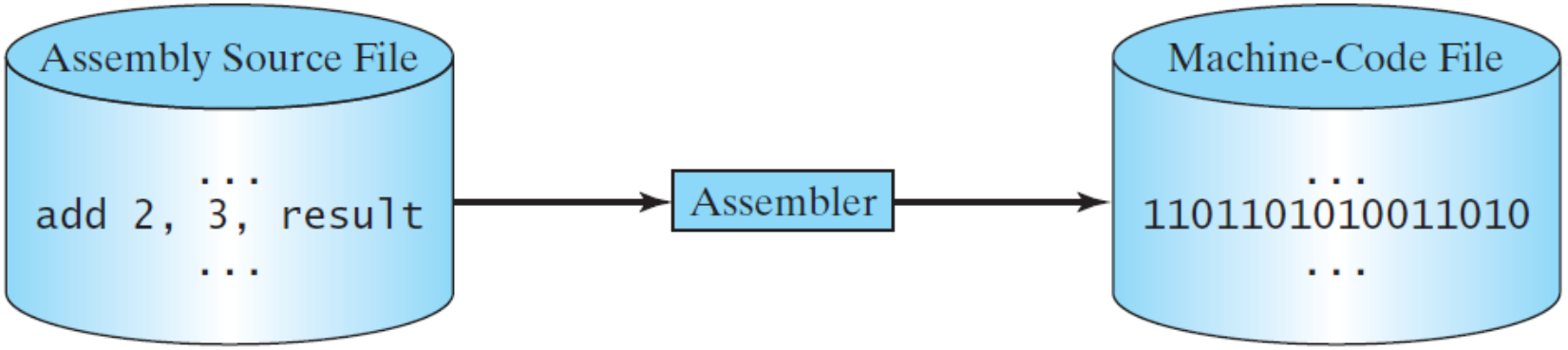
Chapter 1 Introduction to Computers, Programs, and Java

Hu Zheng
huzheng@bupt.edu.cn

Programming Languages

Machine Language Assembly Language High-Level Language

```
1101101010011010
```




```
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

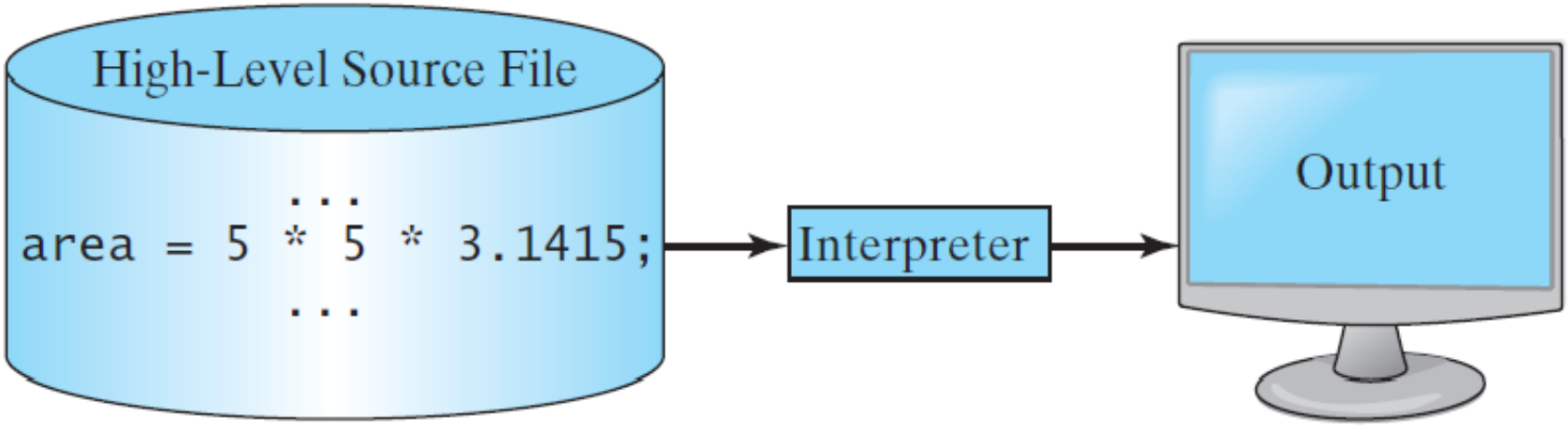
A program written in a high-level language is called a *source program* or *source code*.

A source program must be translated into machine code for execution.

A Translation can be done using another programming tool called an *interpreter* or a *compiler*.

Interpreting Source Code

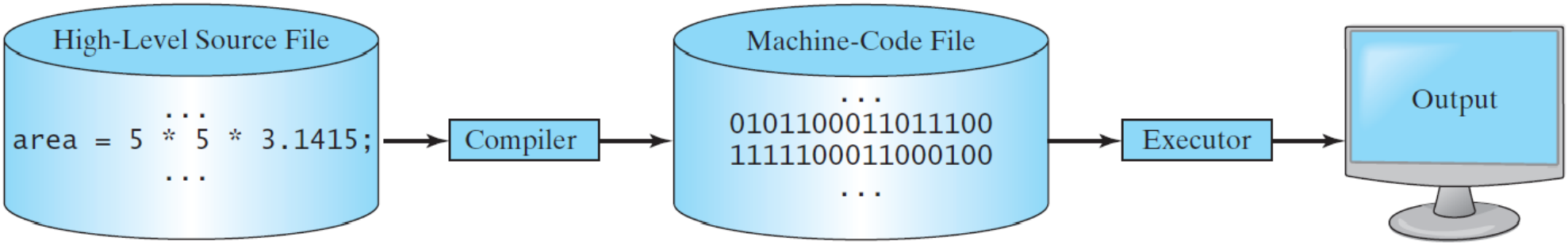
An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away.



Note: a statement from the source code may be translated into several machine instructions.

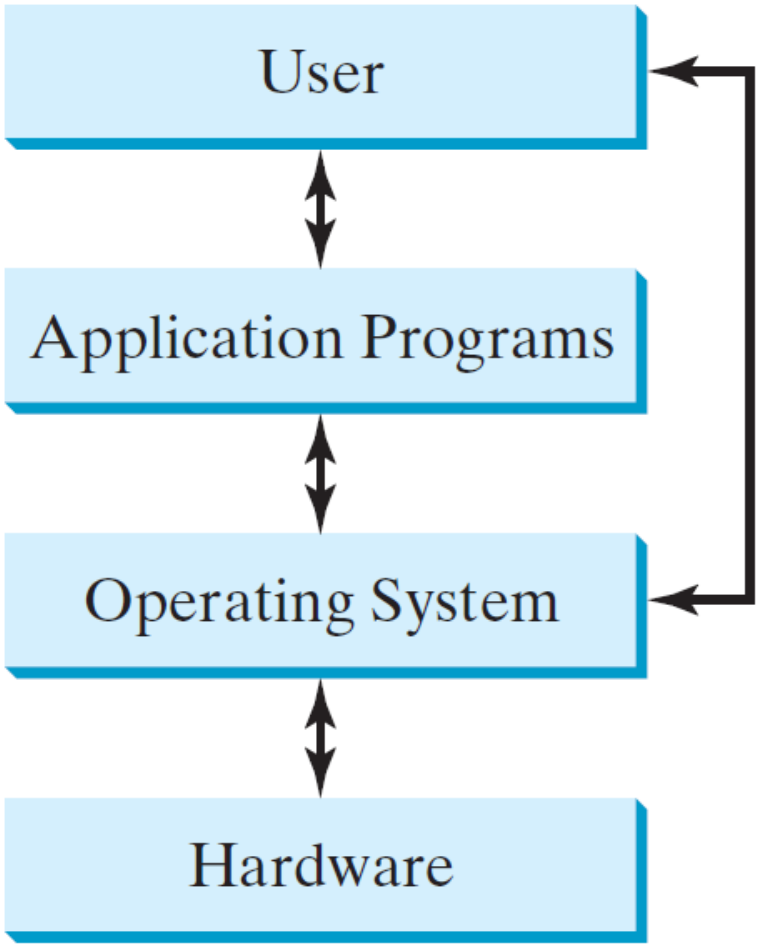
Compiling Source Code

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



Operating Systems

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



Why Java?

The future of computing is being profoundly influenced by the Internet

- Java is a general purpose programming language.
- Java is the Internet programming language.
- Whoops, there are new languages…
  - however, blah blah blah

TIOBE 2020年9月编程语言1-10排行榜

TIOBE Index for September 2020

September Headline: Programming Language C++ is doing very well

Back in 2003, the programming language C++ was a real winner. It peaked at 17.53% in August 2003, being close to the number #2 position and becoming winner of the programming language award of 2003. From then on C++ went downhill. After 2005 it didn't hit the 10% any more and in 2017 it scored an all time low of 4.55%. But if compared to last year, C++ is now the fastest growing language of the pack (+1.48%). I think that the new C++20 standard might be one of the main causes for this. Especially because of the new modules feature that is going to replace the dreadful include mechanism. C++ beats other languages with a positive trend such as R (+1.33%) and C# (+1.18%). On the other hand, Java is in real trouble with a loss of -3.18% in comparison to last year. - *Paul Jansen, CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	15.95%	+0.74%
2	1	↓	Java	13.48%	-3.18%
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	↑	PHP	2.49%	+0.62%
9	19	⇈	R	2.37%	+1.33%
10	8	↓	SQL	1.76%	-0.19%

Platform JAVA

Culture

Ecosystem

Java, Web, and Beyond

- Java can be used to develop standalone applications.
- Java can be used to develop applications running from a browser.
- Java can also be used to develop applications for hand-held devices.
- Java can be used to develop applications for Web servers.
- Big Data
- Machine Learning? Weka …Deeplearning4J

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

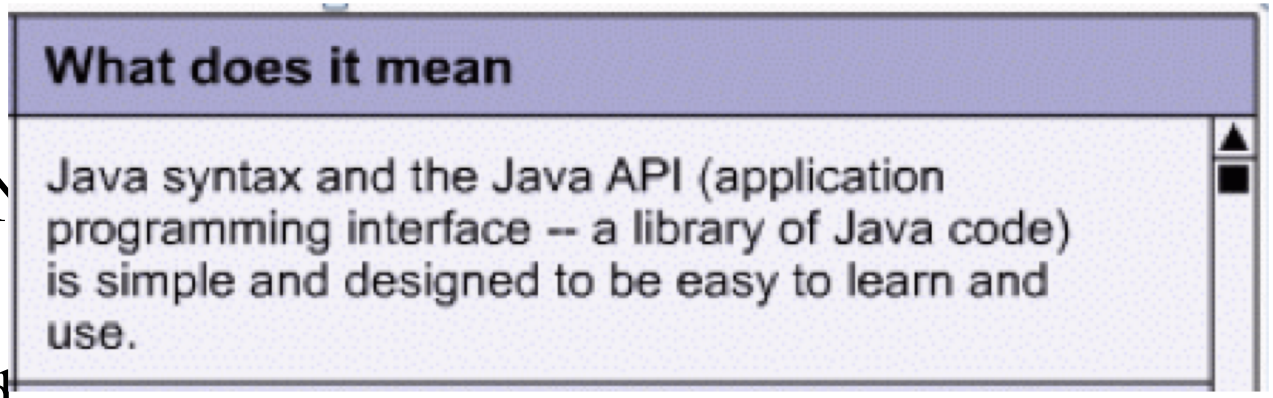
Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-N
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Java is partially modeled on C++, but greatly simplified and improved. Some people refer to Java as "C++--" because it is like C++ but with more functionality and fewer negative aspects.



Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Java is inherently object-oriented. Java was designed from the start to be object-oriented. Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.

One of the central issues in software development is how to reuse code. Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism.

The idea behind objects is quite simple. Instead of designing a long and complex set of instructions or routines to perform one or more tasks, the programmer can break these into various parts. Each part can perform one or more discrete tasks and contains all the data needed to perform the task. This is an object. Java files define an object or a class of objects.

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithre
- Java Is Dynamic

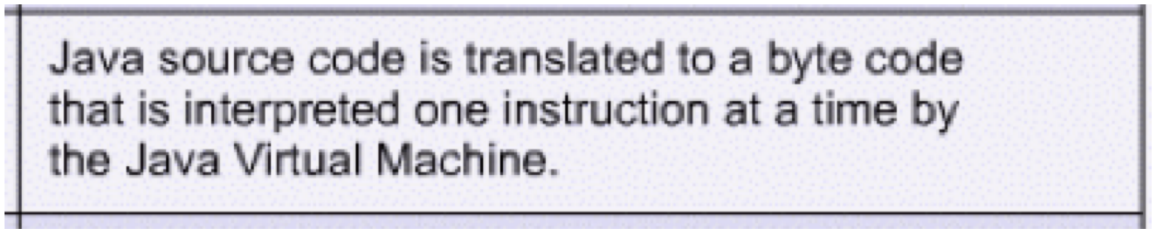
Distributed computing involves several computers working together on a network. Java is designed to make distributed computing easy. Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.

Java language provides a library of program routines to open and access objects across the Internet via URLs with the same ease as when accessing a local file system. You can send data across the net easily with Java programs.

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).



Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Java compilers can detect many problems that would first show up at execution time in other languages.

Java has eliminated certain types of error-prone programming constructs found in other languages.

Java has a runtime exception-handling feature to provide programming support for robustness.

The design of the Java platform ensures that the programs run correctly and do not break when the unexpected happens.

Companion Website

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithi
- Java Is Dynami

Java implements several security mechanisms to protect your system against harm caused by stray programs.

Java programs can be downloaded from a location on the network. Such downloaded Java programs cannot access files or destroy programs on your computer.

Companion Website

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Java language programs can be written on one machine and run on a machine with a different type of CPU.

Write once, run anywhere

With a Java Virtual Machine (JVM), you can write one program that will run on any platform.

Companion Website

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.

Companion Website

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Multithreading is when multiple Java programs are running simultaneously and sharing data and instructions. Multithreading provides the user the ability to view an animation or video clip, listen to music in a browser, and search for information on the World Wide Web.

Multithread programming is smoothly integrated in Java, whereas in other languages you have to call procedures specific to the operating system to enable multithreading.

Companion Website

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neut
- Java Is Portable
- Java Is Multithreaded
- Java Is Dynamic

Dynamic has several meanings as it applies to the Java language. For example, the language has special capabilities that allow the program to call upon resources as needed dynamically when the program is running in memory. Such a resource can be another Java routine. The software development process can also be dynamic. Java allows a programmer to approach a solution to a problem in incremental steps. At each step of the solution, a small number of routines can be built with newer routines using the work of already defined routines. This encourages reuse of code.

Java was designed to adapt to an evolving environment. New code can be loaded on the fly without recompilation. There is no need for developers to create, and for users to install, major new software versions. New features can be incorporated transparently as needed.

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) or Java 5
- JDK 1.6 (2006) or Java 6
- JDK 1.7 (2011) or Java 7
- JDK 1.8 (2014) or Java 8
- Java 9 (2017)
- Java 10 (2018)
- Java 11(18.9 LTS) (2018)
- Java 12(2019)
- Java 13 & 14 (2020)

Oracle Java SE Support Roadmap*†

Release	GA Date	Premier Support Until	Extended Support Until	Sustaining Support
6	December 2006	December 2015	December 2018	Indefinite
7	July 2011	July 2019	July 2022*****	Indefinite
8**	March 2014	March 2022	March 2025	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	September 2026	Indefinite
12 (non-LTS)	March 2019	September 2019	Not Available	Indefinite
13 (non-LTS)	September 2019***	March 2020	Not Available	Indefinite

JDK Editions

- Java Standard Edition (Java SE)
  - can be used to develop client-side standalone applications or applets.
  - This class uses J2SE to introduce Java programming.
- Java Enterprise Edition (Java EE)
  - can be used to develop server-side applications such as Java servlets, Java Server Pages, and Java ServerFaces.
- Java Micro Edition (Java ME).
  - can be used to develop applications for mobile devices such as cell phones.
  - Java SDK for Android is not Jave ME

Java API

Application Programming Interface

- 1) Java Core API
- 2) Java Standard Extension API (javax)
- 3) Provided by Vendors or Organizations (3rd parties)

Editors and Popular Java IDEs

- Common Text Editors
  - NotePad
  - TextEdit
  - Editplus
  - UltraEdit
- IDE (Integrated Development Environment)
  - Eclipse http://www.eclipse.org/ recommended
  - IntelliJ IDEA recommended （IDEA 校园版）
    https://www.jetbrains.com/community/education/
    https://www.jetbrains.com/student/
  - NetBeans

JDK Installation

- https://docs.oracle.com/en/java/javase/11/install/overview-jdk-installation.html

Installation

JDK 8 recommended

The JDK can be installed on the following platforms:

- Oracle Solaris
- Microsoft Windows
- Linux
- macOS

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

You can download JDK from Java SE Development Kit Downloads page.

For supported processors and browsers, see Oracle JDK Certified System Configurations.

The JDK documentation is a separate download. See Java SE Documentation.

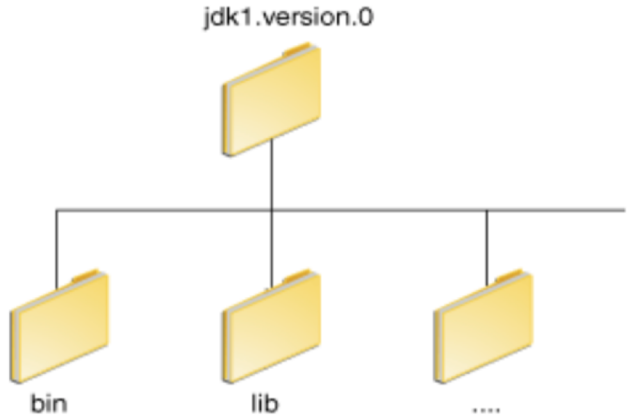
PATH and CLASSPATH

- https://docs.oracle.com/javase/tutorial/essential/environment/paths.html

PATH and CLASSPATH

This section explains how to use the PATH and CLASSPATH environment variables on Microsoft Windows, Solaris, and Linux. Consult the installation instructions includ bundle for current information.

After installing the software, the JDK directory will have the structure shown below.



PATH=/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin:$PATH

CLASSPATH=.:/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/lib *其实可以不专门配置*

A Simple Java Program

Example 1.1

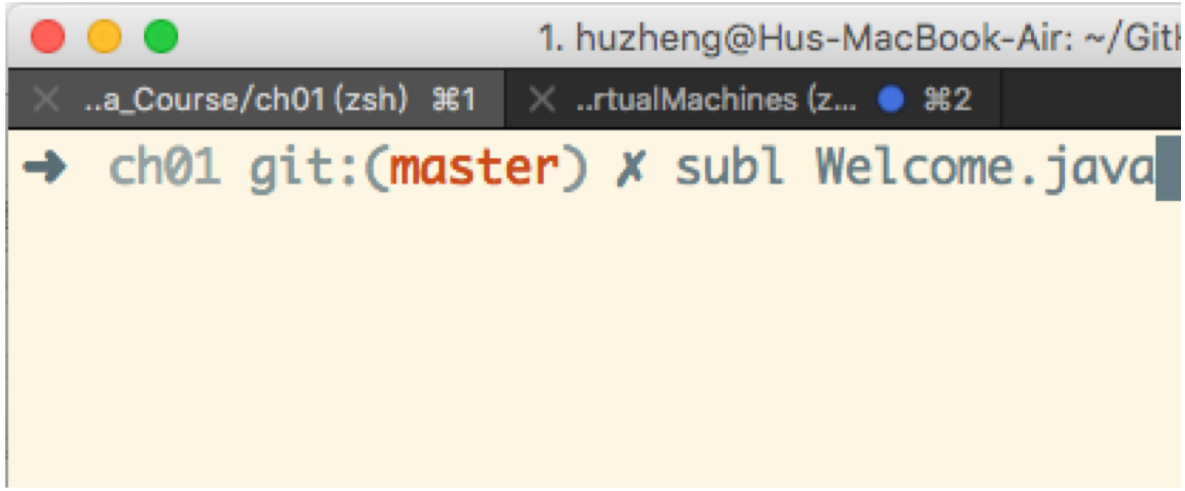
```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java
  world!");
  }
}
```

Animation 

Java as a language!

Creating and Editing Using NotePad

To use text editor for Welcome.java from the Command Window.



Welcome.java

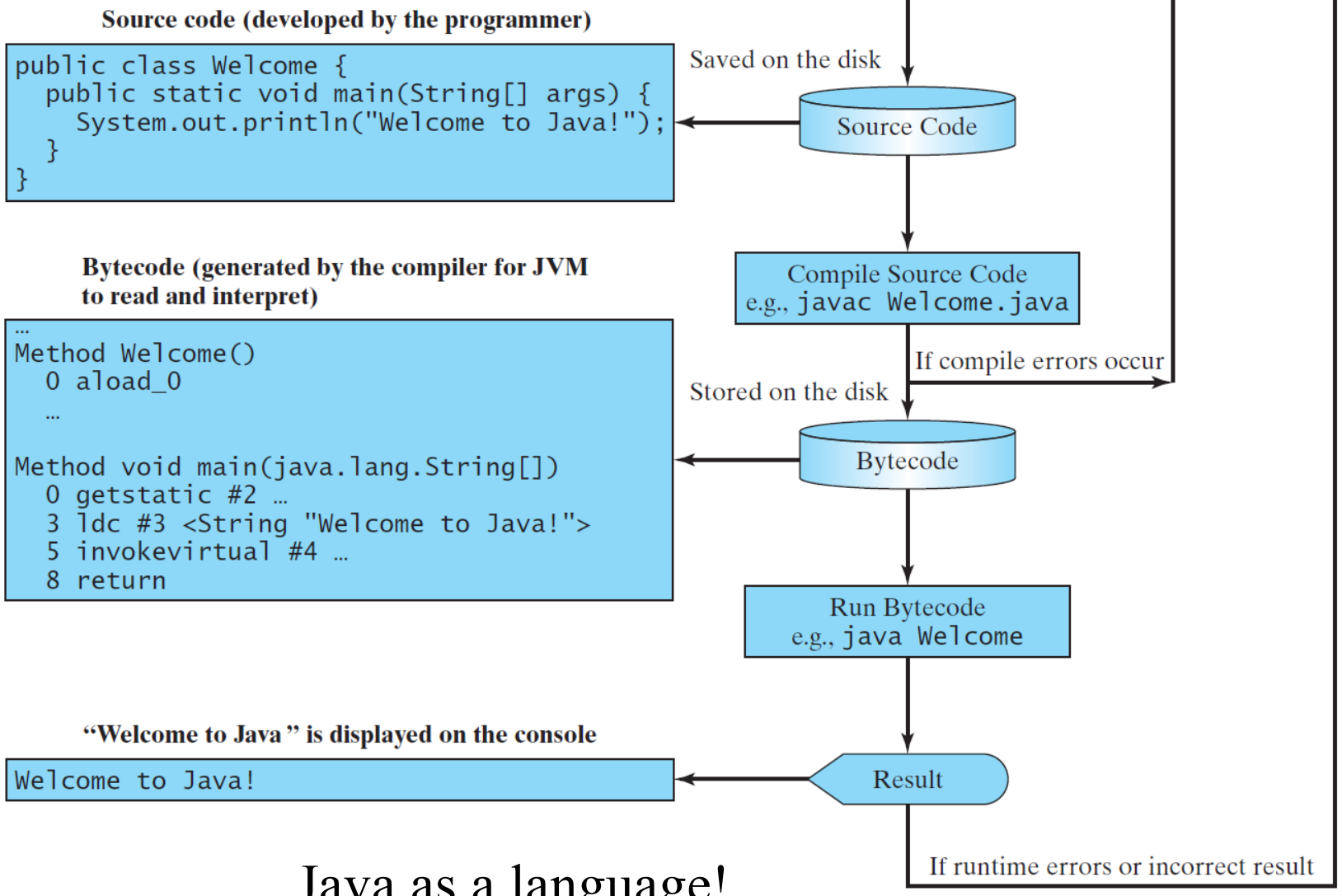
```
1 // This application program prints Welcome to Java!
2
3 package ch01;
4
5 public class Welcome {
6   public static void main(String[] args) {
7     // Display message Welcome to Java! on the console
8     System.out.println("Welcome to Java!");
9   }
10 }
```

Creating, Compiling, and Running Programs

```java
// This application program prints Welcome to Java!

package ch01;

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



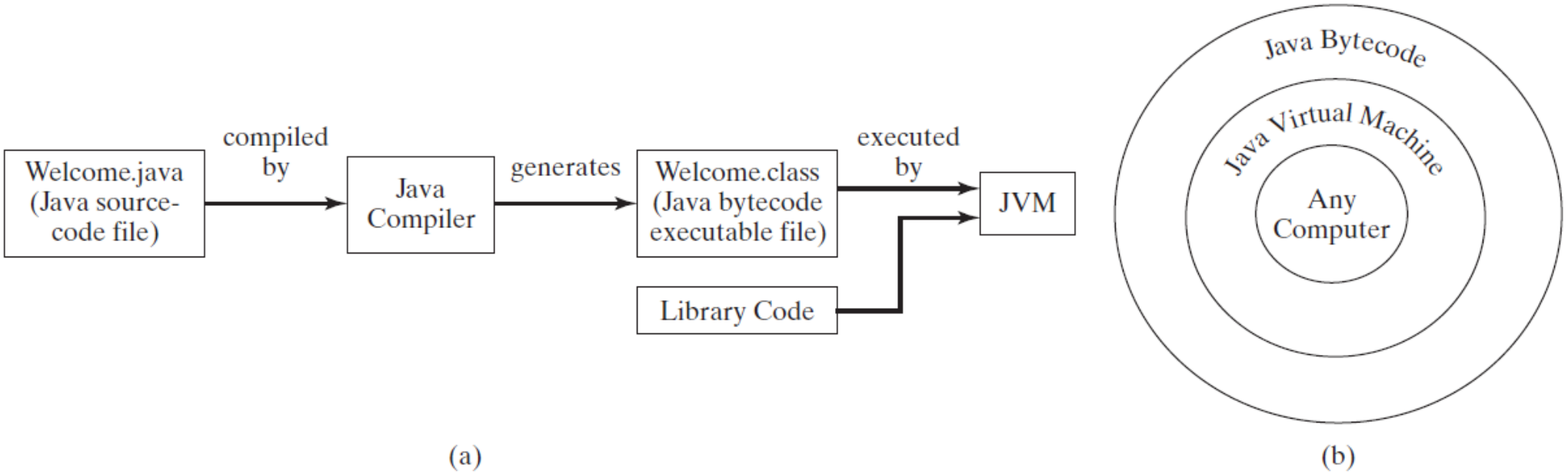
Java as a language!

Compiling Java Source Code

Write the program once, and compile the source program into a special type of object code, known as *bytecode*.

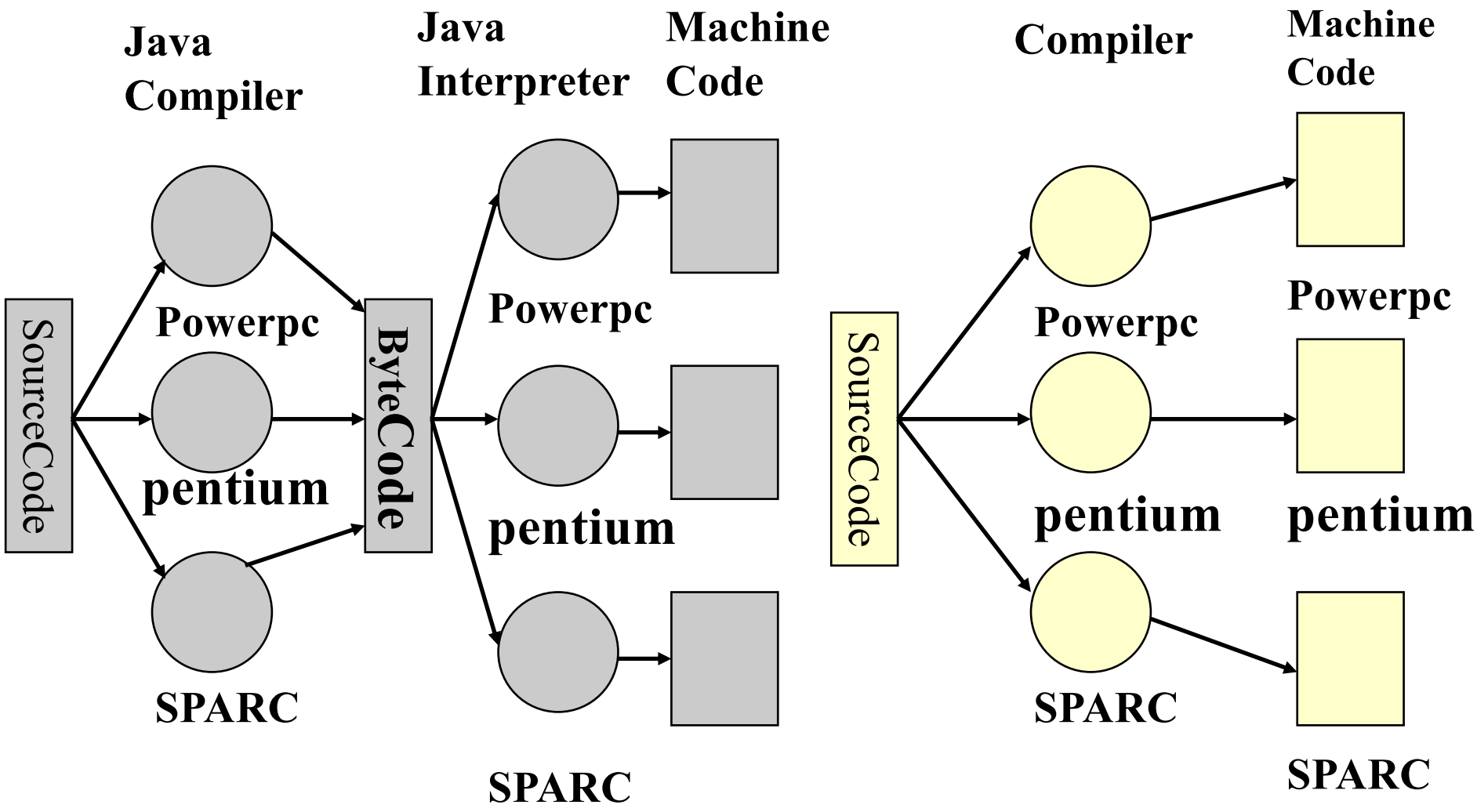
The bytecode can then run on any computer with a Java Virtual Machine.

Java Virtual Machine is a software that interprets Java bytecode.



Java Virtual Machin (JVM) as a Platform!

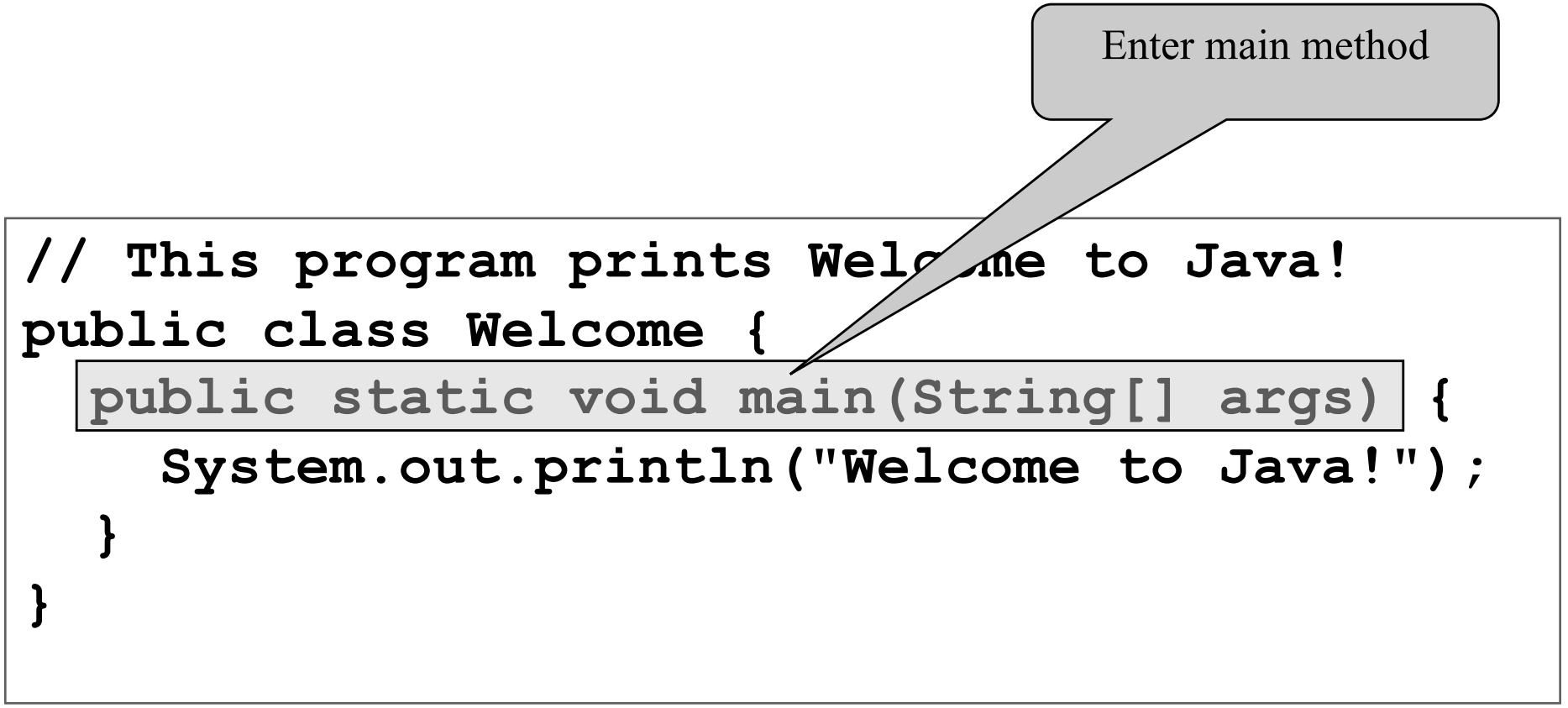
Neutral/Platform Independent



Java Virtual Machin (JVM) as a Platform!

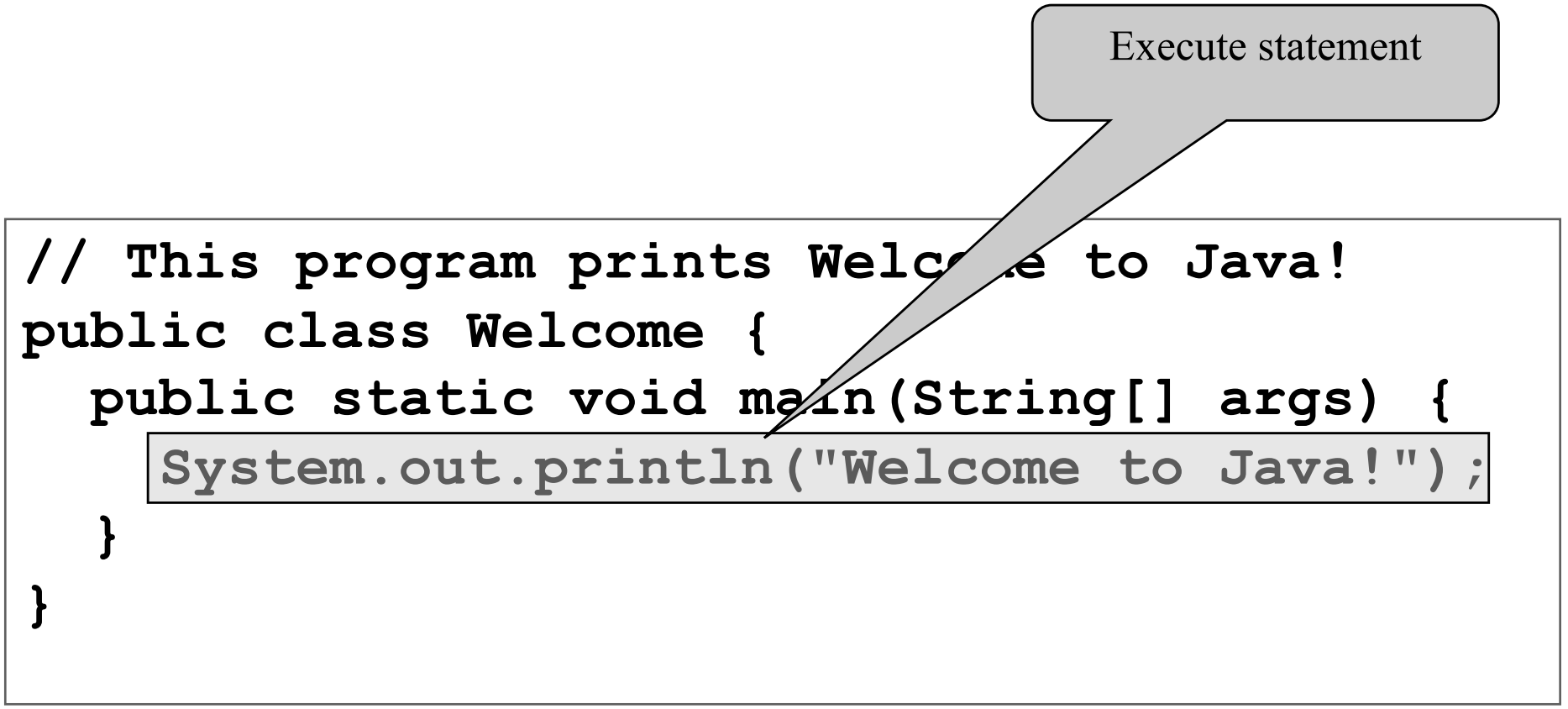
animation

Trace a Program Execution



animation

Trace a Program Execution



animation

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!");
  }
}
```

print a message to the console

```
..a_Course/ch01 (zsh) ⌘1   ..rtualMachines (z... ⌘2
➜  ch01 git:(master) ✗ subl Welcome.java
➜  ch01 git:(master) ✗ javac Welcome.java
➜  ch01 git:(master) ✗ ls -l Welcome.*
-rw-r--r--  1 huzheng  staff  429 10  7 05:26 Welcome.class
-rwxr-xr-x  1 huzheng  staff  251 10  7 05:13 Welcome.java
➜  ch01 git:(master) ✗ java -cp .. ch01.Welcome
Welcome to Java!
```

One More Simple Example

WelcomeWithThreeMessages

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.

```
// 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.

System.out.println("Welcome to Java!")

is 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!");
  }
}
```

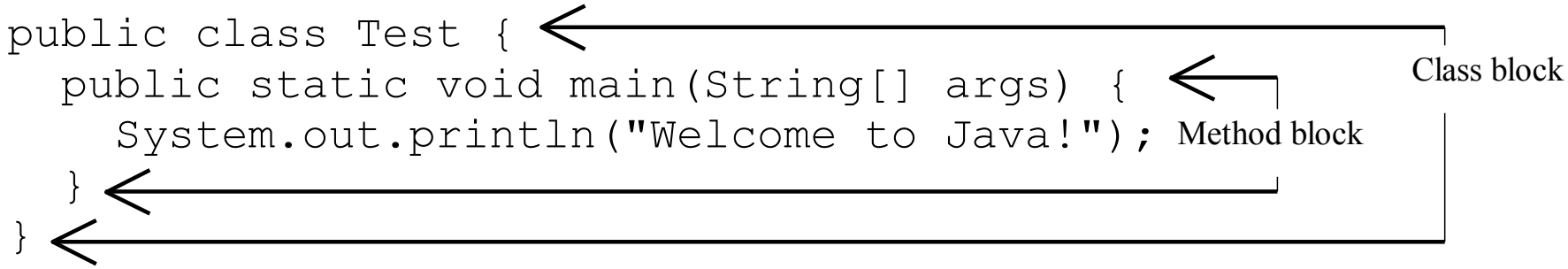
Reserved words

Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.

```
// 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 lexonents of a program.



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!");
  }
}
```

Source Code Structure

- Zero or up to one Package Statement
- Zero or multiple Import Statements
- Zero or multiple Class Declaration
- Zero or multiple Interface Declaration
- Only one Public Class or Interface
- Java source code file name must be as same as the public class name
- Up to one package declaration. The first line of java file
- Import class as specific as possible
  - import com.abc.dollapp.main.*;
  - Better: import com.abc.dollapp.main.AppMain;

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.

FileResource.java

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` `CalculatorApp`

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");
  }
}
```

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

Runtime Errors

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

ShowRuntimeErrors

Logic Errors

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

ShowLogicErrors

A bigger Example *

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