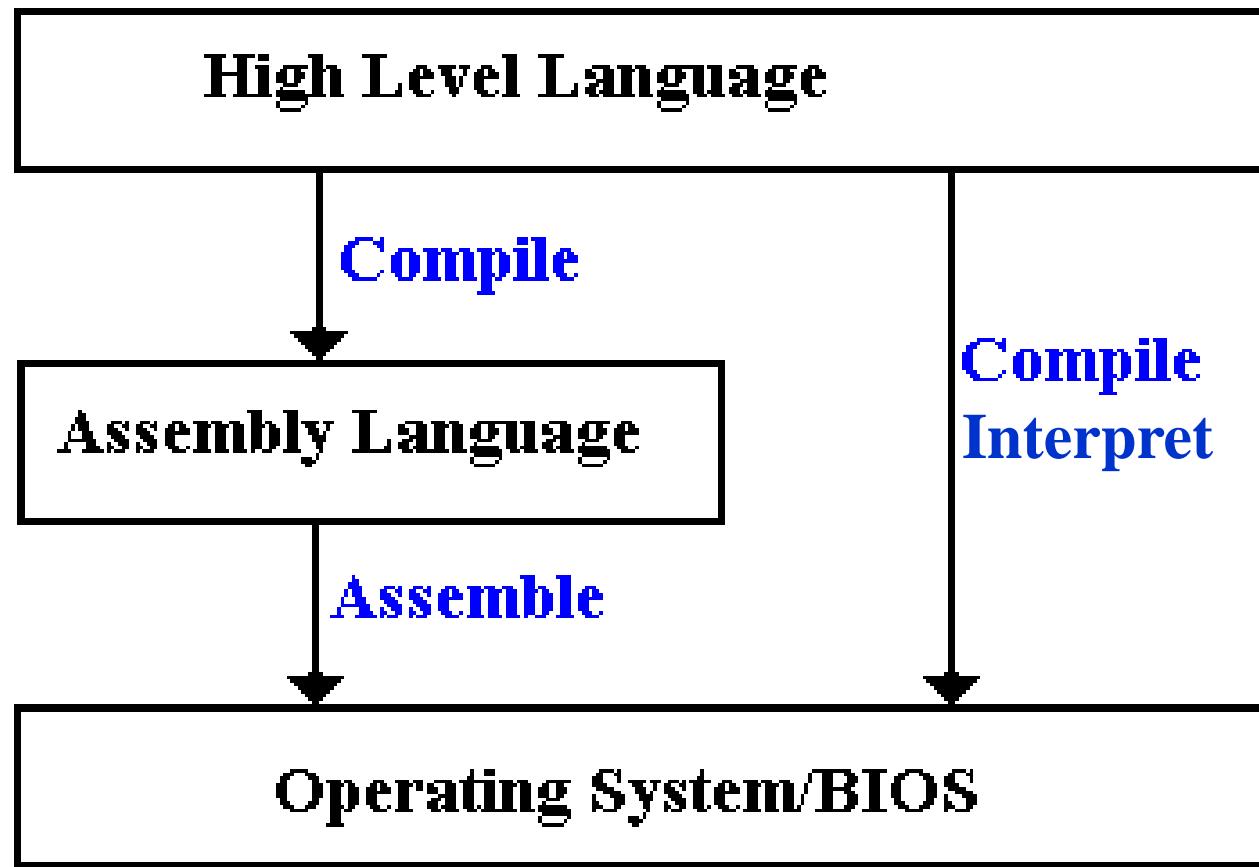
Programming Languages

Machine Language Assembly Language **High-Level Language**

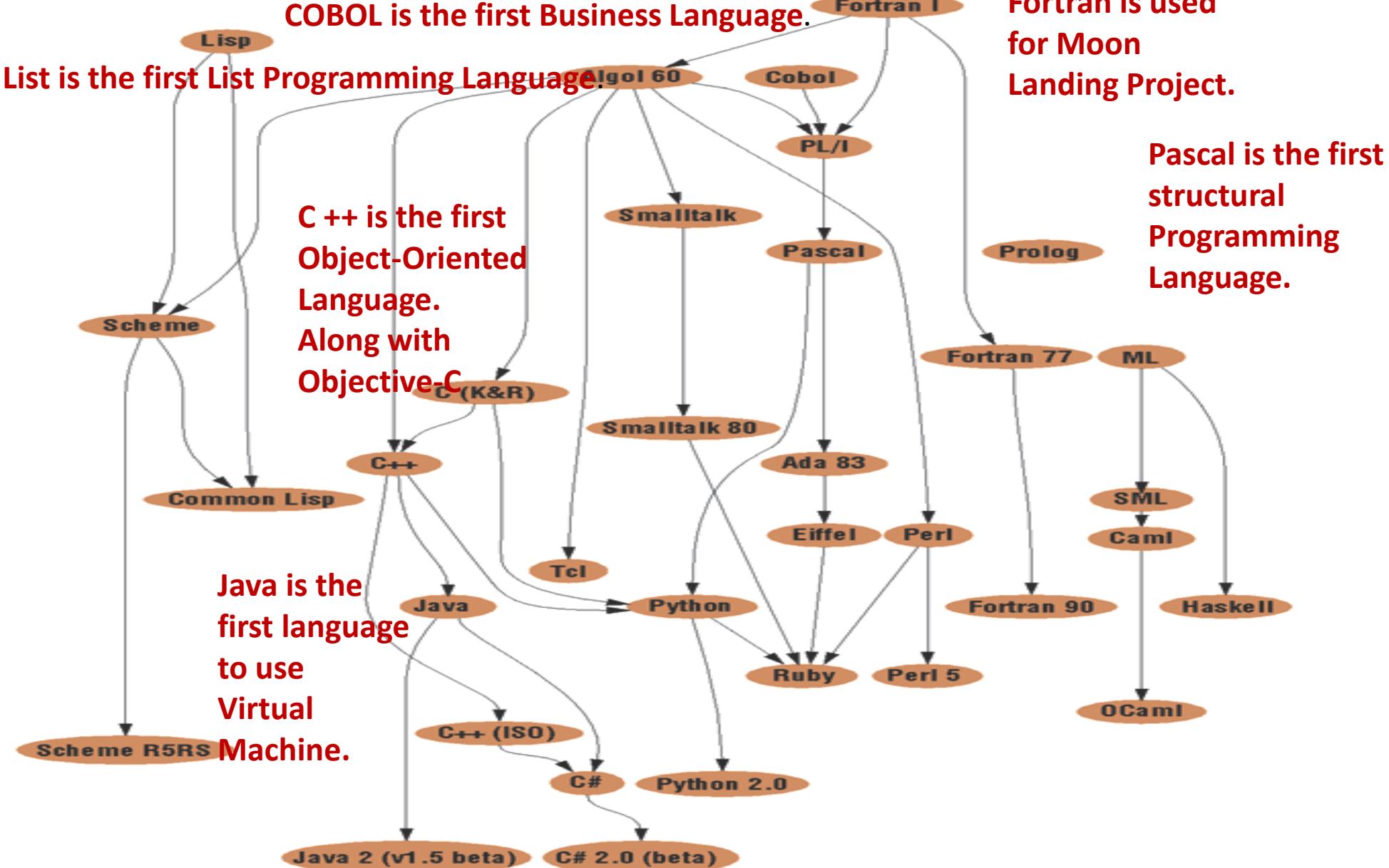
The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

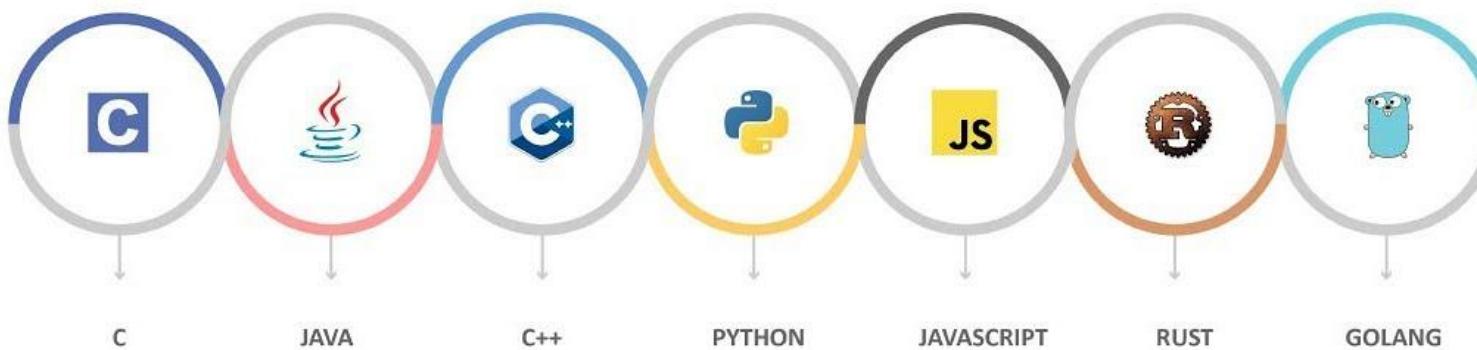
High Level Languages



1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004



Most Popular programming languages 2018-2019





Java Knowledge

Lecture 7



Java Comments



Demonstration Program

OnePlusOne.java

Program Structure

- Demonstrate basic Java program parts using OnePlusOne.java
- **Basic Program Structure:**

```
public static void main(String[] args) {
    // Input -----
        // code for input here.
    // Processing -----
        // code for processing here.
    // Output -----
        // code for output here.
}
```

Java comments and code block marks

Characters	Name	Description
//	Double Slash	Line comment
/* */	Slash star and star slash	Opening and closing of comment text
/** */	Slash double-star and star-slash	Opening and closing of Javadoc comment text Javadoc comment can be extracted into HTML file using the JDK's Javadoc command (Use to describe a module, a method or a variable)
{ }	Braces	For a code block.
[]	brackets	For the index variable
()	parenthesis	For the boundary of an expression or a logic conditions
“ “	double quotes	For boundary of a string of text data



Java Doc

What is JavaDoc?

- *Doc comments* (also known informally as *Javadoc comments*, although this technically violates trademark usage) document your APIs for other programmers who use your classes and for maintenance programmers.
- Doc comments standardize the way source code is documented.
- Documentation is kept in the source file, so it's easier for developers to keep it in sync with the code.
- You can document packages, classes, constructors, fields, and methods.

AP Computer Science Part X New Tab Overview (Java Platform S... X

<https://docs.oracle.com/javase/8/docs/api/>

Apps Bookmarks Caller Center Google 15 Google Calendar - ... Facebook YouTube Yahoo Google Translate On-line School School Other bookmarks

Java™ Platform Standard Ed. 8 Java™ Platform Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color

All Classes

AbstractAction
AbstractAnnotationValueVisitor
AbstractAnnotationValueVisitor
AbstractAnnotationValueVisitor
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeCo
AbstractDocument.Content
AbstractDocument.ElementEdi
AbstractElementVisitor6
AbstractElementVisitor7

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its

Defining JavaDoc

- Use `/** * /` comments right before the entities that are to be documented.
- You can have whitespace between the doc comment and the declaration, but no other code. For example, don't put import statements between your doc comment and a class declaration.
- If a doc comment line begins with a * preceded by optional whitespace, those characters are ignored.
- As of Java 1.4, leading whitespace is preserved if a line does not begin with a * character. This allows you to include formatted code fragments (wrapped with HTML `<pre>` tags) in your documentation.

Defining JavaDoc

- A doc comment consists of an optional main description followed by an optional tag section.
- **A doc comment can contain HTML markup**, but keep it simple (as in, keep it `simple`).
- The first sentence of the main description (ending in a period followed by a space, tab, line terminator, or first block tag) is used as the summary description for the declared entity.

Doc Comment Tags

Block tags have the format `@tag description`. There are many block tags available, but the more commonly used ones are:

`@author name` Author Name (class/interface only)

`@version major.minor.patch` Version number (class/interface only)

`@param name description` Description of parameter (method only)

`@return description` Description (method only)

`@throws Throwable description` Description of exception (exceptions are discussed in the next module)

`@deprecated explanation` Explanation (method only)

`@see package.class#member` label A hyperlink to a reference package/class/member or field.

See demo and command line demo ...



AP Subset

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Comments /* */, //, and /** */ Javadoc @param and @return comment tags		Javadoc tool
Primitive Types int, double, boolean		char, byte, short, long, float
Operators Arithmetic: +, -, *, /, % Increment/Decrement: ++, -- Assignment: =, +=, -=, *=, /=, %= Relational: ==, !=, <, <=, >, >= Logical: !, &&, Numeric casts: (int), (double) String concatenation: +	1, 2, 3, 4, 5	&, , ^ (char), (float) StringBuilder Shift: <<, >>, >>> Bitwise: ~, &, , ^ Conditional: ?:
Object Comparison object identity (==, !=) vs. object equality (equals), String compareTo		implementation of equals Comparable
Escape Sequences \", \\, \n inside strings		\', \t, \unnnn
Input / Output System.out.print, System.out.println	6	Scanner, System.in, System.out, System.err, Stream input/output, GUI input/output, parsing input: Integer.parseInt, Double.parseDouble formatting output: System.out.printf

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Exceptions ArithmaticException, NullPointerException, IndexOutOfBoundsException, ArrayIndexOutOfBoundsException, IllegalArgumentException		try/catch/finally throw, throws assert
Arrays 1-dimensional arrays, 2-dimensional rectangular arrays, initializer list: { ... }, row-major order of 2-dimensional array elements	7, 8	new <i>type</i> [] { ... } , ragged arrays (non-rectangular), arrays with 3 or more dimensions
Control Statements if, if/else, while, for, enhanced for (for-each), return		switch, break, continue, do-while
Variables parameter variables, local variables, private instance variables: visibility (private) static (class) variables: visibility (public, private), final		final parameter variables, final local variables, final instance variables
Methods visibility (public, private), static, non-static, method signatures, overloading, overriding, parameter passing	9, 10	visibility (protected), public static void main(String[] args), command line arguments, variable number of parameters, final
Constructors super(), super(args)	11, 12	default initialization of instance variables, initialization blocks, this(args)

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Classes new, visibility (public), accessor methods, modifier (mutator) methods Design/create/modify class. Create subclass of a superclass (abstract, non-abstract). Create class that implements an interface.	13, 14	final, visibility (private, protected), nested classes, inner classes, enumerations
Interfaces Design/create/modify an interface.	13, 14	
Inheritance Understand inheritance hierarchies. Design/create/modify subclasses. Design/create/modify classes that implement interfaces.		
Packages <code>import packageName.className</code>		<code>import packageName.*</code> , static import, package <code>packageName</code> , class path
Miscellaneous OOP “is-a” and “has-a” relationships, null, this, <code>super.method(args)</code>	15, 16	<code>instanceof</code> (class) cast <code>this.var</code> , <code>this.method(args)</code> ,
Standard Java Library <code>Object</code> , <code>Integer</code> , <code>Double</code> , <code>String</code> , <code>Math</code> , <code>List<E></code> , <code>ArrayList<E></code>	17, 18	<code>clone</code> , autoboxing, <code>Collection<E></code> , <code>Arrays</code> , <code>Collections</code>



Java Coding Habits

In Java, a class definition starts with the word “class” (sometimes preceded by “public” or “private”).

A method definition starts with naming information.

```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello world!");  
4     }  
5 }
```

Braces in Java are like indentations in Python or parentheses in Scheme.

Class name

Inside the class braces are definitions of methods and variables.

Statements are separated with semicolons.

Methods also have a pair of braces that include the statements of the method.

```
public void RefreshCatalog()
{
    if (IsCacheValid)
    {
        ResetFilterToDefaults();
    }
    else
    {
        RepopulateCatalogFromService();
    }
}
```

```
public void RefreshCatalog() {
    if (IsCacheValid) {
        ResetFilterToDefaults();
    }
    else {
        RepopulateCatalogFromService();
    }
}
```

A space between parameters

```
function pow(x, n) {  
    let result = 1;  
    for (let i = 0; i < n; i++) {  
        result *= x;  
    }  
  
    return result;  
}  
  
let x = prompt("x?", "");  
let n = prompt("n?", "");  
if (n < 0) {  
    alert(`Power ${n} is not supported,  
    please enter an integer number, greater than 0`);  
} else {  
    alert(pow(x, n));  
}
```

No space
between the function name and the bracket
between the bracket and the parameter

Figure bracket {
on the same line, after a space

Indentation
2 spaces

A space
after for/if/while...

An empty line
between logical blocks

Spaces
around operators

A semicolon ;
is mandatory

A space
between
parameters

Lines are not very long

} else { without a line break

Spaces around a nested call

Java Standard Naming Conventions

Package Name - A package should be named in lowercase characters.

Class Name - Class names should be nouns in UpperCamelCase.

Interface Name - Interface name should start with an uppercase letter and be an adjective.

Method Name - Methods should be verbs and in lowerCamelCase.

Variable Name - Variable name should in lowerCamelCase.

Constant Variable - Constant variable names should be written in upper characters separated by underscores.

Abstract Class Name - Abstract class name must start with Abstract or Base prefix.

Exception Class Name - Exception class name must end with Exception suffix

Code Listing 48: Naming Conventions in Java

Avoid	Preferable
<pre>1. class icecream{ 2. int flavourtype; 3. final int size=2; 4. void getflavourtype () { 5. return flavourtype; 6. } 7. }</pre>	<pre>1. class IceCream{ 2. int flavourType; 3. final int SIZE=2; 4. void getFlavourType () { 5. return flavourType; 6. } 7. }</pre>
