Think Java

CHAPTER 1: COMPUTER PROGRAMMING

DR. ERIC CHOU IEEE SENIOR MEMBER

Objectives

- The goal of this book is to teach you to **think like a computer scientist**. This way of thinking combines some of the best features of mathematics, engineering, and natural science. Like mathematicians, computer scientists use formal languages to denote ideas, specifically computations.
- Like engineers, they design things, assembling components into systems and evaluating trade-offs among alternatives. And like scientists, they observe the behavior of **complex systems**, form **hypotheses**, and **test predictions**.

Objectives

- An important skill for a computer scientist is **problem solving**. It involves the ability to formulate problems, think creatively about solutions, and express solutions clearly and accurately. As it turns out, the process of learning to program computers is an excellent opportunity to develop problem-solving skills.
- On one level you will be learning to write Java programs, a useful skill by itself. But on another level you will use programming as a means to an end. As we go along, that end will become clearer.

Software Installation and Homework Assignments

- BlueJ: http://BlueJ.org
- PeaZip
- Assignments: http://Moodle.eCode24.com

Topics

- Introduction to Computers
- Programming Language and Compiler
- Java Language and Programming Environment
- Introduction to Programming Flow

LECTURE 1

What is a computer?

A Computer System

All the different parts of a computer, including the devices you plug into it, are known collectively as 'a computer system'.

Monitor (Output)

A monitor, or visual display unit (VDU), displays the information generated by a computer. Touch-screen monitors also allow input from the user.

Webcam (Input)

A webcam enables live video communication over the Internet. Webcams are often integrated within computers.

Optical Disc Drive (Storage)

CD, DVD and Blu-ray disc drives are used to read information on the associated optical discs. To write (record) information onto an optical disc, a writer is required. The optical disc must also be writable (R) or rewritable (RW).

Speakers (Output)

Speakers are used to output sound.

Base Unit (Processing)

A base unit stores the motherboard, Central Processing Unit (CPU), Random-Access Memory (RAM) and other essential components.

Printer (Output)

A printer creates physical copies of on-screen data and information.

Scanner (Input)

A scanner is used to convert physical objects, usually printed images and text, into digital images.

Headset (Input and Output)

A headset is comprised of headphones (speakers) and a microphone. Headsets are commonly used to communicate over the Internet and for online gaming.

Keyboard (Input)

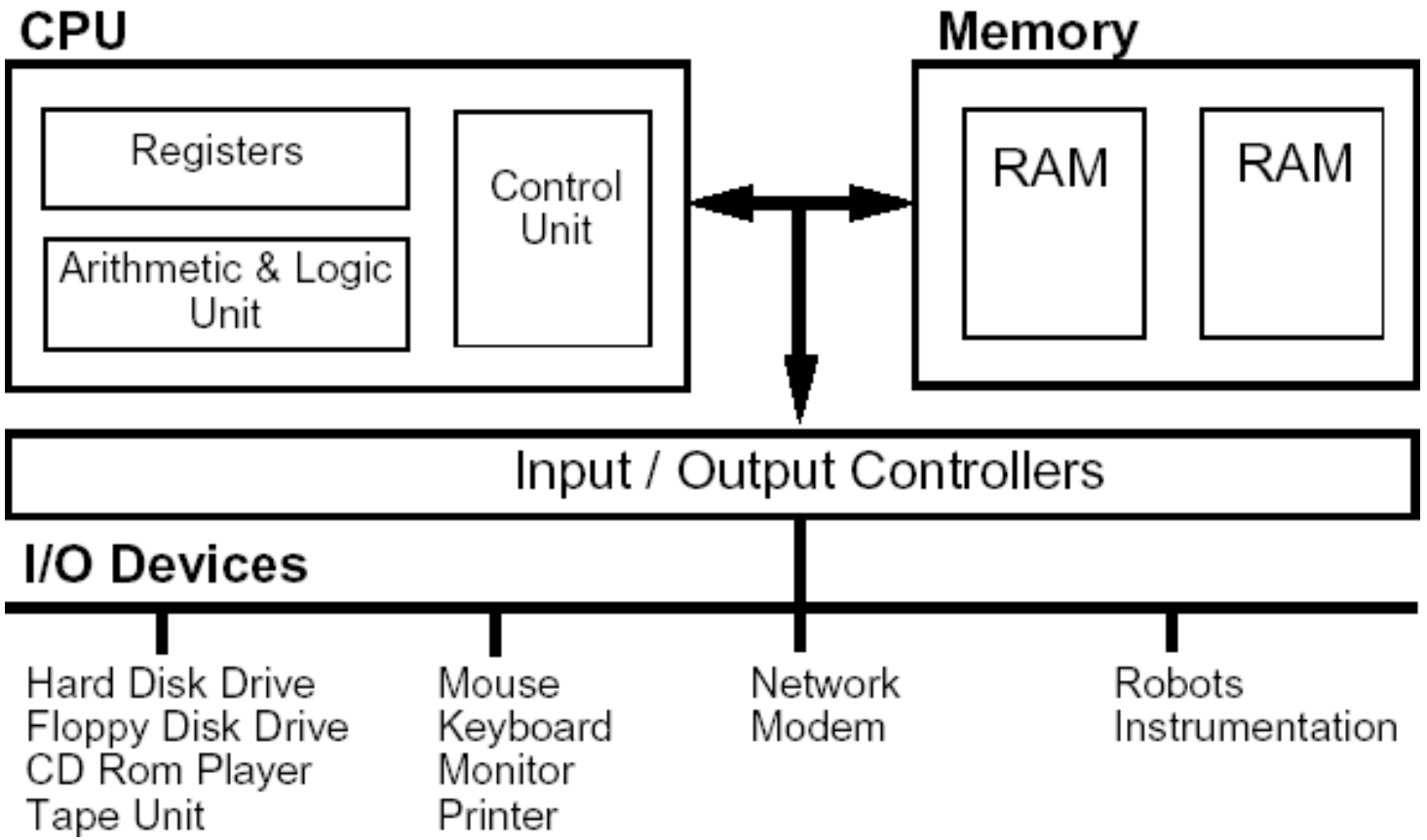
A keyboard is used to enter information into a computer system. Keyboards contain a number of keys, including alphanumeric characters, punctuation symbols and various function keys.

Mouse (Input)

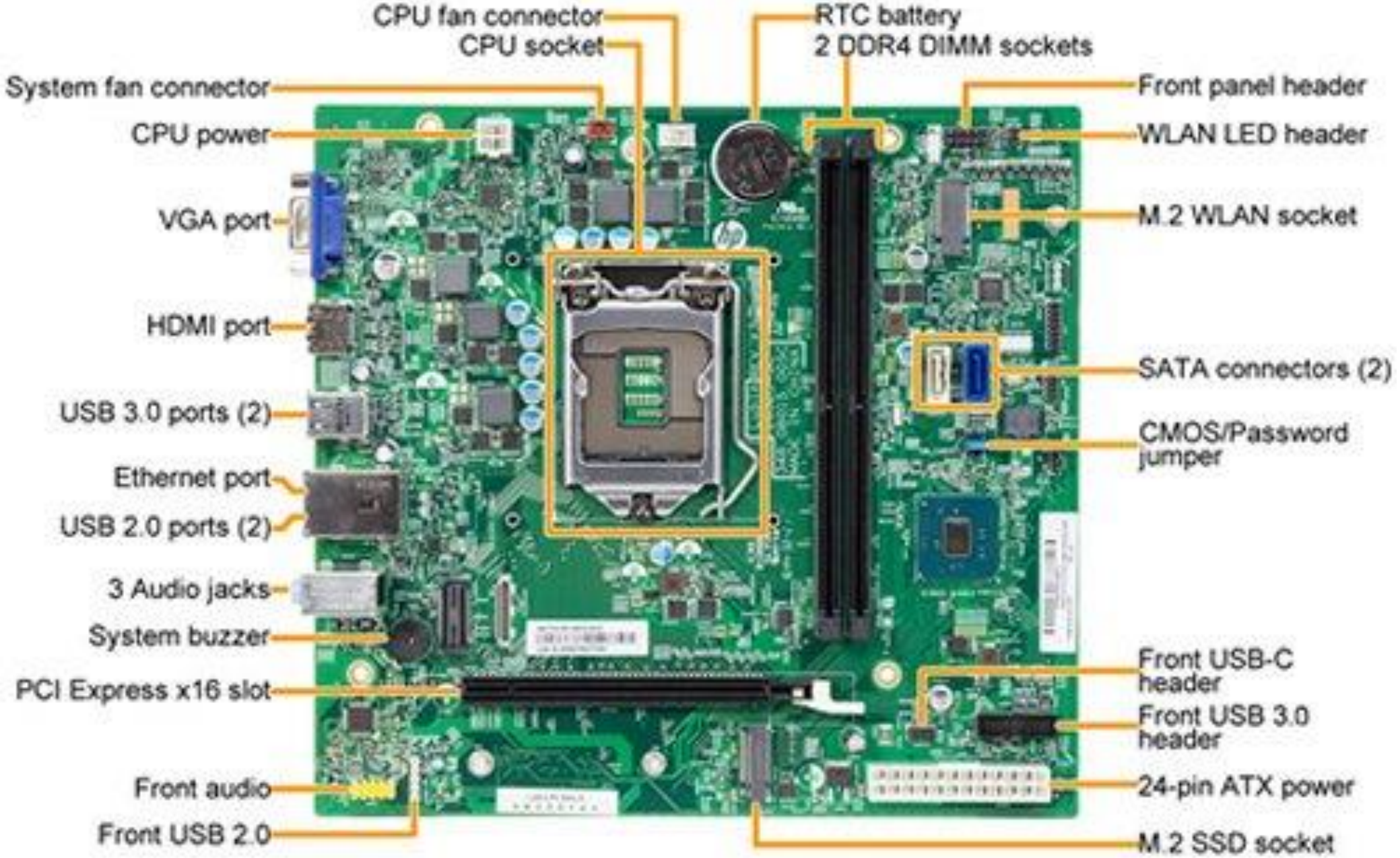
A mouse is used to move the cursor on a monitor, enabling control of the Graphical User Interface (GUI).

USB Flash Drive (Storage)

A USB flash drive is a portable storage device that connects to a computer's USB port.



What is a Computer?



LECTURE 2

What is programming?

What is programming?

- A **program** is a sequence of instructions that specifies how to perform a computation on computer hardware. The computation might be something mathematical, like solving a system of equations or finding the roots of a polynomial.
- It could also be a symbolic computation, like searching and replacing text in a document or (strangely enough) compiling a program.

What is programming?

The details look different in different languages, but a few basic instructions appear in just about every language:

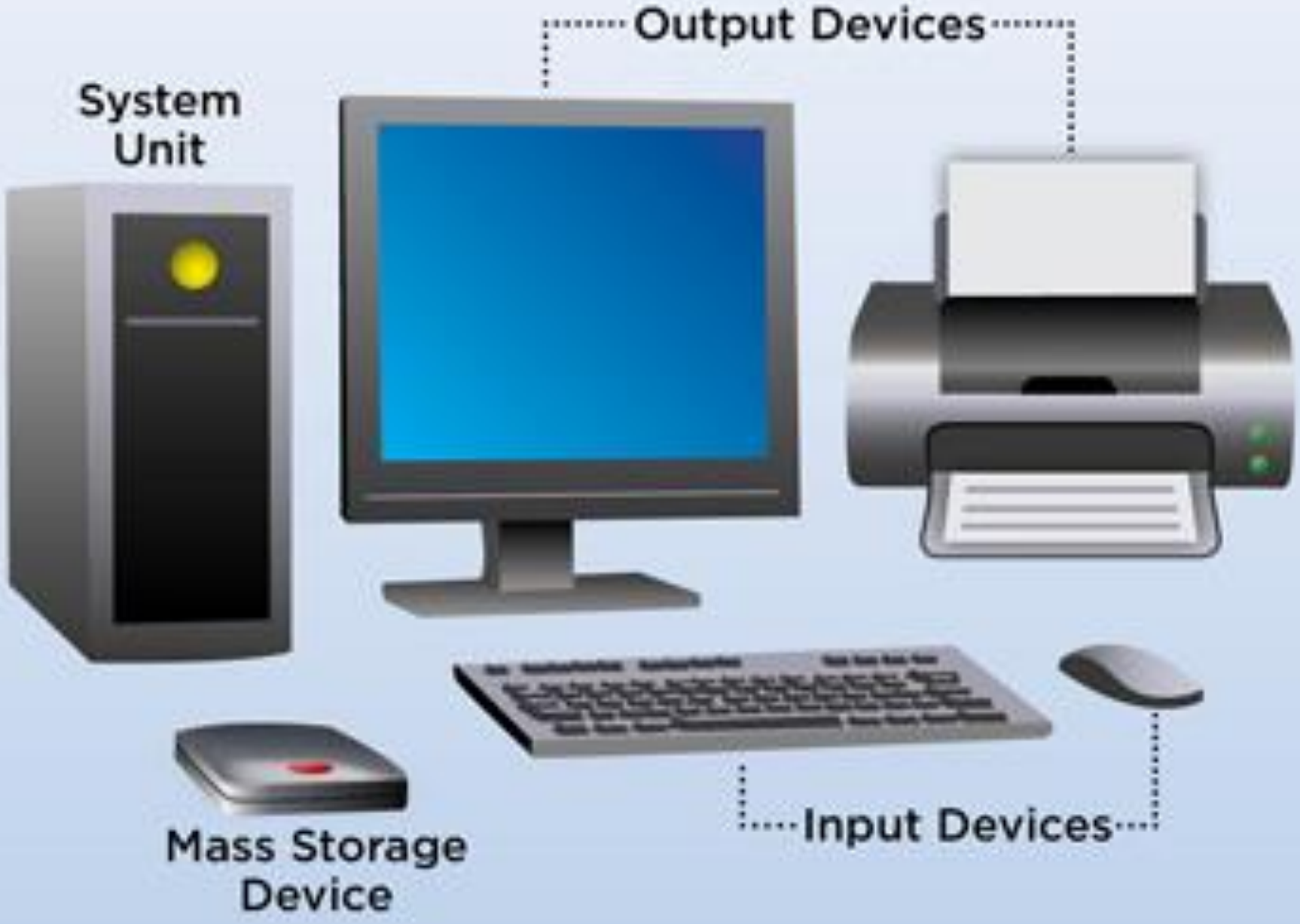
input: Get data from the keyboard, a file, a sensor, or some other device.

output: Display data on the screen, or send data to a file or other device.

math: Perform basic mathematical operations like addition and division.

decisions: Check for certain conditions and execute the appropriate code.

repetition: Perform some action repeatedly, usually with some variation.


Output Devices
System Unit
Mass Storage Device
Input Devices

LECTURE 3

The hello world program

Demo: Hello.java

```
public class Hello {
  public static void main(String[] args) {
     // generate some simple output
     System.out.println("Hello, World!");
  }
}
```

Statement

- Java programs are made up of *class* and *method* definitions, and methods are made up of *statements*. A **statement** is a line of code that performs a basic action. In the hello world program, this line is a **print statement** that displays a message on the screen:

```
System.out.println("Hello, World!");
```

Statement

- System.out.println displays results on the screen; the name println stands for “print line”. Confusingly, *print* can mean both “display on the screen” and “send to the printer”. In this book, we’ll try to say “display” when we mean output to the screen. Like most statements, the print statement ends with a semicolon (;).
- Java is “case-sensitive”, which means that uppercase and lowercase are not the same. In this example, System has to begin with an uppercase letter; system and **SYSTEM** won’t work.

Method

- A **method** is a named sequence of statements. This program defines one method named main:

```
public static void main(String[] args)
```

- The name and format of main is special: when the program runs, it starts at the first statement in main and ends when it finishes the last statement. Later, we will see programs that define more than one method.

Class

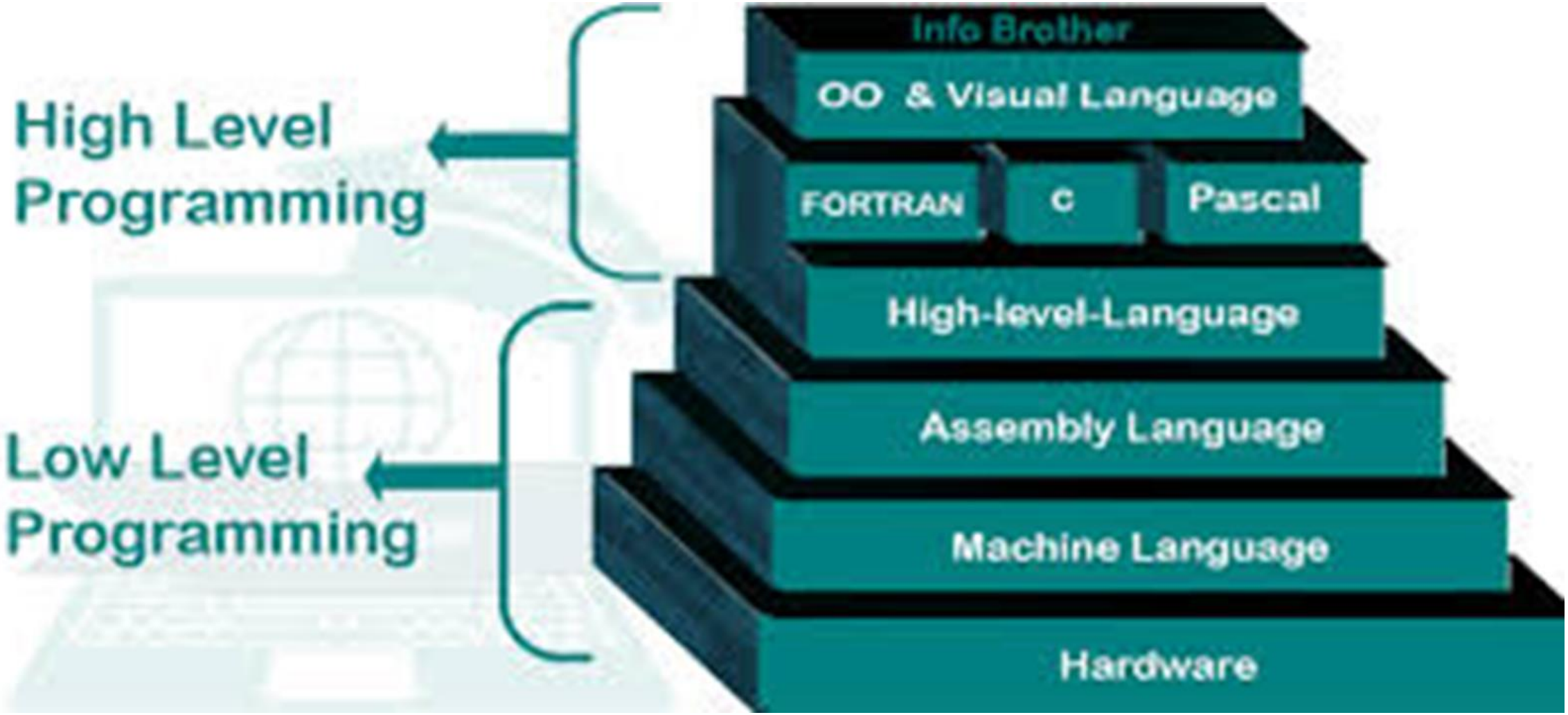
- A **class** is a collection of methods. This program defines a class named Hello. You can give a class any name you like, but it is conventional to start with a capital letter. The name of the class has to match the name of the file it is in, so this class has to be in a file named Hello.java.
- Java uses curly braces ({ and }) to group things together.
  In Hello.java, the outermost braces contain the class definition, and the inner braces contain the method definition.

Comment

- The line that begins with two slashes (//) is a **comment**, which is a bit of English text that explains the code.
- When Java sees //, it ignores everything from there until the end of the line. Comments have no effect on the execution of the program, but they make it easier for other programmers (and your future self) to understand what you meant to do.

LECTURE 4

Compiling Java programs


High Level Programming
Low Level Programming
Info Brother
OO & Visual Language
FORTRAN
C
Pascal
High-level-Language
Assembly Language
Machine Language
Hardware

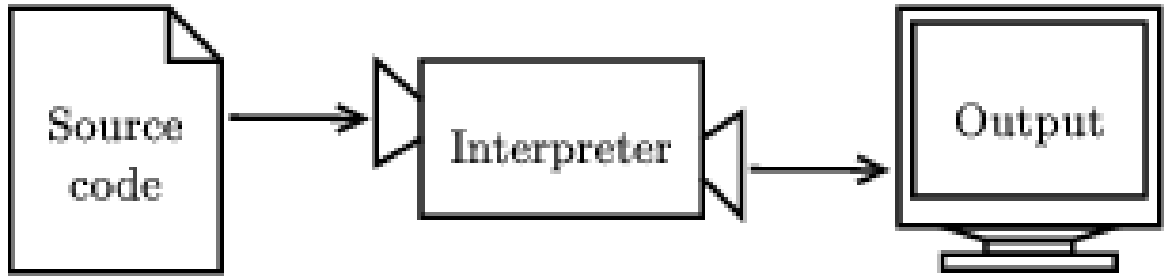
Programming Languages

- The programming language you will learn in this book is Java, which is a **high-level language**. Other high-level languages you may have heard of include Python, C and C++, PHP, Ruby, and JavaScript.
- Before they can run, programs in high-level languages have to be translated into a **low-level language**, also called “machine language”. This translation takes some time, which is a small disadvantage of high-level languages. But high-level languages have two major advantages:
- It is *much* easier to program in a high-level language. Programs take less time to write, they are shorter and easier to read, and they are more likely to be correct.
- High-level languages are **portable**, meaning they can run on different kinds of computers with few or no modifications. Low-level programs can only run on one kind of computer, and have to be rewritten to run on another.

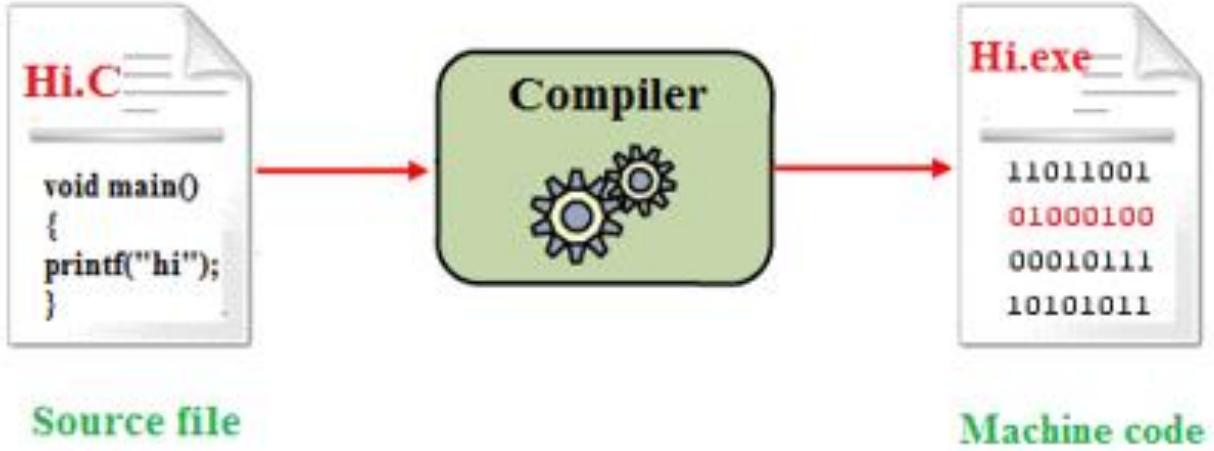
How many programming languages should I learn?

Interpreter

- Two kinds of programs translate high-level languages into low-level languages: interpreters and compilers.
- An **interpreter** reads a high-level program and executes it, meaning that it does what the program says. It processes the program a little at a time, alternately reading lines and performing computations.
- Figure 1.2 shows the structure of an interpreter.

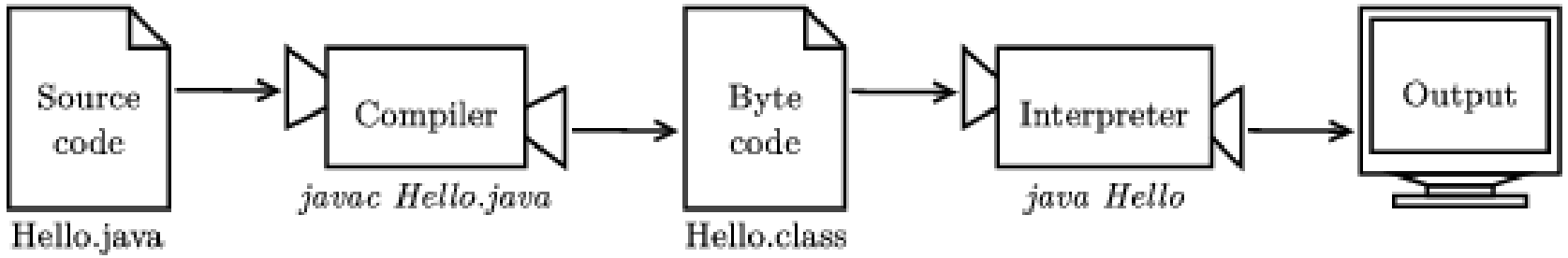


Compiler



- In contrast, a **compiler** reads the entire program and translates it completely before the program starts running. In this context, the high-level program is called the **source code**, and the translated program is called the **object code** or the **executable**.
- Once a program is compiled, you can execute it repeatedly without further translation. As a result, compiled programs often run faster than interpreted programs.

Hybrid



- Java is *both* compiled and interpreted. Instead of translating programs directly into machine language, the Java compiler generates **byte code**. Similar to object code, byte code is easy and fast to interpret. But it is also portable, so it is possible to compile a Java program on one machine, transfer the byte code to another machine, and run the byte code on the other machine. This ability is an advantage of Java over some other high-level languages.
- Figure 1.3 shows the steps of the development process. The Java compiler is a program named `javac`. It translates `.java` files into `.class` files that store the resulting byte code. The Java interpreter is a program named `java`, which is short for “Java Virtual Machine” (JVM)

Java Programs

- The programmer writes source code in the file `Hello.java` and uses `javac` to compile it. If there are no errors, the compiler saves the byte code in the file `Hello.class`.
- To run the program, the programmer uses `java` to interpret the byte code. The result of the program is then displayed on the screen.

Java Is...

- Simpler than C++
- An Object Oriented Language
- Platform-independent
- Used on the Internet
- General purpose
- Secure
- Capable of containing multiple threads
- Capable of supporting various multimedia
- Programmed to collect garbage automatically
- Reliable and Robust

Types of Java Programs

- **Applets** appear in a web page much in the same way as images do, but unlike images, applets are dynamic and interactive.
  - Used to create animations, games, ecommerce, etc.
- **Applications** are programs that can be run from the command line on a computer.
  - Sun's Star Office is a word processor, spreadsheet, etc. completely written in the Java language.
- **Servlets** are programs that respond to requests from clients.

Java Programming Tools

- **Integrated Development Environment (IDE):** software application that provides programmers with a programming environment equipped with the tools that they need to quickly and efficiently develop code.
  - Examples: JCreator, JPad, Eclipse, JBuilder, BlueJ
- **Java Development Kit (JDK)**: contains a compiler, a debugger, extensive class libraries, an execution environment, and sample codes.
  - **Java Virtual Machine (JVM)**: reads the bytecode found in the class file and translates the commands into machine language instructions for that computer's specific processor
    - The JVM is included with the JDK

LECTURE 5

Displaying two messages

Demo: Hello2.java

```
public class Hello2 {
   public static void main(String[] args) {
      // generate some simple output
      System.out.println("Hello, World!");
      // first line
      System.out.println("How are you?");
      // another line
   }

}
```

String and Output

- Phrases that appear in quotation marks are called **string**s, because they contain a sequence of characters strung together in memory. Characters can be letters, numbers, punctuation marks, symbols, spaces, tabs, etc.
- **System.out.println** appends a special character, called a **newline**, that moves to the beginning of the next line.

Demo: Goodbye.java

```
public class Goodbye {
    public static void main(String[] args) {
        System.out.print("Goodbye, ");
        System.out.println("cruel world");
    }
}
```

LECTURE 6

Formatting source code

print and println

```
public class Goodbye{
  public static void main(String[] args) {
    System.out.print("Goodbye, ");
    System.out.println("cruel world");
  }
}
```

Demo: Goodbye.java

```
public class Goodbye {
   public static void main(String[] args){
      System.out.print("Goodbye, ");
      System.out.println("cruel world");
   }
}
```

LECTURE 7

Escape sequences

Escape Sequence

- Each `\n` is an **escape sequence**, or two characters of source code that represent a single character. (The backslash allows you to "escape" the string to write special characters.)
- Notice there is no space between `\n` and `How`. If you add a space there, there will be a space at the beginning of the second line.

`\n`	newline
`\t`	tab
`\"`	double quote
`\`	backslash

In-Class Demonstration Program

- Table Design
- ASCII Art

TABLE.JAVA AND ART.JAVA

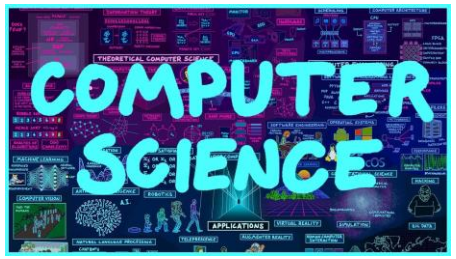
LECTURE 8

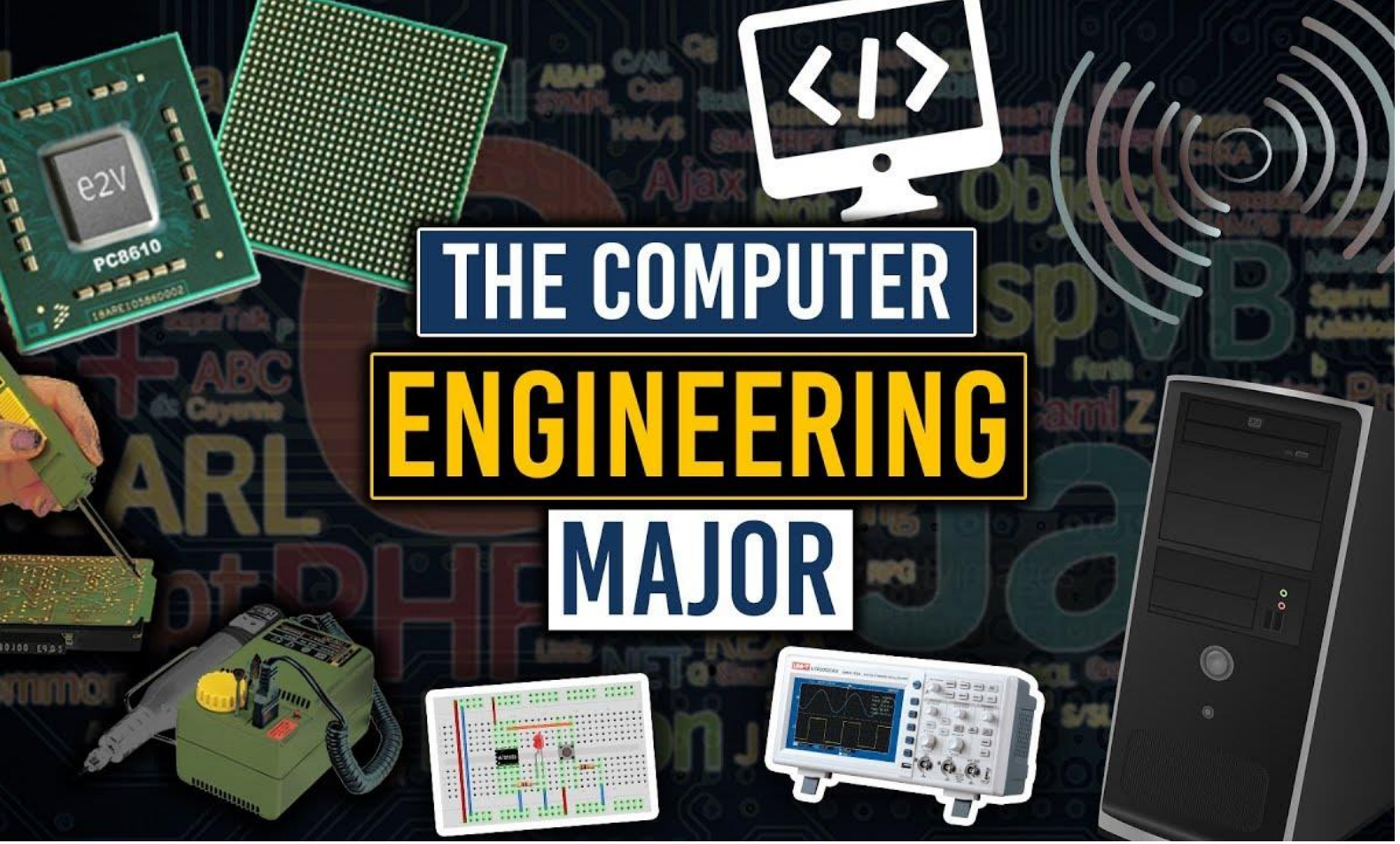
What is computer science?

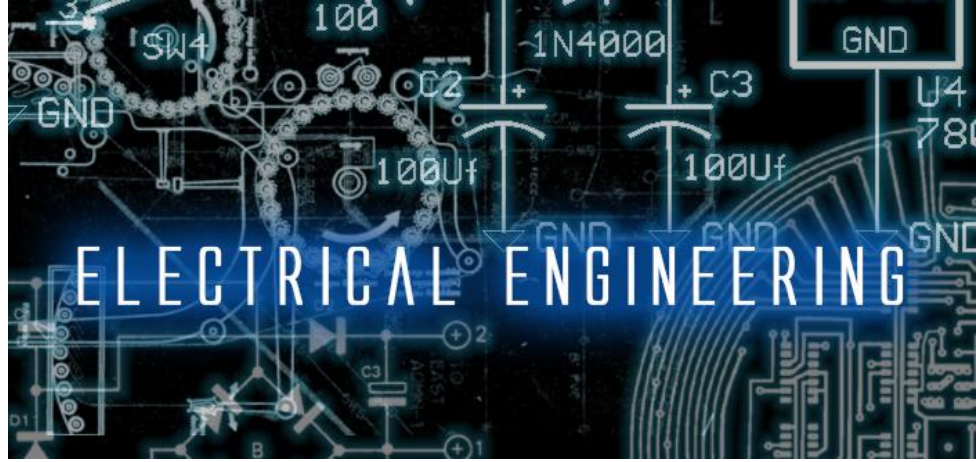
Physical

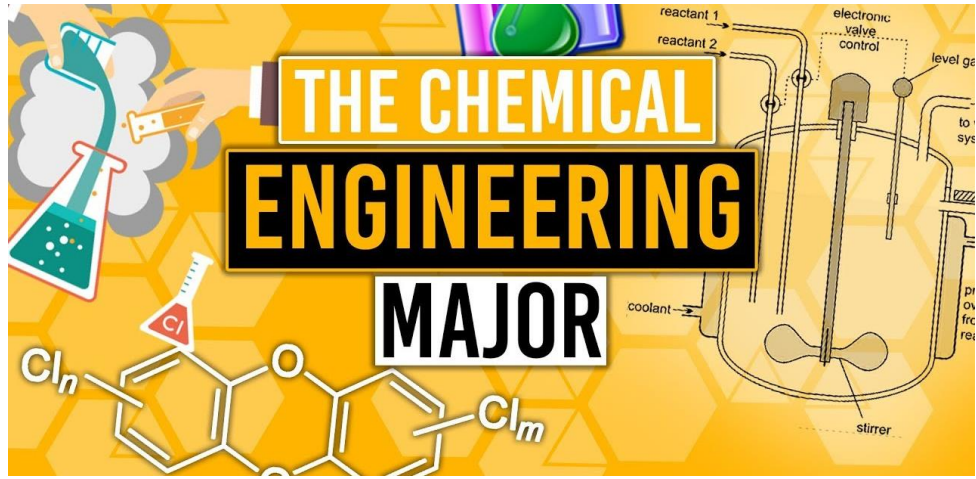
Science

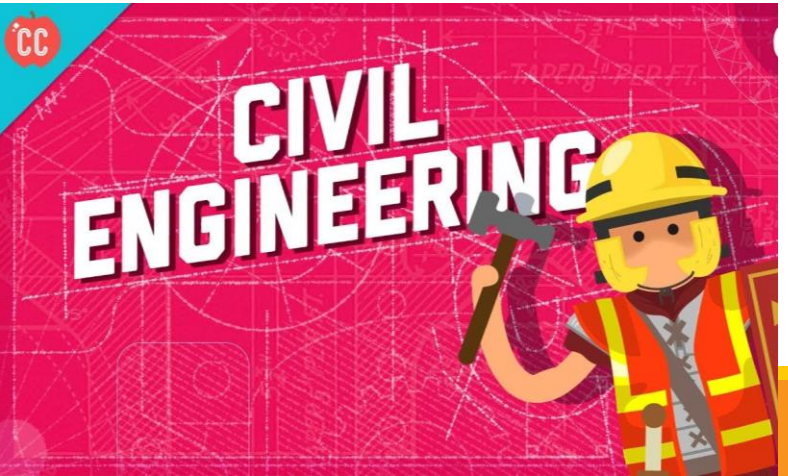
Earth
Science




THE COMPUTER
ENGINEERING
MAJOR


ELECTRICAL ENGINEERING


THE CHEMICAL
ENGINEERING
MAJOR


CIVIL
ENGINEERING


CHEMICAL
ENGINEERI


Mechanical
Engineering

Algorithm

- An **algorithm** is a sequence of steps that specifies how to solve a problem. Some algorithms are faster than others, and some use less space in computer memory.
- **Computer science** is the science of algorithms, including their discovery and analysis. As you learn to develop algorithms for problems you haven't solved before, you will learn to think like a computer scientist.

Design – Test – Debugging

- Designing algorithms and writing code is difficult and error-prone. For historical reasons, programming errors are called **bugs**, and the process of tracking them down and correcting them is called **debugging**.
- As you learn to debug your programs, you will develop new problem-solving skills. You will need to think creatively when unexpected errors happen.

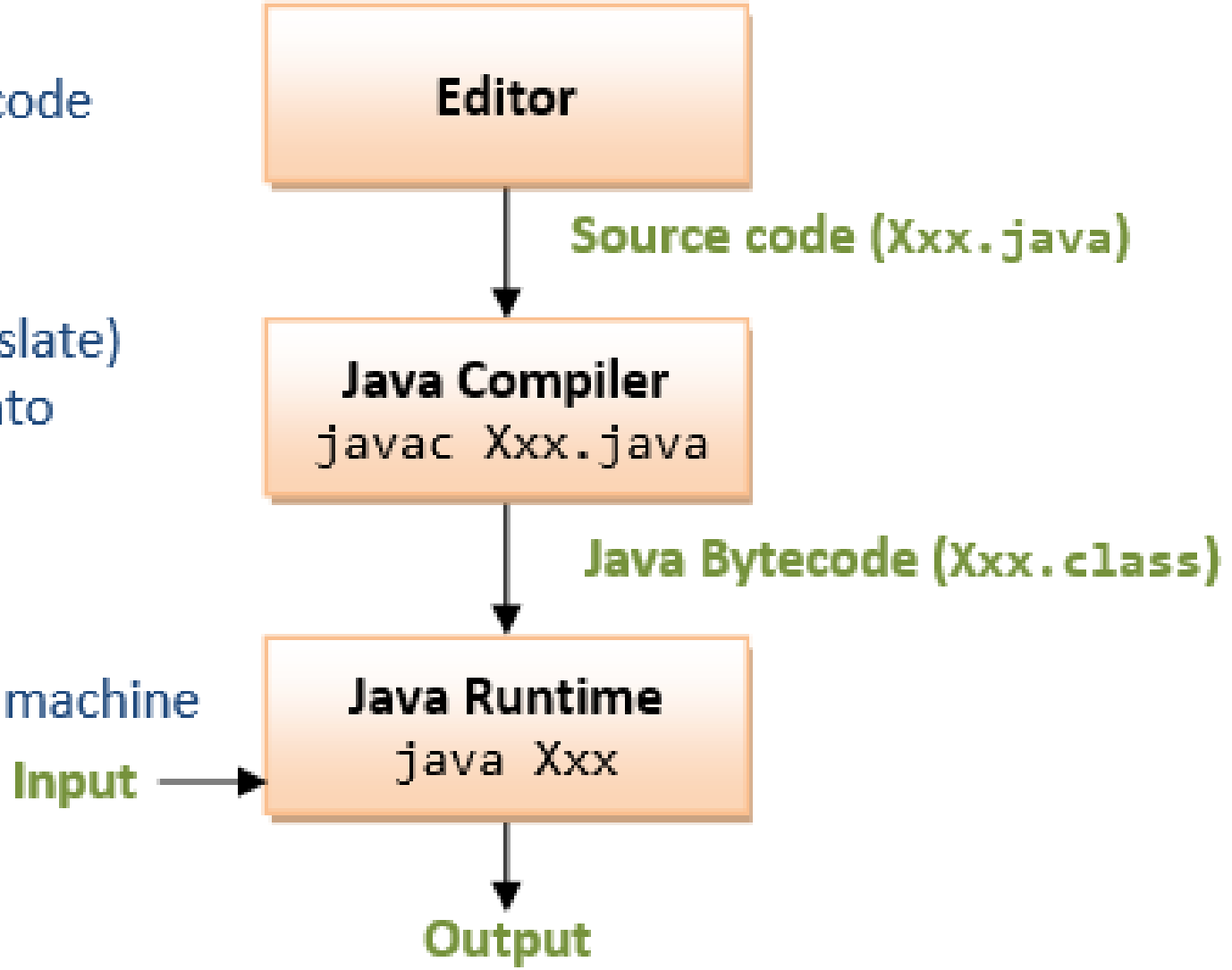
LECTURE 9

Debugging programs

Step 1
Write source code

Step 2
Compile (Translate) source code into machine code

Step 3
Execute (Run) machine code



Top 10 Java Debugging Tips

1. Use conditional breakpoint
2. Use exception breakpoints
3. Watchpoint
4. Step filtering
5. Evaluate (inspect and watch)
6. Drop to frame
7. Environment variables
8. Show logical structure
9. Modify values of variable
10. Stop in Main


4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
THE
SOFTWARE
DEVELOPMENT
CYCLE
6
MAINTENANCE
2
ANALYSIS
1
PLANNING

Synotive

Homework

Homework

- Textbook Exercise: 1.1, 1.2, 1-3
- Project 1