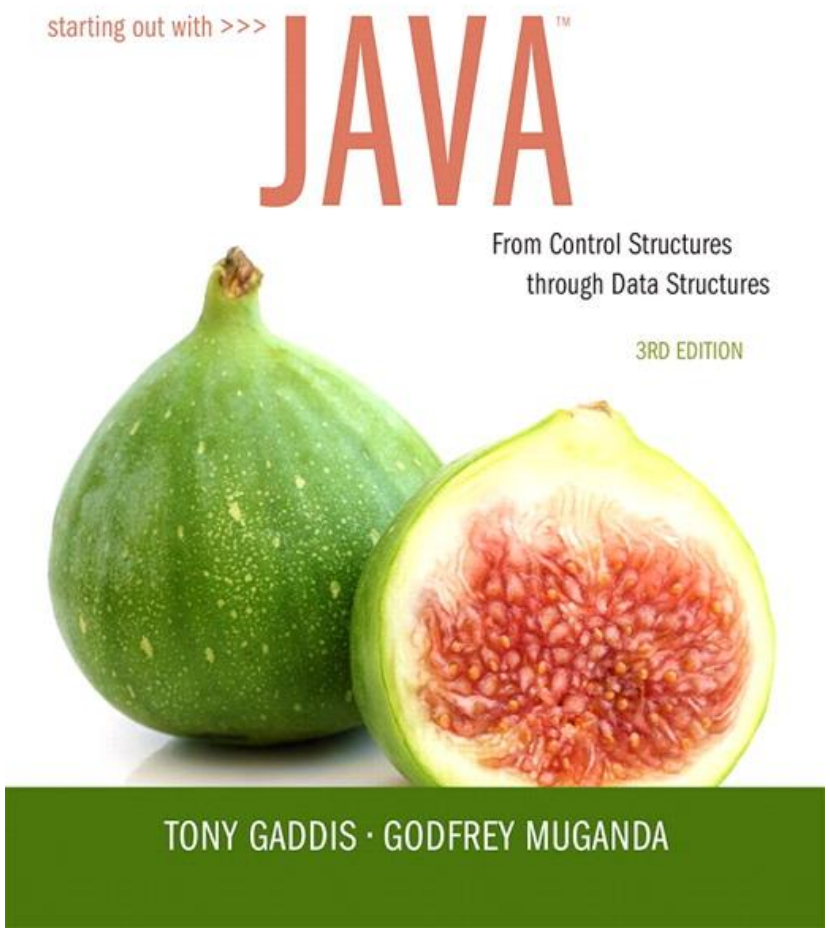
CHAPTER 1

Introduction to Computers and Java



Chapter Topics

Chapter 1 discusses the following main topics:

- Introduction
- Why Program?
- Computer Systems: Hardware and Software
- Programming Languages
- What Is a Program Made Of?
- The Programming Process
- Object-Oriented Programming

Java History

- 1991 - Green Team started by Sun Microsystems (now owned by Oracle).
- *7 Handheld controller for multiple entertainment systems.
- There was a need for a programming language that would run on various devices.
- Java (first named Oak) was developed for this purpose.

Introduction

- Java enabled web browser (*HotJava*) demonstrated at 1995 Sun World conference.
- Java incorporated into Netscape shortly after.
- Java is “cross platform”, meaning that it can run on various computer operating systems.

Java Applications and Applets

- Java programs can be of two types:
  - Applications
    - Stand-alone programs that run without the aid of a web browser.
    - Relaxed security model since the user runs the program locally.
  - Applets
    - Small applications that require the use of a Java enabled web browser to run.
    - Enhanced security model since the user merely goes to a web page and the applet runs itself.

Why Program?

- Computers are tools that can be programmed to perform many functions, such as:
  - spreadsheets
  - databases
  - word processing
  - games
  - etc.
- Computers are versatile because they can be programmed.
- Computer Programmers implement programs that perform these functions.

Why Program?

Aspects of a computer program that must be designed:

- The logical flow of the instructions
- The mathematical procedures
- The layout of the programming statements
- The appearance of the screens
- The way information is presented to the user
- The program's "user friendliness"
- Manuals, help systems, and/or other forms of written documentation.

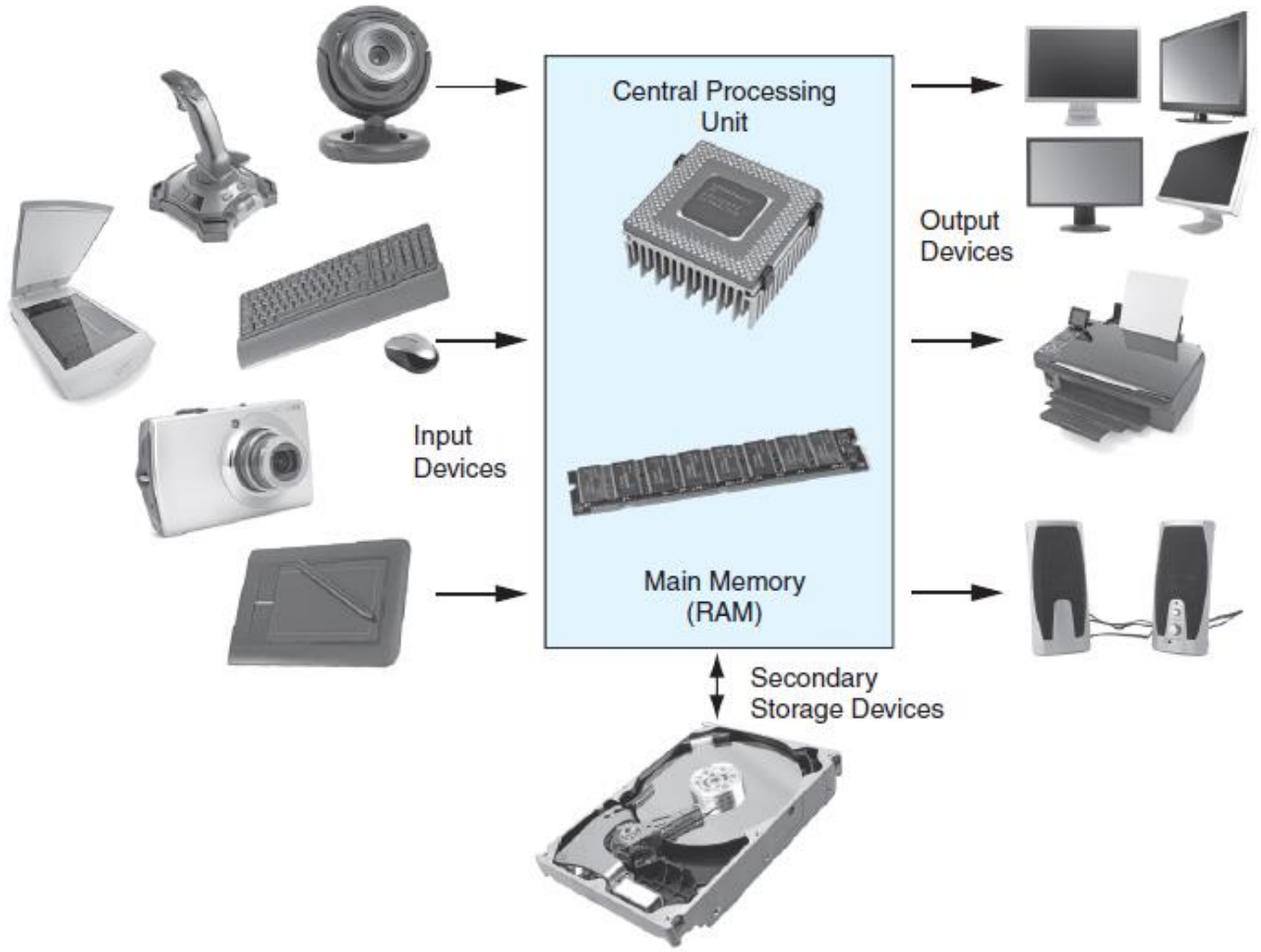
Why Program?

- Programs must be analytically correct as well.
- Programs rarely work the first time they are programmed.
- Programmers must perform the following on a continual basis:
  - analyze,
  - experiment,
  - correct, and
  - redesign.
- Programming languages have strict rules, known as *syntax*, that must be carefully followed.

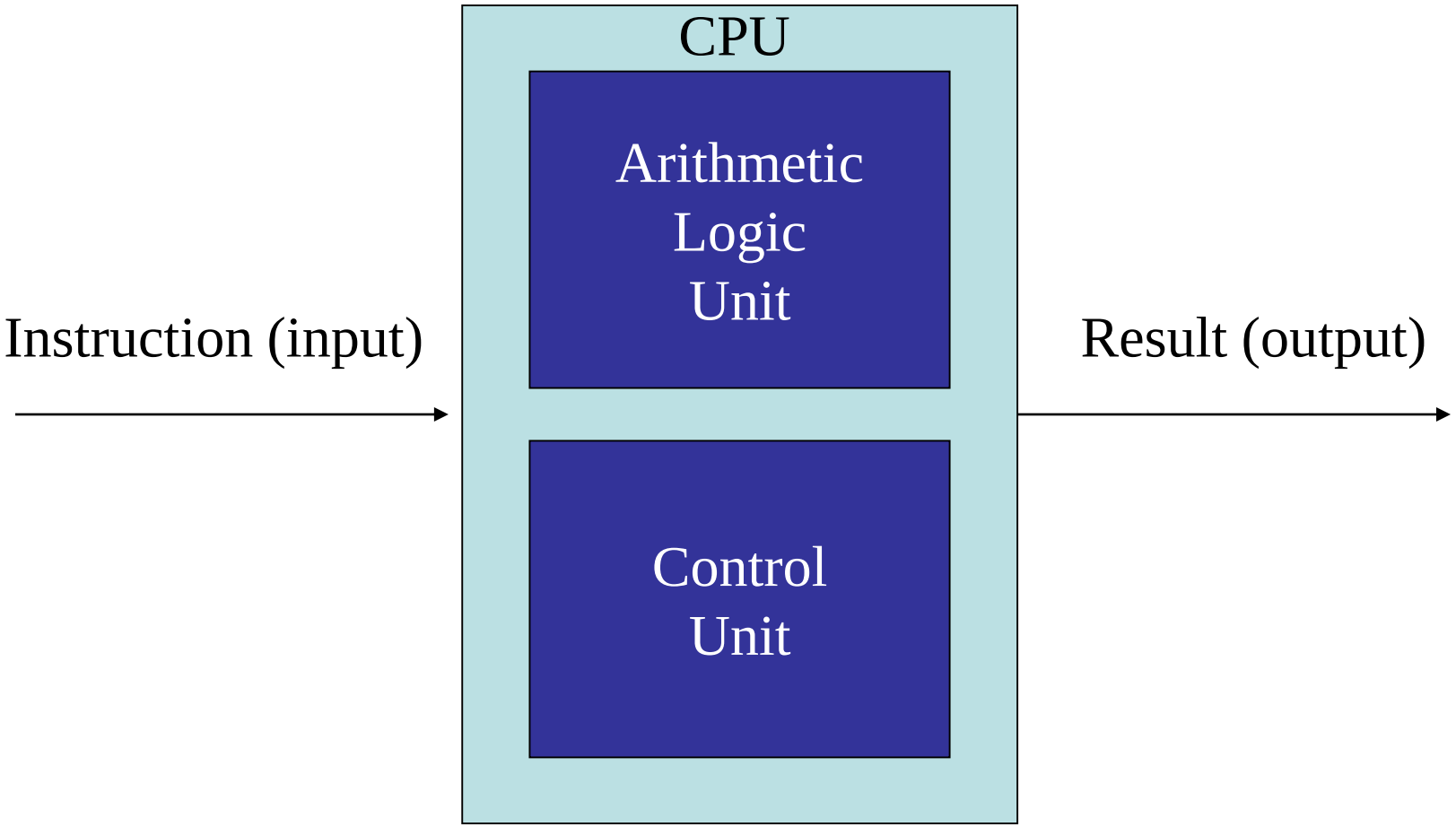
Computer Systems: Hardware

- Computer hardware components are the physical pieces of the computer.
- The major hardware components of a computer are:
  - The central processing unit (CPU)
  - Main memory
  - Secondary storage devices
  - Input and Output devices

Computer Systems: Hardware

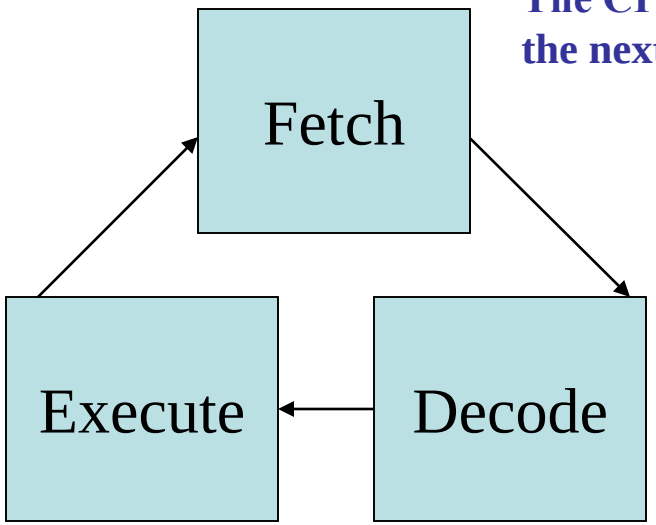


Computer Systems: Hardware
Central Processing Unit



Computer Systems: Hardware
Central Processing Unit

– The CPU performs the fetch, decode, execute cycle in order to process program information.



The CPU's control unit fetches, from main memory, the next instruction in the sequence of program instructions.

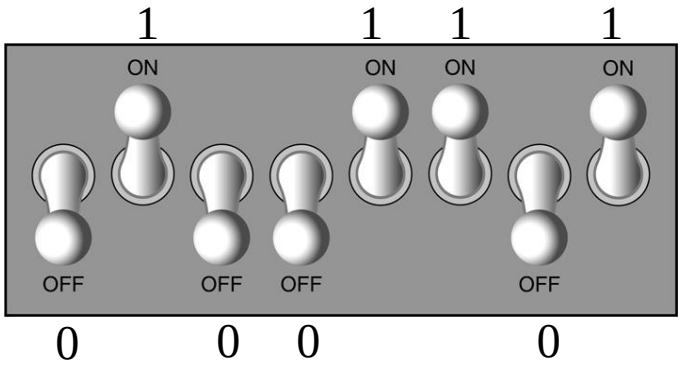
The instruction is encoded in the form of a number. The control unit decodes the instruction and generates an electronic signal.

The signal is routed to the appropriate component of the computer (such as the ALU, a disk drive, or some other device). The signal causes the component to perform an operation.

Computer Systems: Hardware
Main Memory

- Commonly known as *random-access memory (RAM)*
- RAM contains:
  - currently running programs
  - data used by those programs.
- RAM is divided into units called *bytes*.
- A byte consists of eight *bits* that may be either on or off.

Computer Systems: Hardware
Main Memory

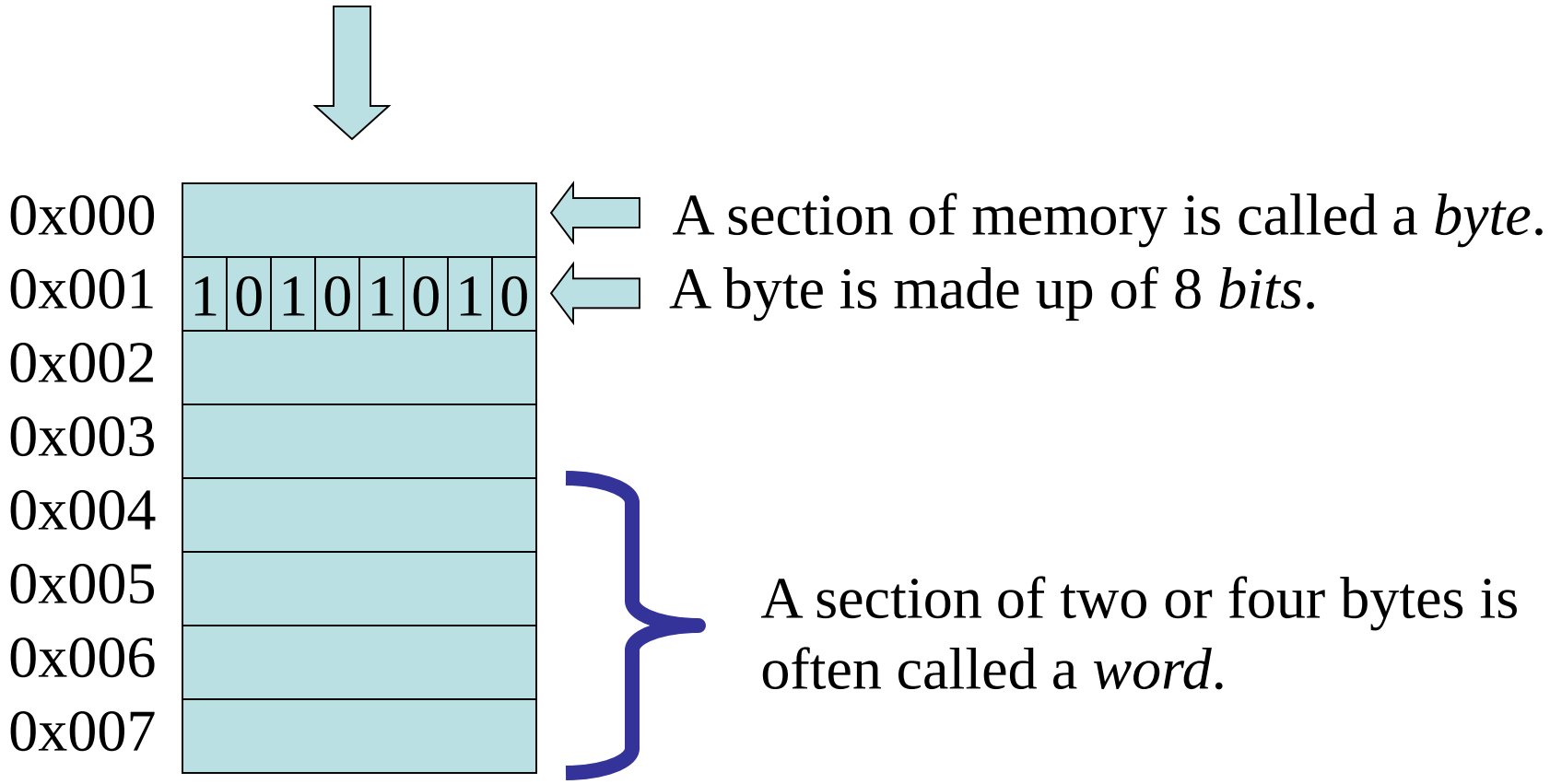


- A bit is either on or off:
  - 1 = on
  - 0 = off
- The bits form a pattern that represents a character or a number.
- Each byte in memory is assigned a unique number known as an *address*.
- RAM is *volatile*, which means that when the computer is turned off, the contents of RAM are erased.

Computer Systems: Hardware

Main Memory

Main memory can be visualized as a column or row of cells.



Computer Systems: Hardware
Secondary Storage Devices

- Secondary storage devices are capable of storing information for longer periods of time (*non-volatile*).
- Common Secondary Storage devices:
  - Disk drive
  - External drive
  - CD drive
  - Solid state drive
  - USB drive
  - DVD drive

Computer Systems: Hardware
Input Devices

- Input is any data the computer collects from the outside world.
- That data comes from devices known as *input devices*.
- Common input devices:
  - Keyboard
  - Mouse
  - Scanner
  - Digital camera

Computer Systems: Hardware
Output Devices

- Output is any data the computer sends to the outside world.
- That data is displayed on devices known as *output devices*.
- Common output devices:
  - Monitors
  - Printers
- Some devices such as disk drives perform input and output and are called *I/O devices* (input/output).

Computer Systems: Software

- Software refers to the programs that run on a computer.
- There are two classifications of software:
  - Operating Systems
  - Application Software

Computer Systems: Software
Operating Systems

- An operating system is a set of programs that manages the computer's hardware devices and controls their processes.
- Most all modern operating systems are multitasking.

Computer Systems: Software
Operating Systems

- A multitasking operating system is capable of running multiple programs at once.
  - Unix
  - Linux
  - Mac OS
  - Windows
- The technique is called time sharing.
- A multitasking system divides the allocation of hardware resources and the attention of the CPU among all the executing programs.

Computer Systems: Software
Application Software

- *Application software* refers to programs that make the computer useful to the user.
- Application software provides a more specialized type of environment for the user to work in.
- Common application software:
  - Spreadsheets
  - Word processors
  - Accounting software
  - Tax software
  - Games

Programming Languages

- A program is a set of instructions a computer follows in order to perform a task.
- A programming language is a special language used to write computer programs.
- A computer program is a set of instructions that enable the computer to solve a problem or perform a task.
- Collectively, these instructions form an *algorithm*

Programming Languages

- An algorithm is a set of well defined steps to completing a task.
- The steps in an algorithm are performed sequentially.
- A computer needs the algorithm to be written in *machine language*.
- Machine language is written using *binary numbers*.
- The binary numbering system (base 2) only has two digits (0 and 1).

Programming Languages

- The binary numbers are encoded as a machine language.
- Each CPU has its own machine language.
  - Motorola 68000 series processors
  - Intel x86 series processors
  - ARM processors, etc.
- Example of a machine language instruction:

  1011010000000101

Programming Languages

- In the distant past, programmers wrote programs in machine language.
- Programmers developed higher level programming languages to make things easier.
- The first of these was *assembler*.
- Assembler made things easier but was also processor dependent.

Programming Languages

- High level programming languages followed that were not processor dependent.
- Some common programming languages:

Java	C	Visual Basic
BASIC	C++	Python
COBOL	C#	Ruby
Pascal	PHP	JavaScript

Programming Languages
Common Language Elements

- There are some concepts that are common to virtually all programming languages.
- Common concepts:
  - Key words
  - Operators
  - Punctuation
  - Programmer-defined identifiers
  - Strict syntactic rules.

Programming Languages
Sample Program

```
public class HelloWorld
{
  public static void main(String[] args)
  {
    String message = "Hello World";
    System.out.println(message);
  }
}
```

Programming Languages
Sample Program

- Key words in the sample program are:
  - `public`
  - `class`
  - `static`
  - `void`
- Key words are lower case (Java is a case sensitive language).
- Key words cannot be used as a programmer-defined identifier.

Programming Languages

- Semi-colons are used to end Java statements; however, not all lines of a Java program end a statement.
- Part of learning Java is to learn where to properly use the punctuation.

Programming Languages
Lines vs Statements

- There are differences between lines and statements when discussing source code.

```
System.out.println(
  message);
```

- This is one Java statement written using two lines. Do you see the difference?
- A statement is a complete Java instruction that causes the computer to perform an action.

Programming Languages
Variables

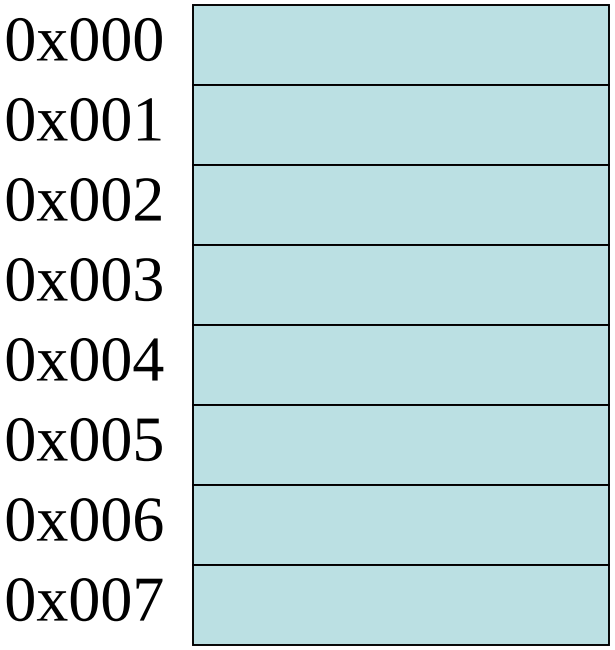
- Data in a Java program is stored in memory.
- Variable names represent a location in memory.
- Variables in Java are sometimes called fields.
- Variables are created by the programmer who assigns it a programmer-defined identifier.

example: `int hours = 40;`

- In this example, the variable *hours* is created as an integer (more on this later) and assigned the value of 40.

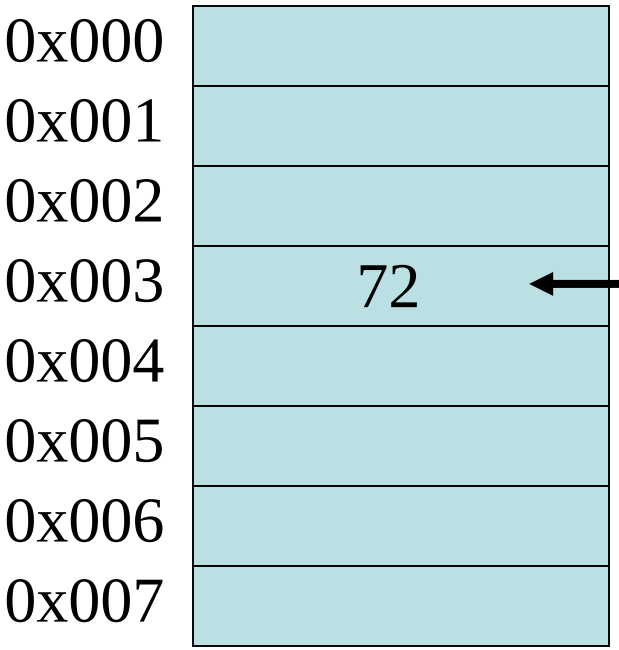
Programming Languages
Variables

- Variables are simply a name given to represent a place in memory.



Programming Languages
Variables

The Java Virtual Machine (JVM) actually decides where the value will be placed in memory.



Assume that the this variable declaration has been made.

```
int length = 72;
```

The variable length is a symbolic name for the memory location 0x003.

The Compiler and the Java Virtual Machine

- A programmer writes Java programming statements for a program.
- These statements are known as *source code*.
- A *text editor* is used to edit and save a Java *source code file.*
- Source code files have a *.java* file extension.
- A *compiler* is a program that translates source code into an executable form.

The Compiler and the Java Virtual Machine

- A compiler is run using a source code file as input.
- Syntax errors that may be in the program will be discovered during compilation.
- *Syntax errors* are mistakes that the programmer has made that violate the rules of the programming language.
- The compiler creates another file that holds the translated instructions.

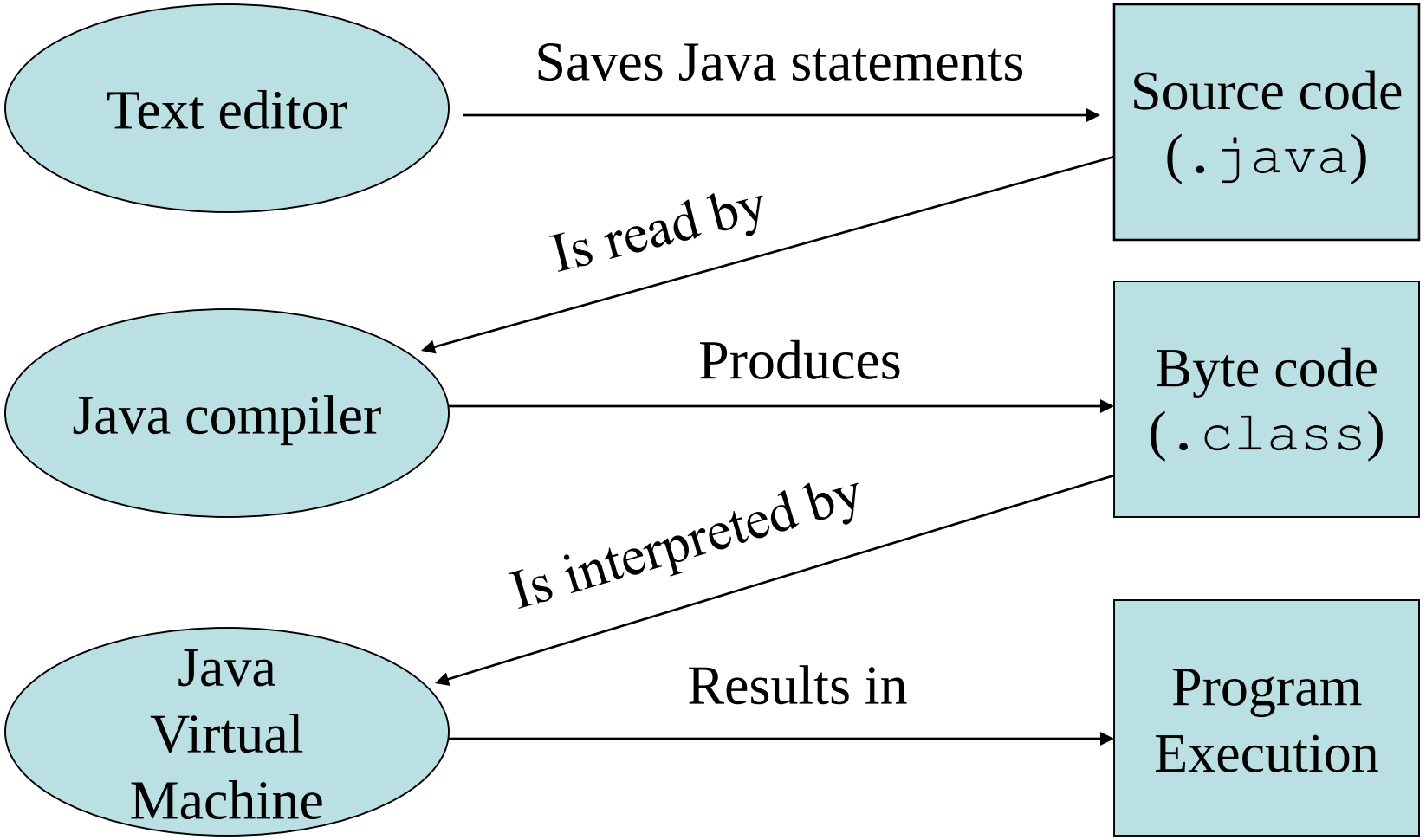
The Compiler and the Java Virtual Machine

- Most compilers translate source code into *executable* files containing *machine code*.
- The Java compiler translates a Java source file into a file that contains *byte code* instructions.
- Byte code instructions are the machine language of the *Java Virtual Machine (JVM)* and cannot be directly executed directly by the CPU.

The Compiler and the Java Virtual Machine

- Byte code files end with the *.class* file extension.
- The JVM is a program that *emulates* a micro-processor.
- The JVM executes instructions as they are read.
- JVM is often called an *interpreter*.
- Java is often referred to as an *interpreted language*.

Program Development Process



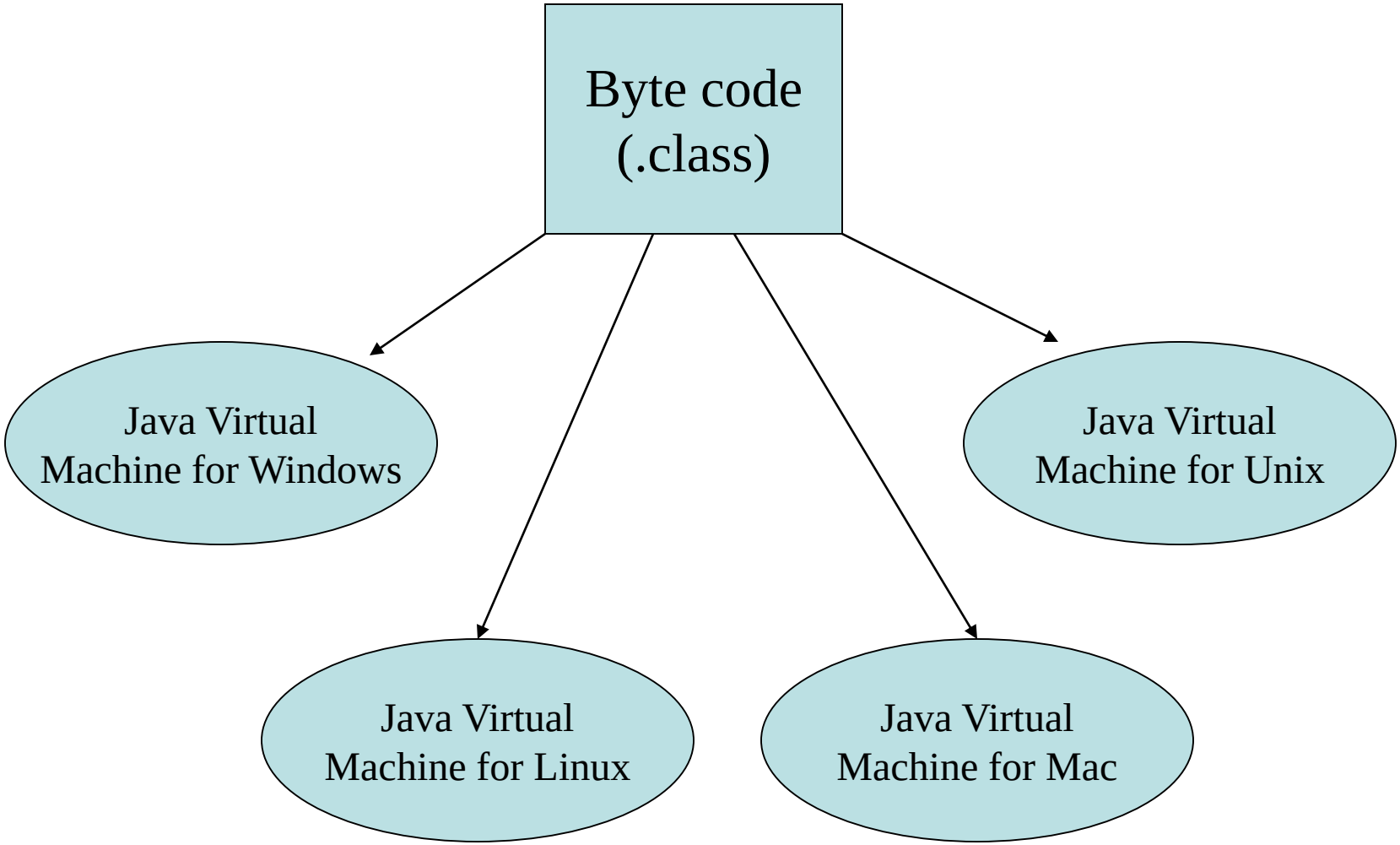
Portability

- *Portable* means that a program may be written on one type of computer and then run on a wide variety of computers, with little or no modification.
- Java byte code runs on the JVM and not on any particular CPU; therefore, compiled Java programs are highly portable.
- JVMs exist on many platforms:
  - Windows
  - Mac
  - Linux
  - Unix
  - BSD
  - Etc.

Portability

- With most programming languages, portability is achieved by compiling a program for each CPU it will run on.
- Java provides an JVM for each platform so that programmers do not have to recompile for different platforms.

Portability



Java Versions

- The software you use to write Java programs is called the Java Development Kit, or JDK.
- There are different editions of the JDK:
  - Java SE - Java2 *Standard Edition*.
  - Java EE - Java2 *Enterprise Edition*.
  - Java ME - Java2 *Micro Edition*.
- Available for download at `http://java.oracle.com`

Compiling a Java Program

- The Java compiler is a *command line* utility.
- The command to compile a program is:

  java filename.java

- javac is the Java compiler.
- The `.java` file extension must be used.

Example: To compile a java source code file named Payroll.java you would use the command:

```
javac Payroll.java
```

The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.
4. Check the model for logical errors.

The Programming Process

5. Enter the code and compile it.
6. Correct any errors found during compilation.

   Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any runtime errors found while running the program.

   Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

Software Engineering

- Encompasses the whole process of crafting computer software.
- Software engineers perform several tasks in the development of complex software projects.
  - designing,
  - writing,
  - testing,
  - debugging,
  - documenting,
  - modifying, and
  - maintaining.

Software Engineering

- Software engineers develop:
  - program specifications,
  - diagrams of screen output,
  - diagrams representing the program components and the flow of data,
  - pseudocode,
  - examples of expected input and desired output.

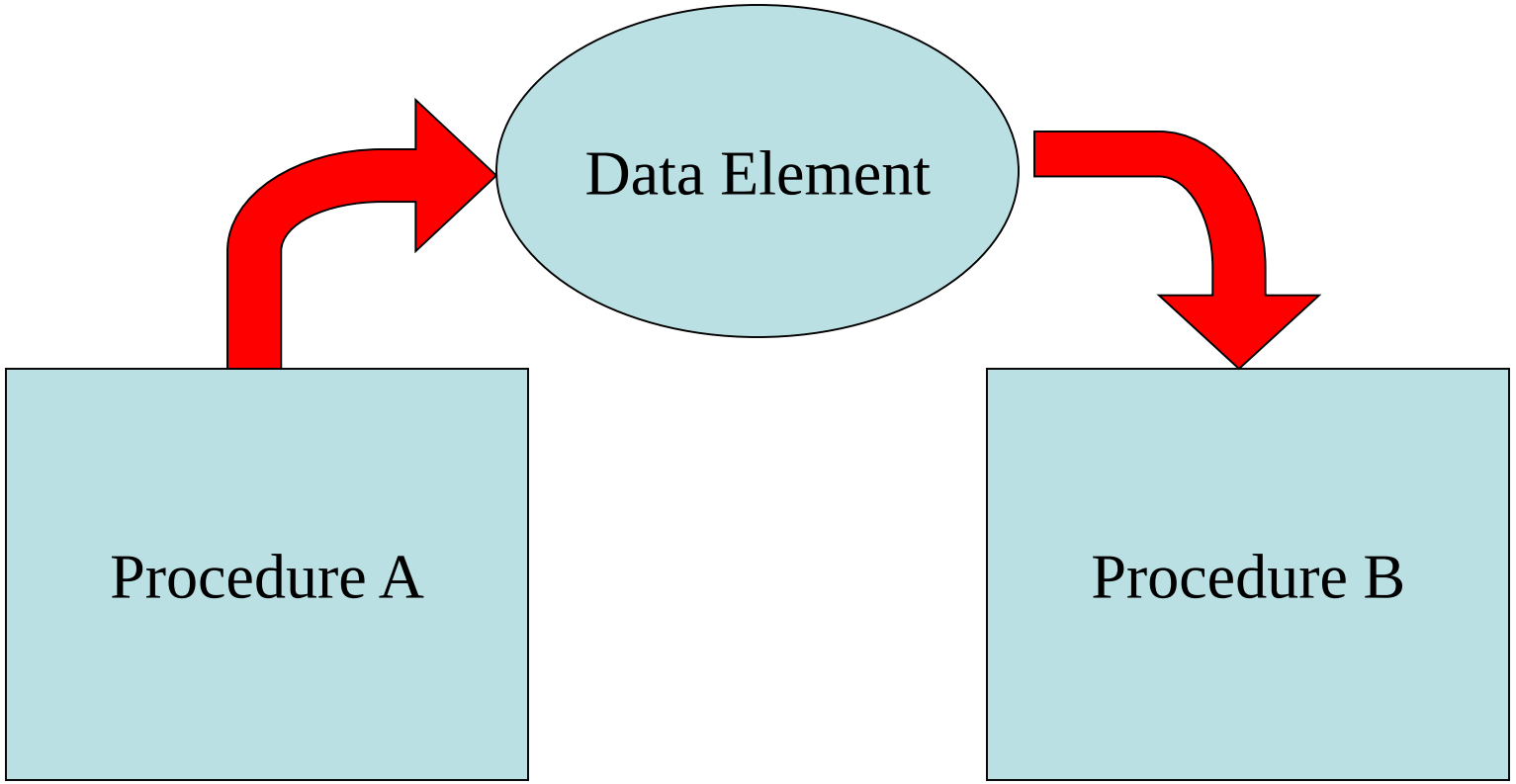
Software Engineering

- Software engineers also use special software designed for testing programs.
- Most commercial software applications are large and complex.
- Usually a team of programmers, not a single individual, develops them.
- Program requirements are thoroughly analyzed and divided into subtasks that are handled by
  - individual teams
  - individuals within a team.

Procedural Programming

- Older programming languages were procedural.
- A *procedure* is a set of programming language statements that, together, perform a specific task.
- Procedures typically operate on data items that are separate from the procedures.
- In a procedural program, the data items are commonly passed from one procedure to another.

Procedural Programming



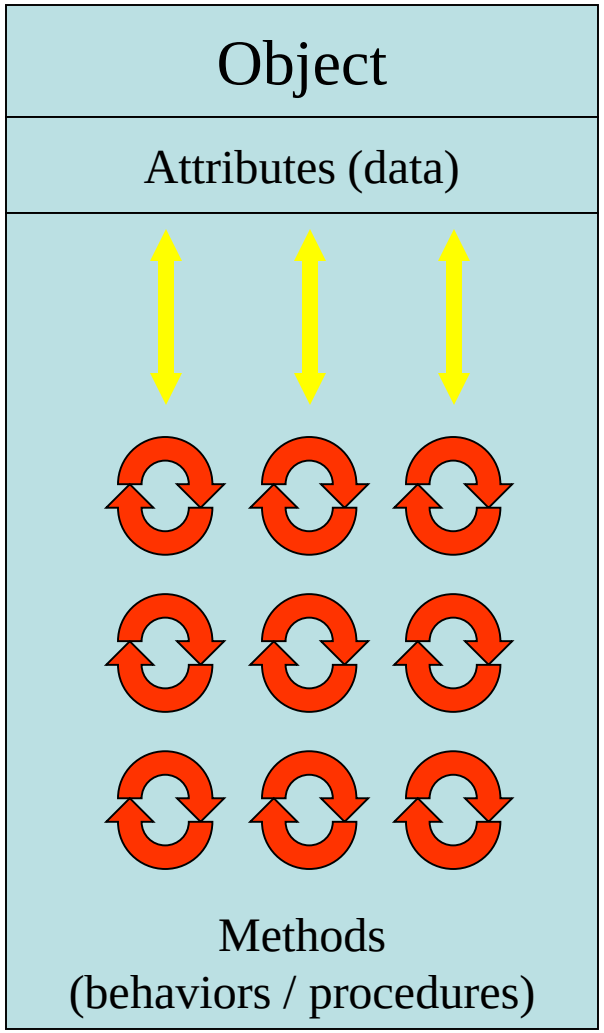
Procedural Programming

- In procedural programming, procedures are developed to operate on the program's data.
- Data in the program tends to be global to the entire program.
- Data formats might change and thus, the procedures that operate on that data must change.

Object-Oriented Programming

- Object-oriented programming is centered on creating objects rather than procedures.
- Objects are a melding of data and procedures that manipulate that data.
- Data in an object are known as *attributes*.
- Procedures in an object are known as *methods*.

Object-Oriented Programming



Object-Oriented Programming

- Object-oriented programming combines data and behavior via *encapsulation*.
- *Data hiding* is the ability of an object to hide data from other objects in the program.
- Only an objects methods should be able to directly manipulate its attributes.
- Other objects are allowed manipulate an object's attributes via the object's methods.
- This indirect access is known as a *programming interface*.

Object-Oriented Programming