

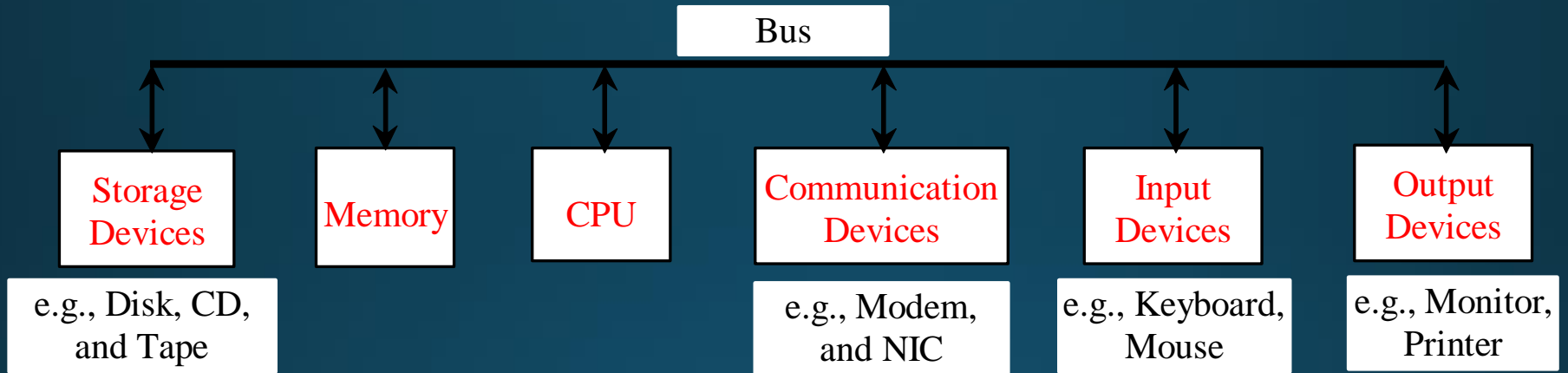
Chapter 1 Introduction to Computers, Programs, and Java

Objectives

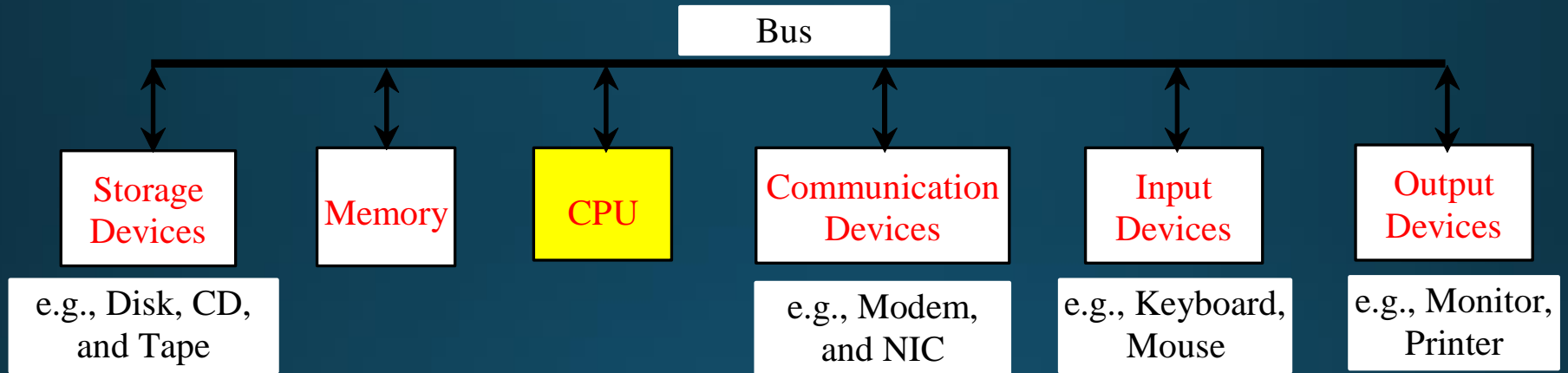
- To understand computer basics, programs, and operating systems (§§1.2–1.4).
- To describe the relationship between Java and the World Wide Web (§1.5).
- To understand the meaning of Java language specification, API, JDK, and IDE (§1.6).
- To write a simple Java program (§1.7).
- To display output on the console (§1.7).
- To explain the basic syntax of a Java program (§1.7).
- To create, compile, and run Java programs (§1.8).
- To use sound Java programming style and document programs properly (§1.9).
- To explain the differences between syntax errors, runtime errors, and logic errors (§1.10).
- To develop Java programs using NetBeans (§1.11).
- To develop Java programs using Eclipse (§1.12).

What is a computer?

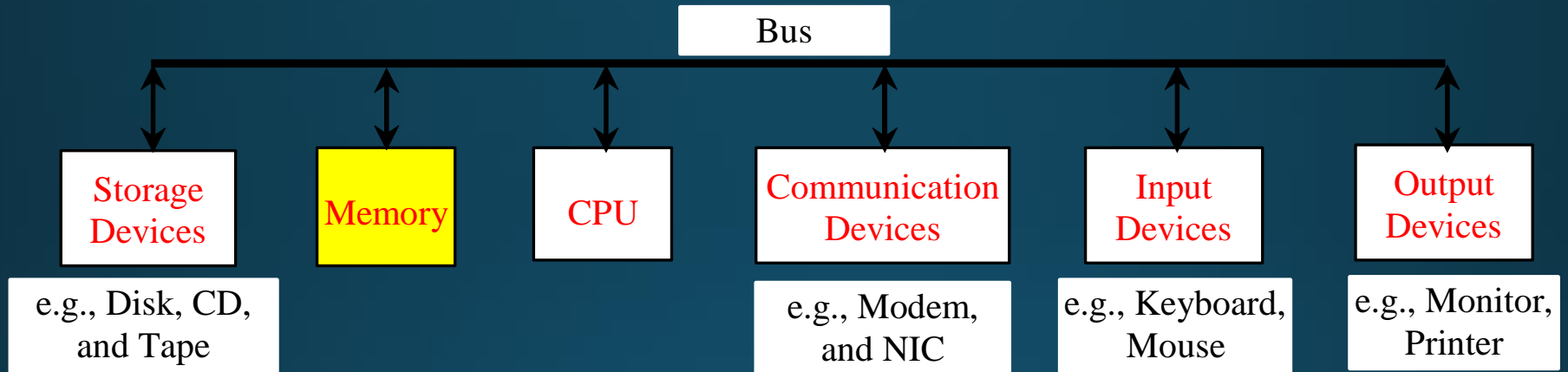
What is a Computer?



CPU



Memory

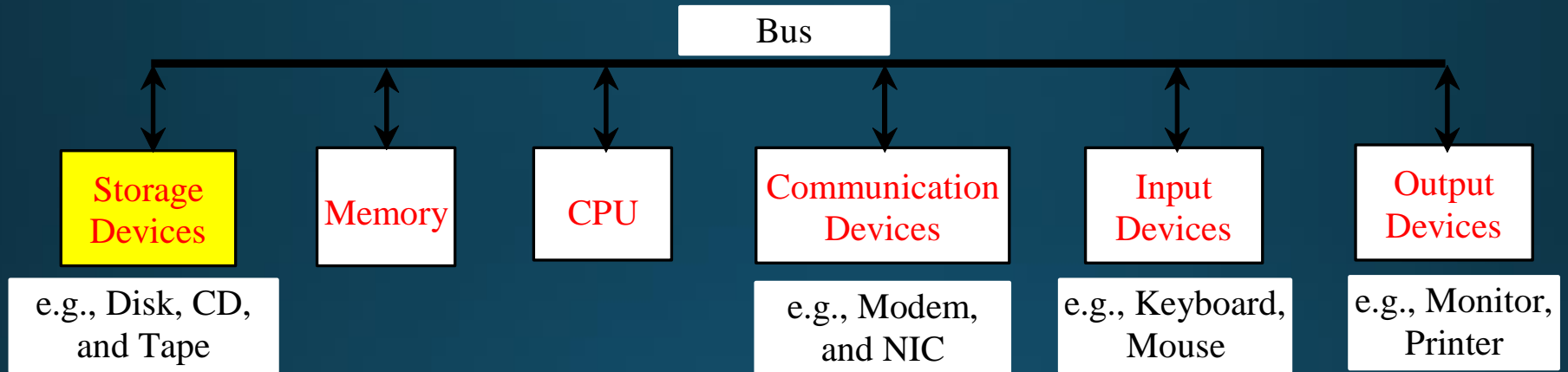


How Data is Stored?

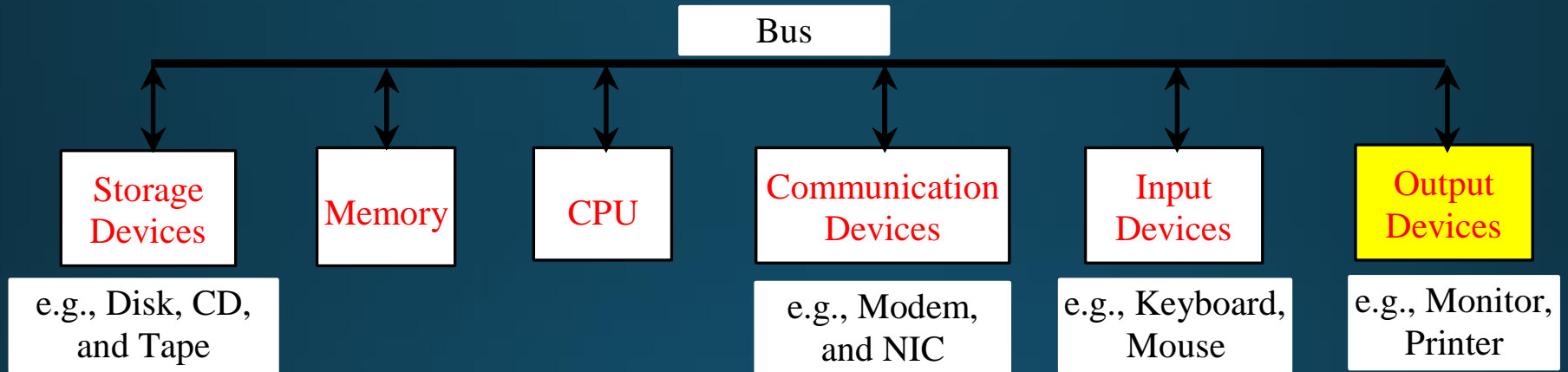
Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01001010	Encoding for character 'J'
2001	01100001	Encoding for character 'a'
2002	01110110	Encoding for character 'v'
2003	01100001	Encoding for character 'a'
2004	00000011	Encoding for number 3

- Digital Devices have two states, **0** and **1**
- Data are encoded as **bits**
- 8 bits = **1 byte**

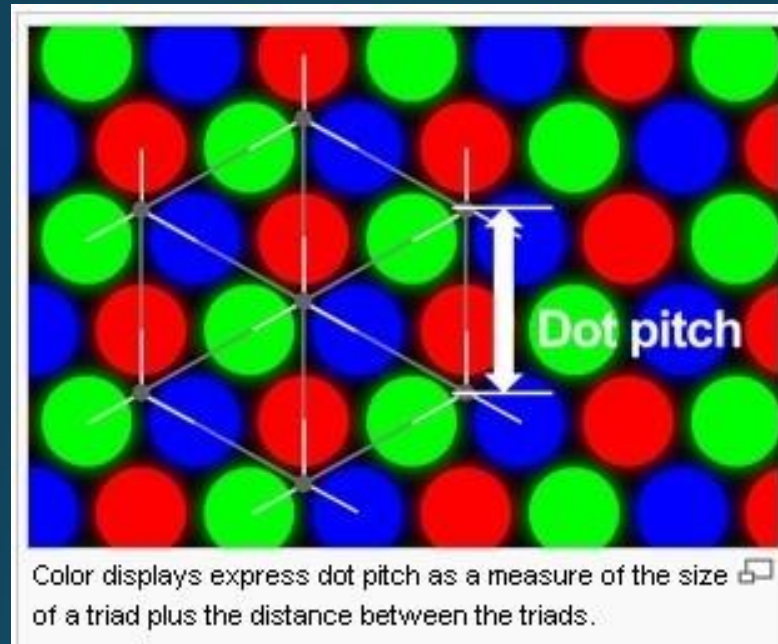
Storage Devices



Output Devices: Monitor

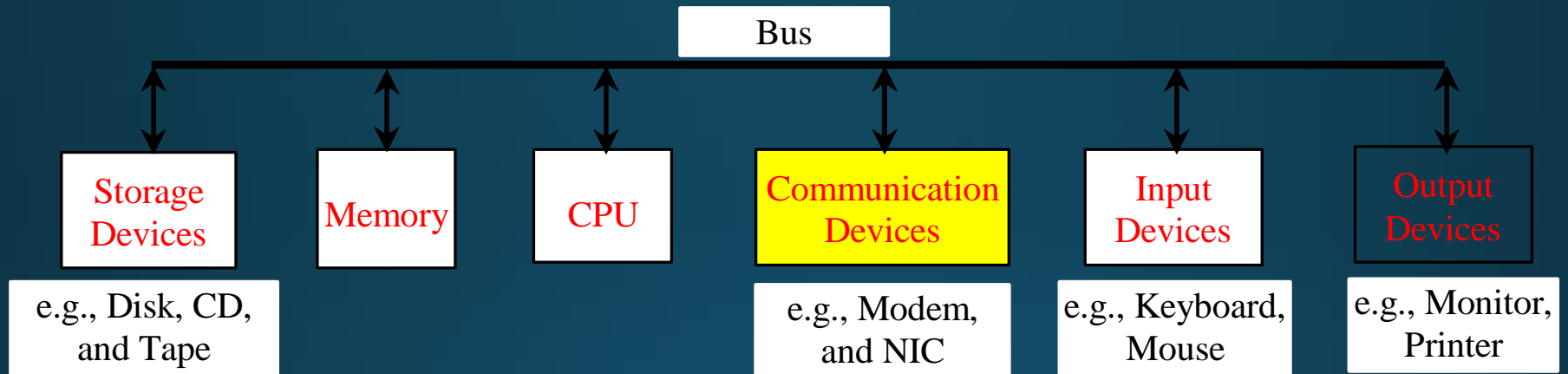


Monitor Resolution and Dot Pitch



From Wikipedia http://en.wikipedia.org/wiki/Dot_pitch

Communication Devices



Programs

- Computer *programs*, known as *software*, are instructions to the computer.
- Programs are written using *programming languages*.

Programming Languages

Machine Language Assembly Language High-Level Language

- A set of primitive instructions built into every computer.
- Written in *binary (machine) code*.

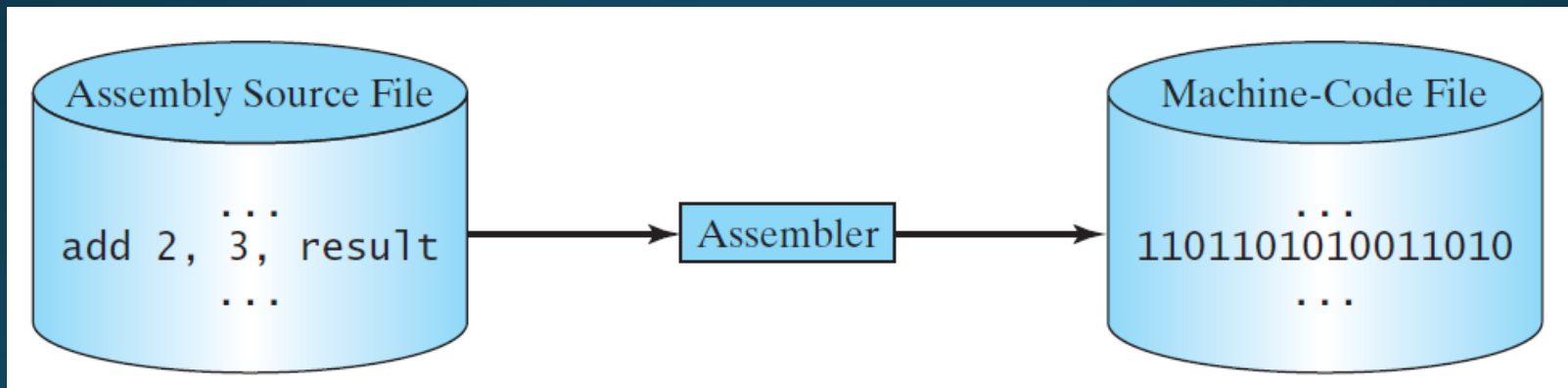
1101101010011010

Programming Languages

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

- Developed to make programs easier for humans to read/write.
- An **assembler** must be used to convert programs into machine code.

ADDF3 R1, R2, R3



Programming Languages

Machine Language Assembly Language High-Level Language

- Uses English-like statements.
- Easy to learn how to read/write programs.

```
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	COMmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. 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*.
- A source program must be *translated* into *machine code* for execution.
- Translation is done using other programming tools called *interpreters* or *compilers*.

Interpreting Source Code

- An *interpreter* reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away.

Interpreting Source Code

- An *interpreter* reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away.
- **JavaScript** and **Python** are *interpreted* languages.

Compiling Source Code

- A ***compiler*** translates the entire source code file into a machine-code file, and then machine-code file is executed.

Compiling Source Code

- A ***compiler*** translates the entire source code file into a machine-code file, and then machine-code file is executed.
- C, C++ and Visual Basic are ***predominantly compiled*** languages.

Partially Compiling Source Code

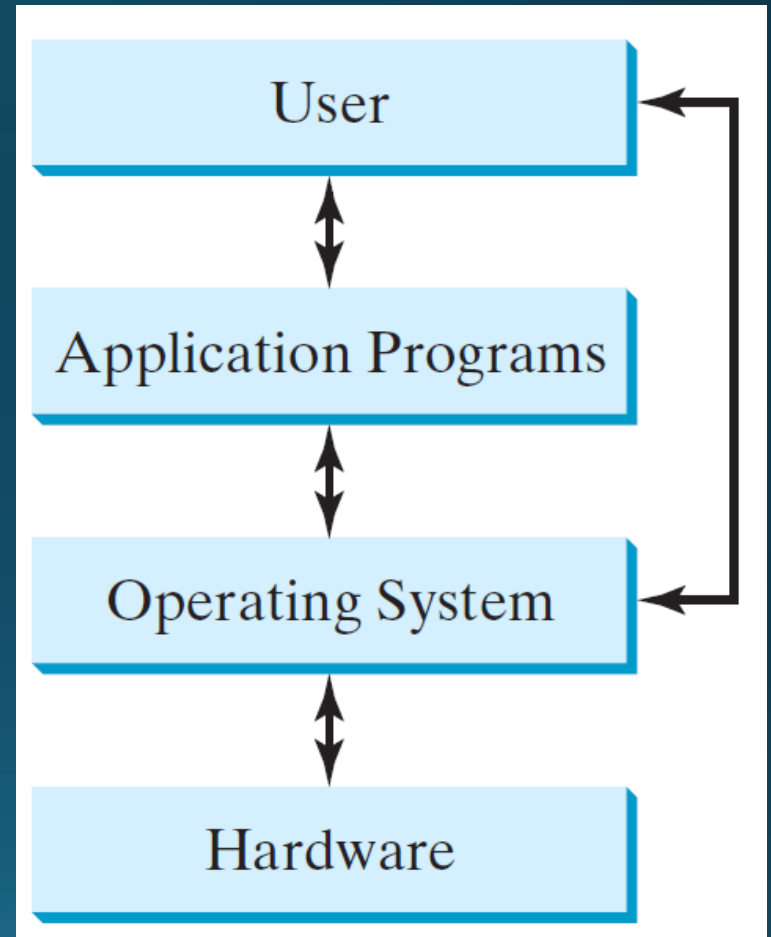
- A ***compiler*** translates an entire source code file into a ***bytecode*** (virtual machine code) file.
- An ***interpreter*** reads one statement from the bytecode file, translates it to the machine code, and then executes it right away.

Partially Compiling Source Code

- A *compiler* translates an entire source code file into a *bytecode* (virtual machine code) file.
- An *interpreter* reads one statement from the bytecode file, translates it to the machine code, and then executes it right away.
- Java is a *partially compiled* language.

Operating Systems

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



Why Java?

- ☞ Java is a *general purpose* programming language.
- ☞ Java is the Internet programming language.

Java's History

- James Gosling and Sun Microsystems
- Oak
- Java, May 20, 1995, Sun World
- HotJava
 - The first Java-enabled Web browser
- Early History Website:

<http://www.java.com/en/javahistory/index.jsp>

What is the JDK?

- The Java Development Kit is a collection of programming tools for developing and testing Java applications.
 - java - the Java application loader
 - javac - converts source code into bytecode
 - jar - packages class libraries into a portable JAR files
- Also includes the Java Runtime Environment.

JDK Versions

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) a. k. a. JDK 5 or Java 5
- JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- JDK 1.8 (2014) a. k. a. JDK 8 or Java 8

JDK Editions

- Java Standard Edition (JSE)
- Java Enterprise Edition (JEE)
- Java Micro Edition (JME).

This book uses JSE to introduce Java programming.

Popular Java IDEs

- NetBeans
- Eclipse

What is Netbeans?

- Netbeas is an Integrated Development Environment (IDE)
- IDE's make a software developer's life easier:
 - Unites all of the JDK's tools under one, unified Graphical User Interface (GUI)
 - Helps programmers to write, debug and test large applications quickly

A Simple Java Program

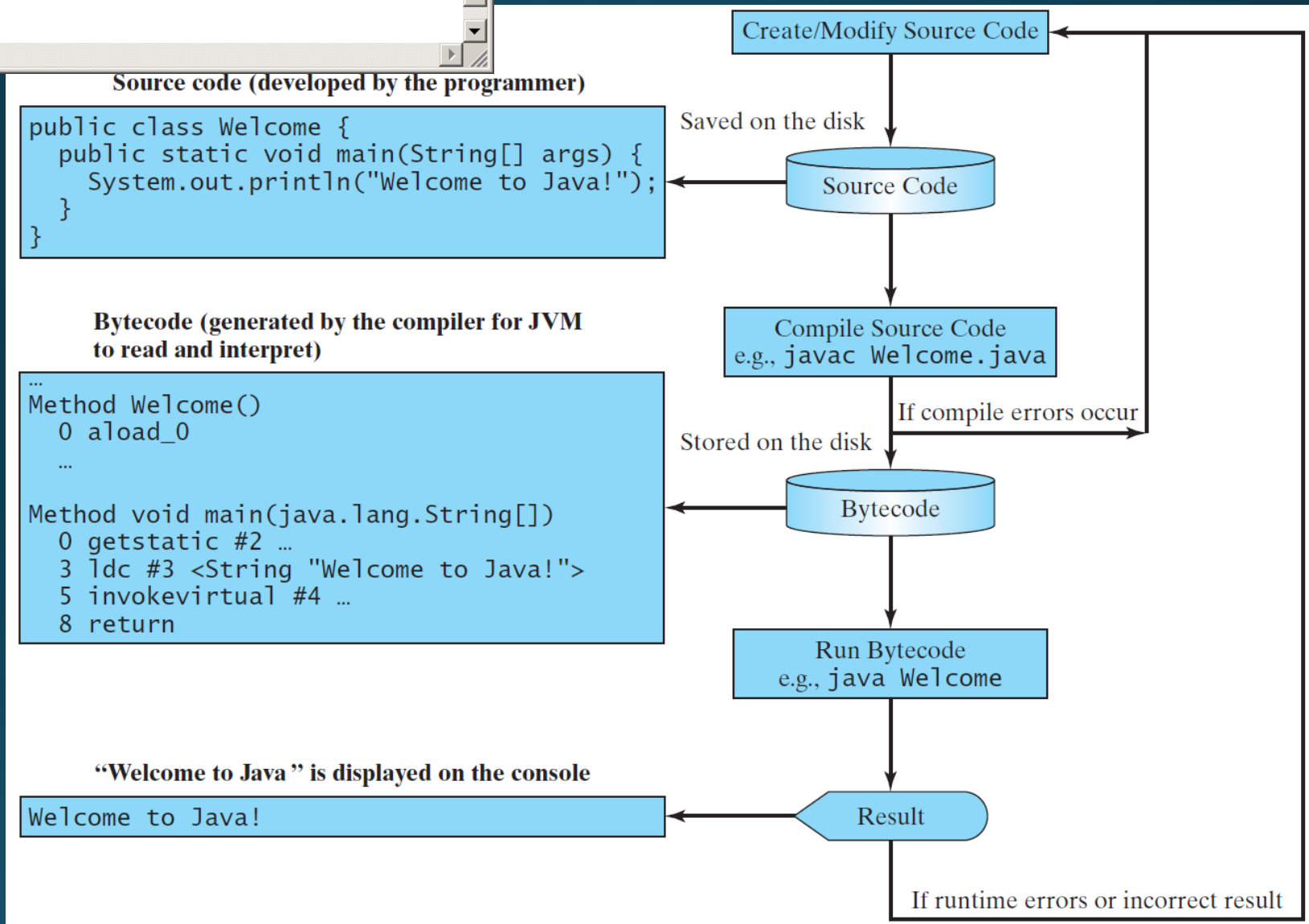
Listing 1.1

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


Welcome - Notepad

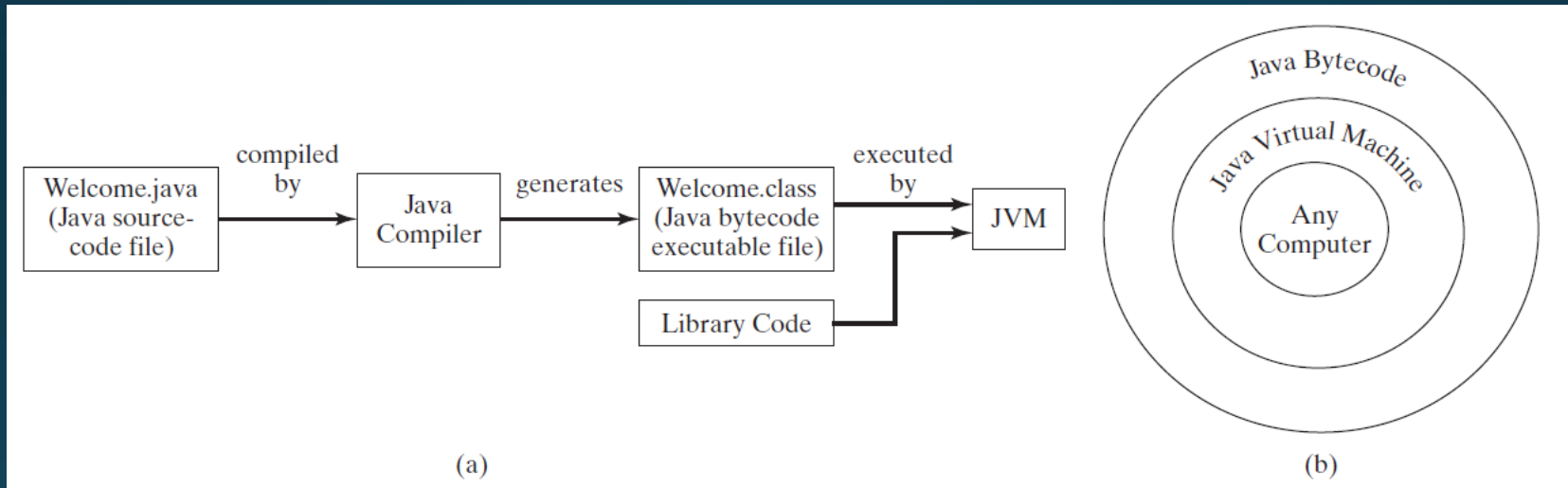
```
File Edit Format View Help
// This application program prints welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Creating, Compiling, and Running Programs



Compiling Java Source Code

- Java, source code is compiled into *bytecode*.
- Bytecode can run on any computer with a Java Virtual Machine (JVM).
- JVM is software that *interprets* Java bytecode.



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!");  
    }  
}
```

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!");  
    }  
}
```

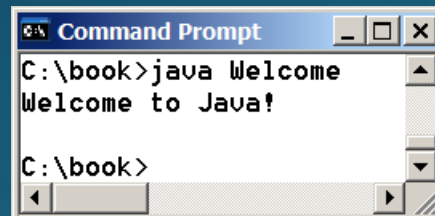
Trace a Program Execution

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!");  
    }  
}
```



The screenshot shows a Windows Command Prompt window with the title bar 'Command Prompt'. The command prompt shows the directory 'C:\book' and the command 'java Welcome' being executed. The output 'Welcome to Java!' is displayed on the next line. The prompt 'C:\book>' is visible at the bottom.

print a message to the console

Two More Simple Examples

WelcomeWithThreeMessages

Run

ComputeExpression

Run

Supplements on the Companion Website

- See Supplement I.B for installing and configuring JDK
- See Supplement I.C for compiling and running Java from the command window for details

www.pearsonhighered.com/liang

Anatomy of a Java Program

Anatomy of a Java Program

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

Class Name

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

Main Method

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

Methods

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

Statement

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

Statement Terminator

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

Reserved Words

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


Blocks

```
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!");  
    }  
}
```

Comments

- *Line comment*: Preceded by two slashes (//) on a line.
- *Paragraph comment*: Enclosed between /* and */ on one or multiple lines.
- *javadoc comment*: javadoc comments begin with /** and end with */.

|| ||
...

```
// 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.

Appropriate Comments

```
/* COMP 110
 * Section: 15095
 * Spring 2014
 * Project: 3
 * Filename: Gradebook.java
 * Programmer: Benjamin Riveira
 * Description: This is a command-line
 * application which allows the user to
 * input an arbitrary number of test
 * scores, totals and averages the
 * scores, then reports the totals and
 * averages for all scores.
 */
```

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
 - Capitalize the first letter of each word in the name.
- Variable and Method names:
 - Use all lowercase. For multi-word names, use “camel case.”


Proper Indentation and Spacing

- Indentation
 - Indent two spaces.
- Spacing
 - Use blank line to separate segments of the code.

Block Styles


Use end-of-line 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");
    }
}
```

The Importance of Good Style

- Helps reduce errors
- Makes code easier to read
- Helps your grade

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

ShowSyntaxErrors

Run

Runtime Errors

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

ShowRuntimeErrors

Run

Logic Errors

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

ShowLogicErrors

Run

Compiling and Running Java from NetBeans

- See Supplement I.D on the Website for details