

Introduction to Computers, Programs, and Java

CS1: Java Programming
Colorado State University

Original slides by Daniel Liang
Modified slides by
Kris Brown, Ben Say, Wim Bohm



LEARNING: definitions

- **Learning** is a biological process. It occurs when networks of neurons in your brain send each other signals.
- **Thinking** is webs (of neurons) sending signals to other webs. A new idea is a newly formed sub network of neurons firing signals at each other
- Two aspects:
 - Understanding
 - Remembering



Understanding

- Comes sometimes in a “flash” (Oh I get it)
- Often times it takes **repeated exposure**, examples, associations, hard work
- If you don’t get it, go for a walk, come back, try more circuits in your brain

what does that mean?

- This requires FOCUS of ATTENTION (concentration) and REPEATED ACTION (do it, do it!)

Understanding cannot be achieved passively; it demands an **active and focused mind.**



Remembering

- Memory: also a biological process involving firing of neurons.
- Memories are **RECONSTRUCTED** (replayed) each time remembering happens.
- **Use it or lose it:** our brain's web connections are not permanent, they need to be reused / reactivated to stay
- Neural nets that get used a lot become stable



Learning

- Learning does NOT just happen to you
it is something **you do to yourself**.

The instructor can teach you theory and practice,
point the way, give you exercises, problems to solve,
do the hoki poki, BUT

- Learning is all **YOUR ACTIVITY**

The basic assumption is that you **want to** learn.

Without that, nothing will work



You won't learn unless you want to!

- Learning relies on your brain, which demands the same maintenance (food, exercise) as the rest of your body.
- Learning is difficult and it requires repeated effort.
- Don't cram the night before a test. Set yourself up for success!

But if done right, it is **GREAT!**

We wish you the best in your college days.



Motivation

A student asks a roommate, “Could you please go shopping for us and buy one carton of milk and, if they have avocados, get six.” A short time later, the roommate returns with six cartons of milk. “Why did you buy six cartons of milk?” asks the student. The reply: “They had avocados.”

Reader’s Digest, September 2013

This is exactly what your Java program will do, because computers do what you ask them to do, not what you want them to do!



Welcome: a first Java program

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Trace a Program Execution

Enter main method

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Trace a Program Execution

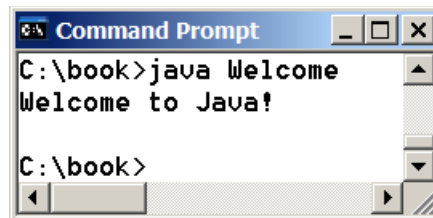
Execute statement

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Trace a Program Execution

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



```
Command Prompt  
C:\book>java Welcome  
Welcome to Java!  
C:\book>
```

print a message to the
console

Anatomy of a Java Program


- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks



Class Name

Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is `Welcome`.


```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Main Method

Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Statement


A statement represents an action or a sequence of actions. The statement `System.out.println("Welcome to Java!")` in the program is a statement to display the greeting "Welcome to Java!".

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Statement Terminator

Every statement in Java ends with a semicolon (;).


```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Reserved words

Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word **class**, it understands that the word after class is the name for the class.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Blocks

A pair of braces in a program forms a block that groups components of a program.

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Class block

Method block




Special Symbols

Character Name	Description
{ }	Opening and closing braces Denotes a block to enclose statements.
()	Opening and closing parentheses Used with methods.
[]	Opening and closing brackets Denotes an array.
//	Double slashes Precedes a comment line.
" "	Opening and closing quotation marks Enclosing a string (i.e., sequence of characters).
;	Semicolon Marks the end of a statement.




{ ... }

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



(...)

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```




•
;

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



// ...

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



'' ... ''

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles



Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.



Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
 - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`.



Proper Indentation and Spacing

- Indentation
 - Indent a consistent number of spaces or tabs.
- Spacing
 - Use blank line to separate segments of the code.



Block Styles

Use a consistent style for braces.

*Next-line
style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line
style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```





Implicit Import and Explicit Import

```
import java.util.* ; //Implicit import
```

```
import java.util.JOptionPane; //Explicit import
```

No performance difference



Programming Errors

- Syntax Errors
 - Detected by the compiler
- Runtime Errors
 - Causes the program to abort
- Logic Errors
 - Produces incorrect result



Syntax Errors

```
public class ShowSyntaxErrors {  
    // This code has syntax error by purpose  
    public static main(String[] args) {  
        System.out.println("Welcome to Java);  
    }  
}
```





Runtime Errors

```
// Program contains runtime errors
public class ShowRuntimeErrors {
    public static void main(String[] args) {
        System.out.println(1 / 0);
    }
}
```



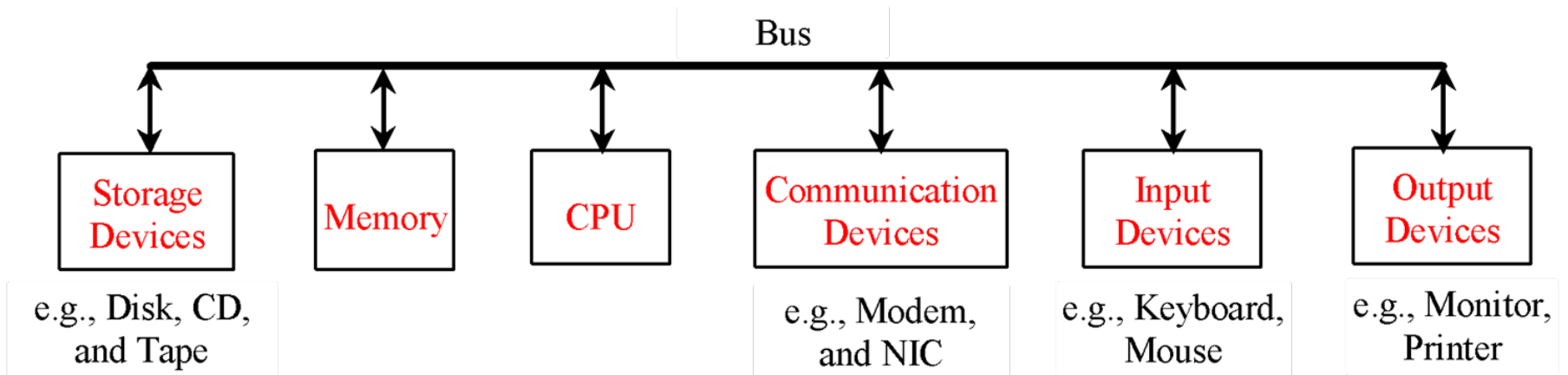
Logic Errors

```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.print("Celsius 35 is ");  
        System.out.print("Fahrenheit ");  
        System.out.println((9 / 5) * 35 + 32);  
    }  
}
```



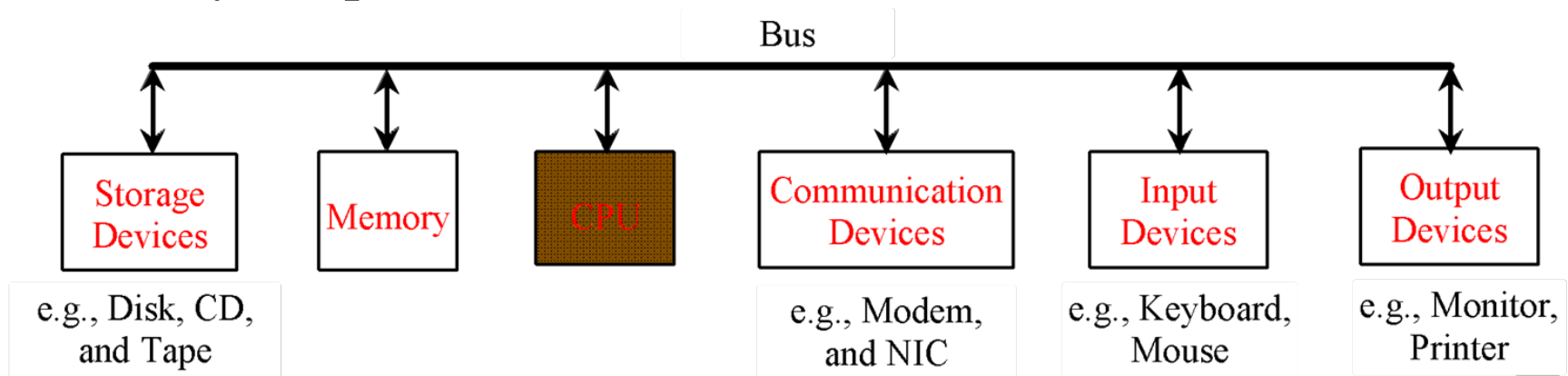
What is a Computer?

A computer consists of a CPU, memory, hard disk, other storage devices, monitor, printer, and communication devices.



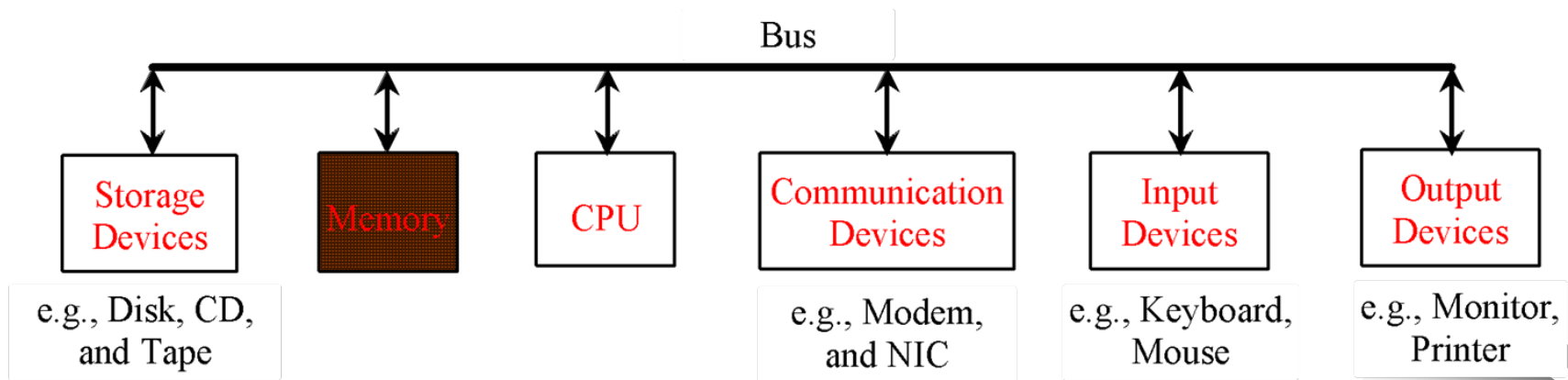
CPU

The **Central Processing Unit (CPU)** is the brain of a computer. It retrieves instructions from memory and executes them. The CPU speed is measured in gigahertz (GHz), with 1 gigahertz equaling 1 billion cycles per second.

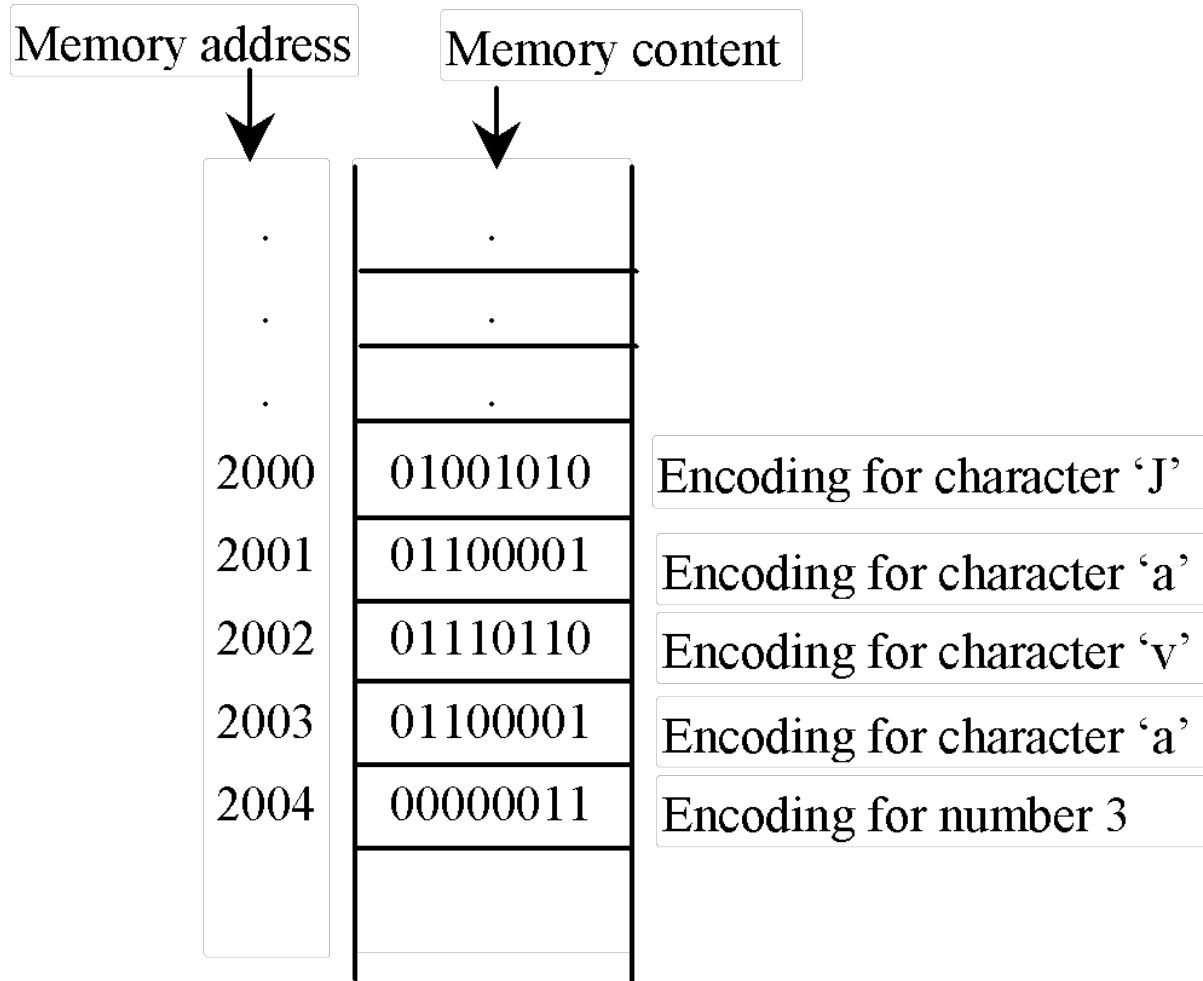


Memory

Memory is to store data and program instructions for CPU to execute. A memory unit is an ordered sequence of **bytes**, each byte holds eight bits. If you buy a PC today, it might have 8 gigabytes (GB) of memory.

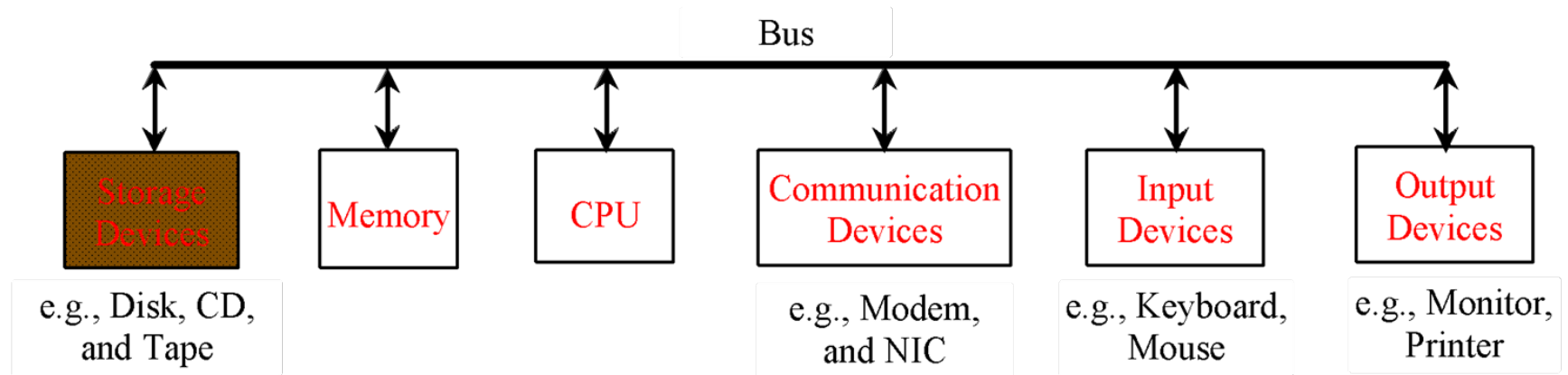


How Data is Stored?



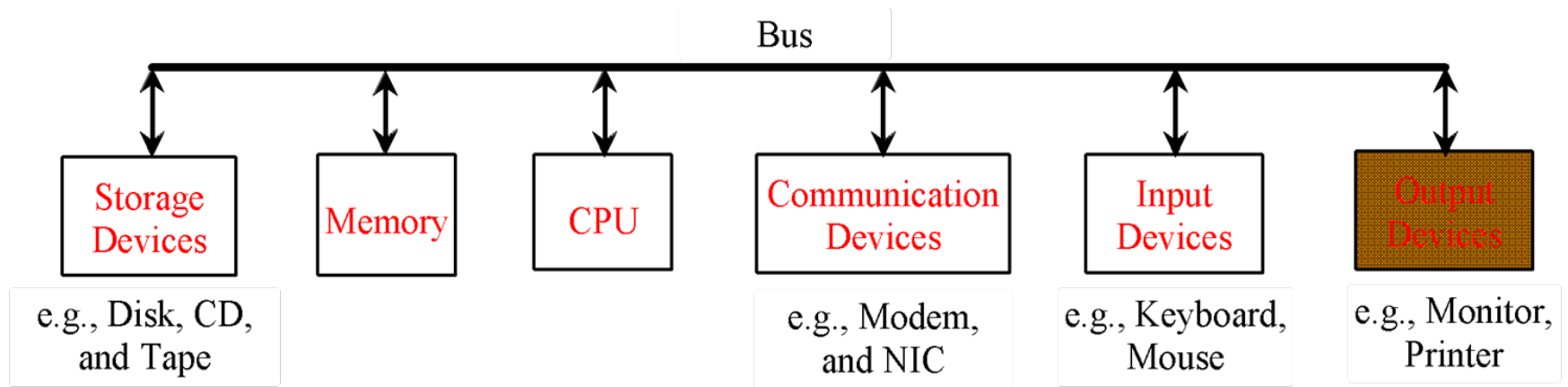
Storage Devices

Memory is **volatile**, because information is lost when the power is off. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them.



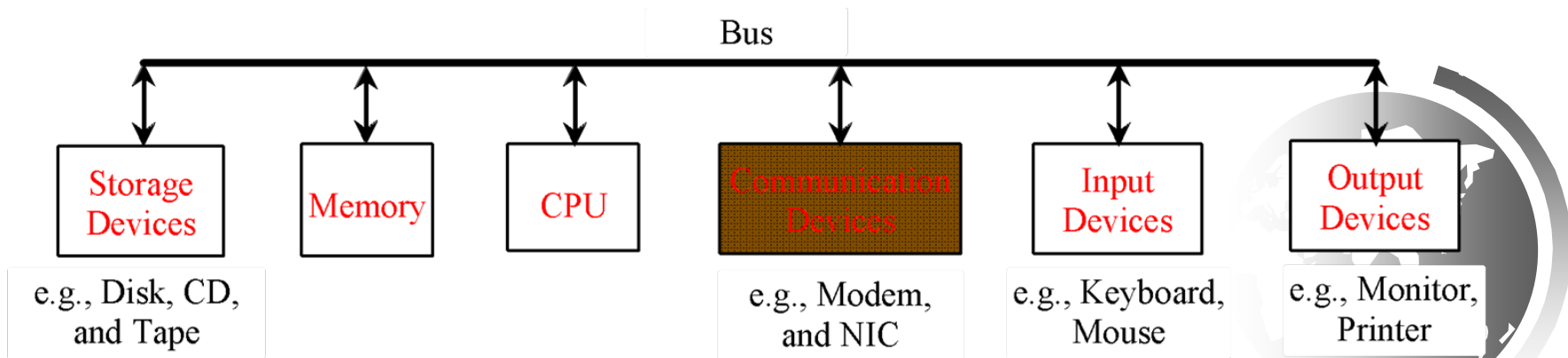
Output Devices: Monitor

The monitor displays information (text and graphics). The resolution and dot pitch determine the quality of the display.



Communication Devices

A *regular modem* uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second). A *DSL* (digital subscriber line) also uses a phone line and can transfer data in a speed 20 times faster than a regular modem. A *cable modem* uses the TV cable line maintained by the cable company. A cable modem is as fast as a DSL. Network interface card (*NIC*) is a device to connect a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations.



Programs

Computer *programs*, known as *software*, are structured sets of instructions to the computer.

You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

Programs are written using programming languages.



Programming Languages

Machine Language Assembly Language High-Level Language

Machine language is a set of primitive instructions built into every computer.

Instructions are in the form of binary code. This is tedious, error prone, hard to read and modify.

For example, to add two numbers, you might write an instruction in binary like this:

```
1101101010011010
```



Programming Languages

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

Assembly languages were developed to make machine programming easy. But one assembly-instruction is still one machine-instruction, so this is still tedious and error prone.

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
```

And the assembler turns this into 1101101010011010

Programming Languages

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

The **high-level languages** are structured and English-like and easier to learn and program. For example, the following is a high-level language statement (C, C++, Java, Python) that computes the area of a circle with radius 5:

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



Popular High-Level Languages

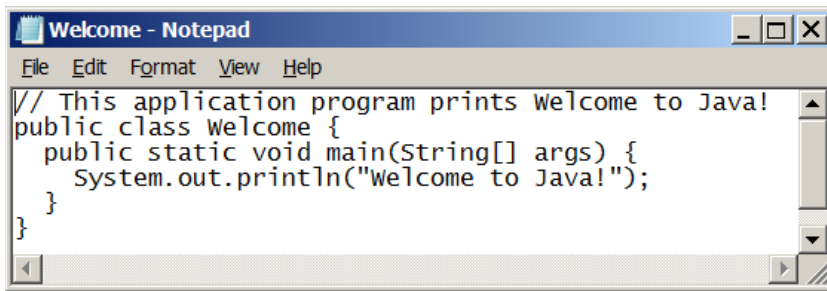
Language	Description
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmm on Business Oriented Language. Used for business applications.
FORTRAN	FORm ula TRANslati on. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

Interpreting/Compiling Source Code

A program written in a high-level language is called a *source program* or *source code*. Because a computer cannot understand a source program, a source program must be translated into machine code for execution. The translation can be done using another programming tool called an *interpreter* or a *compiler*.



Creating, Compiling, and Running Programs



```
Welcome - Notepad
File Edit Format View Help
// This application program prints welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Source code (developed by the programmer)

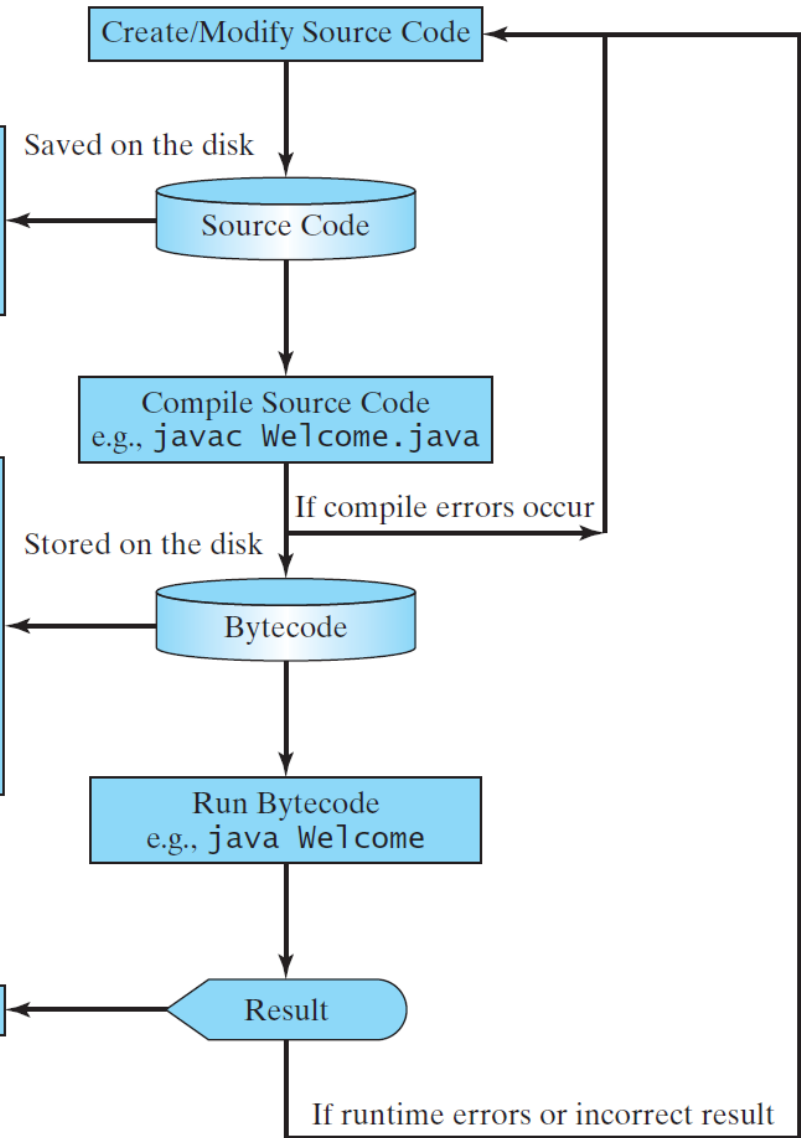
```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

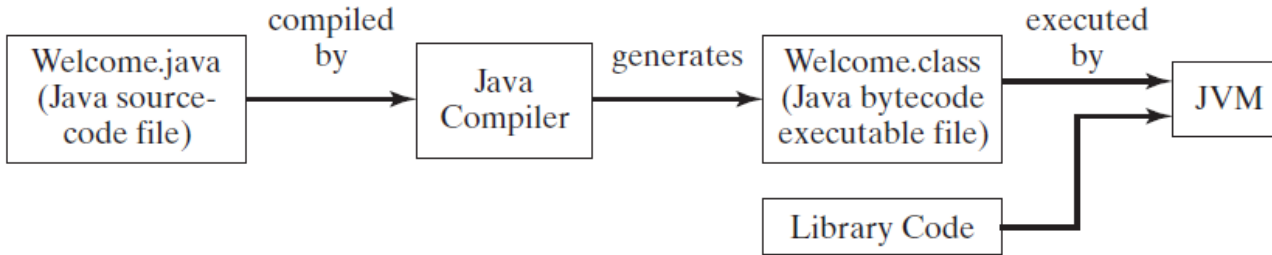
```
...
Method Welcome()
  0 aload_0
  ...
Method void main(java.lang.String[])
  0 getstatic #2 ...
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 ...
  8 return
```

“Welcome to Java” is displayed on the console

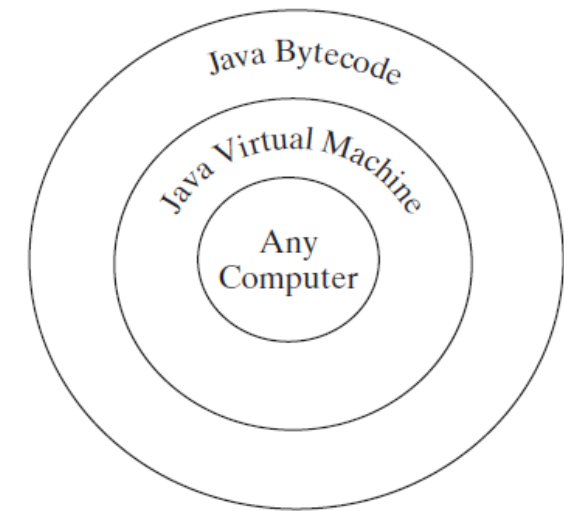
```
Welcome to Java!
```



Compiling Java Source Code



(a)



(b)

Java can be compiled into a special type of object code, known as *bytecode*. This bytecode can run (interpreted) on any computer using JVM (Java Virtual Machine) software.

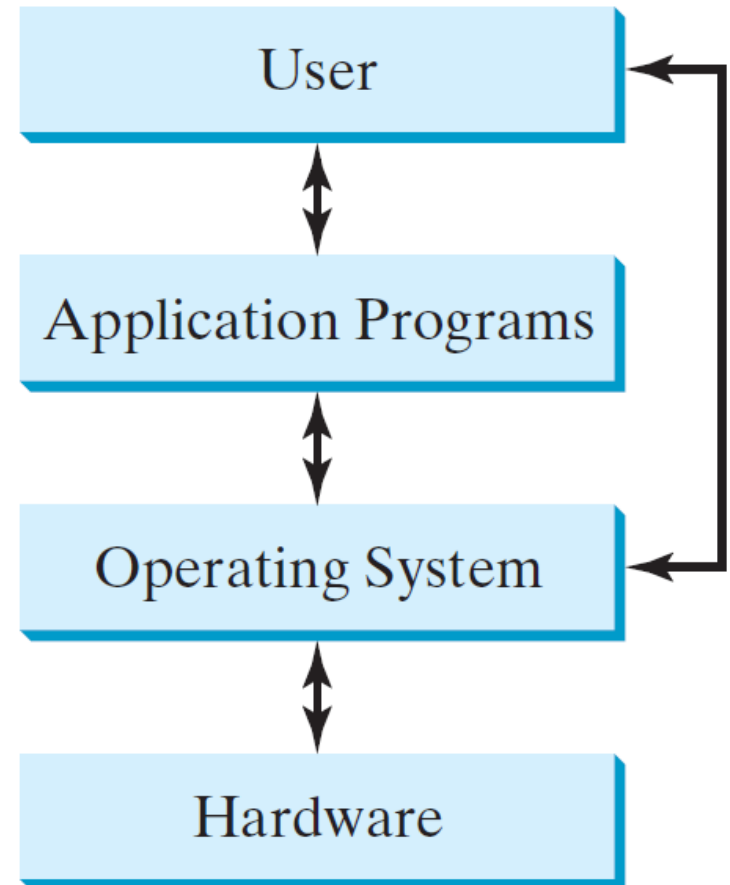


Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities and file system.

Popular operating systems are Microsoft Windows, Mac OS, and Linux.

Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.



Why Java?

Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices.

- Java is a general purpose programming language.

The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future.

- Java is an Internet programming language.





Characteristics of Java

- Java is relatively simple
- Java bytecode is Interpreted (JVM) but can also be compiled to native machine code
- Java is Architecture and OS neutral
- Java's performance keeps improving

www.cs.armstrong.edu/liang/JavaCharacteristics.pdf

