

# Java概述

计算机、程序、Java

Java历史、环境搭建、程序入门

胡铮

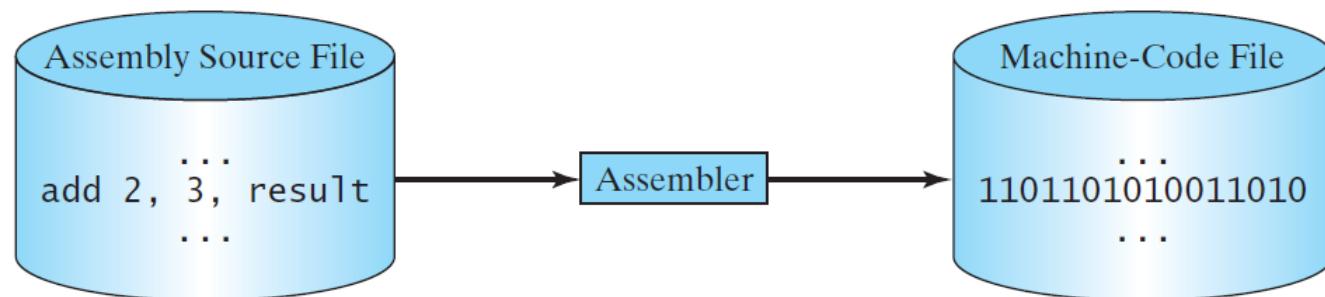
huzheng@bupt. edu. cn  
2022.08.28



# 编程语言

Machine Language    Assembly Language    High-Level Language

1101101010011010



area = 5 \* 5 \* 3.1415;

# 流行的高级编程语言

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 TRANslator. 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.

## 解释/编译 源代码

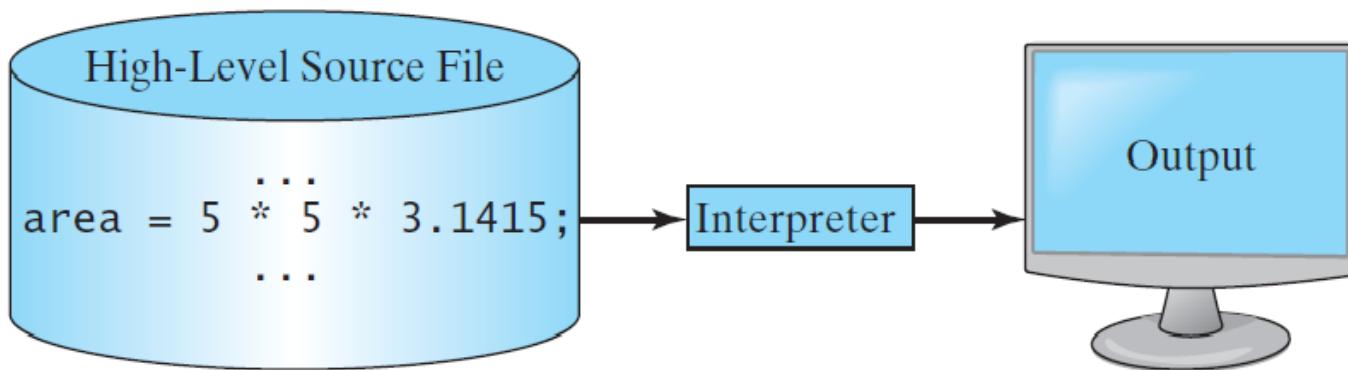
A program written in a high-level language is called a *source program* or *source code*.

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

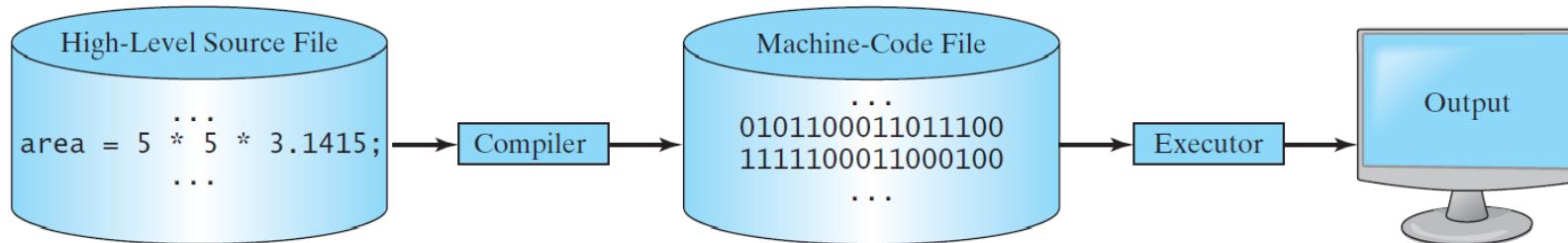
An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away.



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

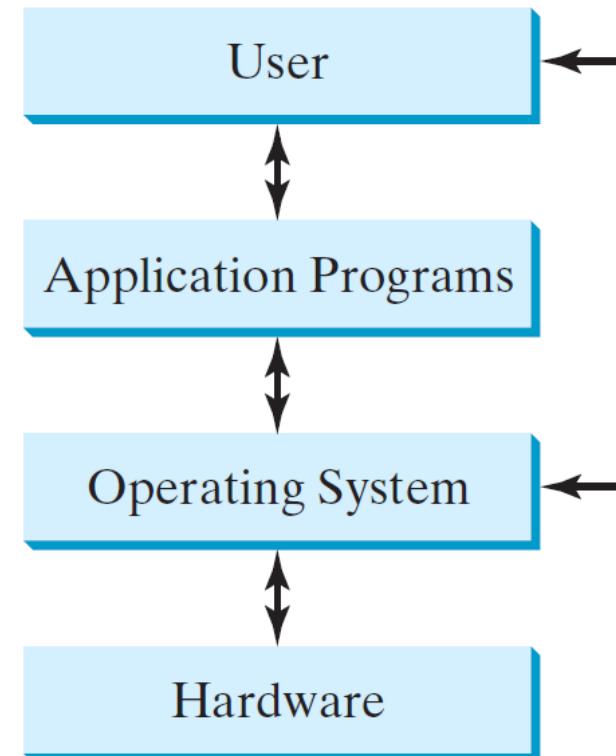
# Compiling Source Code

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



# Operating Systems

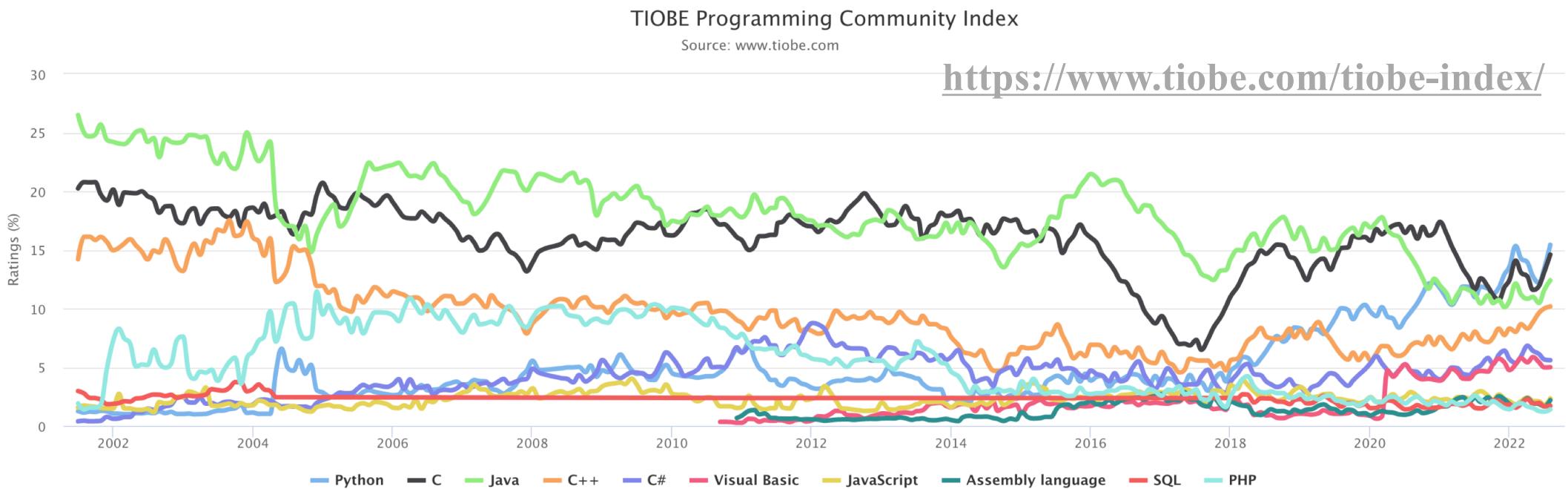
The *operating system* (OS) is a program that manages and controls a computer's activities.



# 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...



# 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 ...

# 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>

# Java特性

- ◆ 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特性

- ◆ 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.

## What does it mean

Java syntax and the Java API (application programming interface -- a library of Java code) is simple and designed to be easy to learn and use.

# Java特性

- ◆ 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.

# Java特性

- ◆ 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

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.

# Java特性

- ◆ 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).

Java source code is translated to a byte code that is interpreted one instruction at a time by the Java Virtual Machine.

# Java特性

- ◆ 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.

# Java特性

- ◆ 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 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.

# Java特性

- ◆ 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.

# Java特性

- ◆ 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.

# Java特性

- ◆ 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.

# Java特性

- ◆ 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

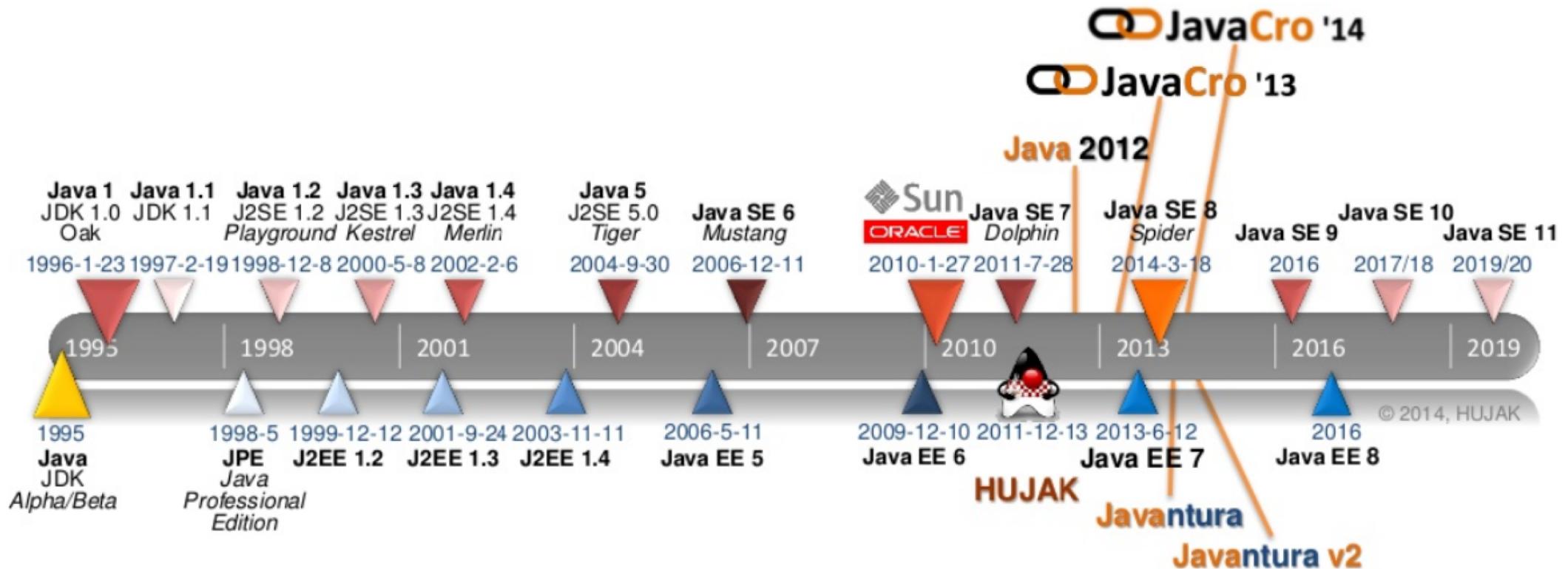
Dynamic has several meanings as it applies to the Java language. For example, the language has special capabilities that allow the program to call upon resources as needed dynamically when the program is running in memory. Such a resource can be another Java routine. The software development process 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.

# 版本一览

## ◆ JDK1.0~JDK18

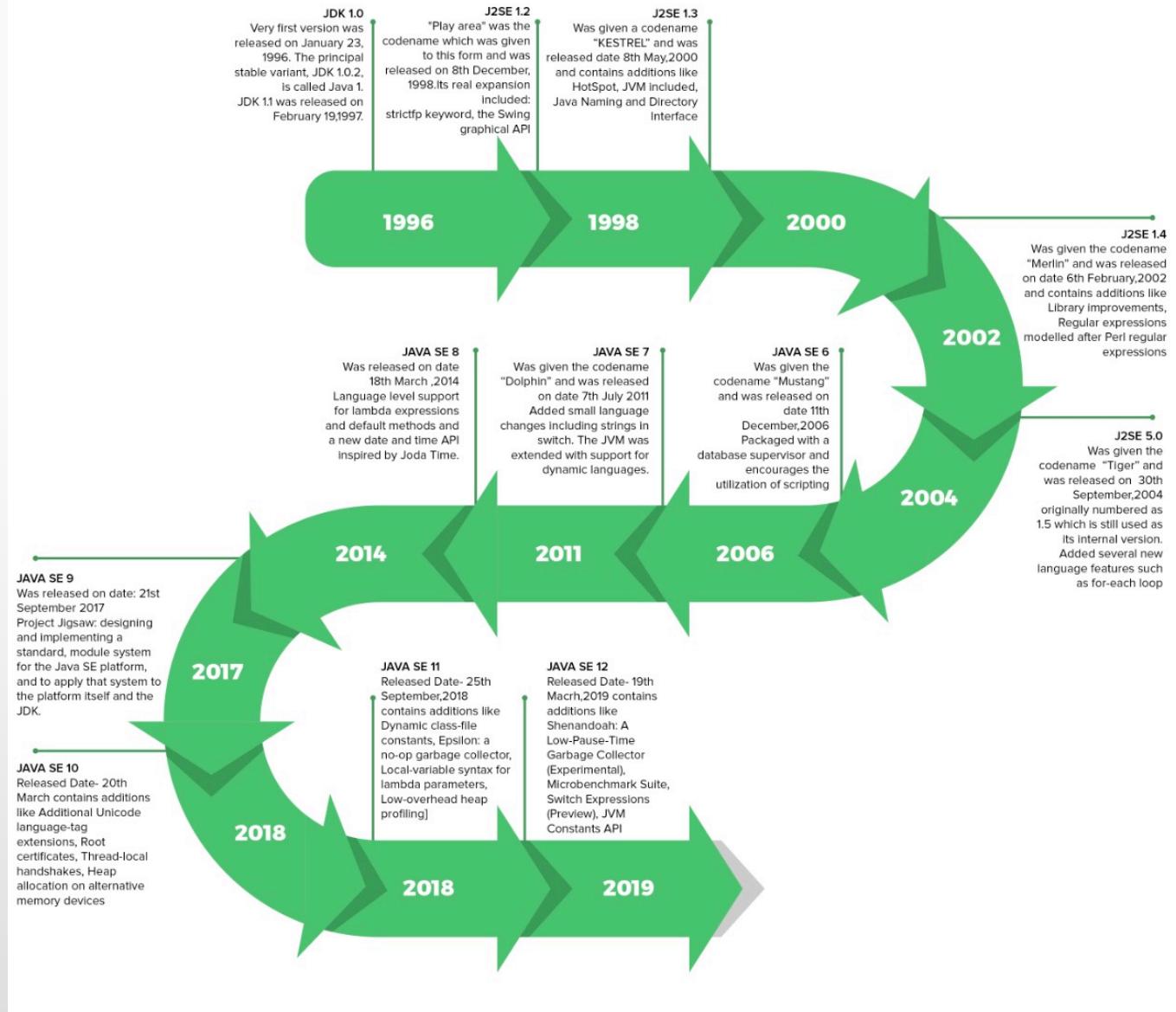
[https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)



# 版本概要

## 里程碑版本

- ◆ 1994年, Hot Java浏览器
- ◆ 1995年, Oak改名为Java
- ◆ 1996年, JDK 1.0
- ◆ 2004年, J2SE 5.0
- ◆ 2014年, JAVA SE 8, 应用最广
- ◆ 2018年, JAVA SE 11, LTS
- ◆ 2021年, JAVA SE 17, LTS, 支持至2029年, 可能应用最广



# JDK选择

Java Development Kit

- JDK与JRE
  - JRE: Java Runtime Environment
- JDK下载
  - <https://www.oracle.com/java/technologies/downloads/> 当前主流版本
  - <https://www.oracle.com/java/technologies/downloads/archive> 以往版本
  - OpenJDK是SUN公司在被Oracle收购前释出的JDK开源版本，与Oracle JDK基本相同
- Java SE/Java ME/Java EE/JavaFX



# Java API

Application Programming Interface

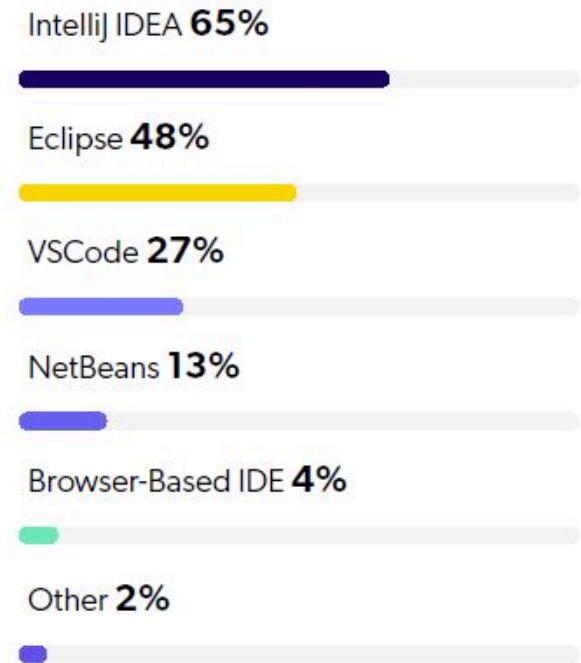
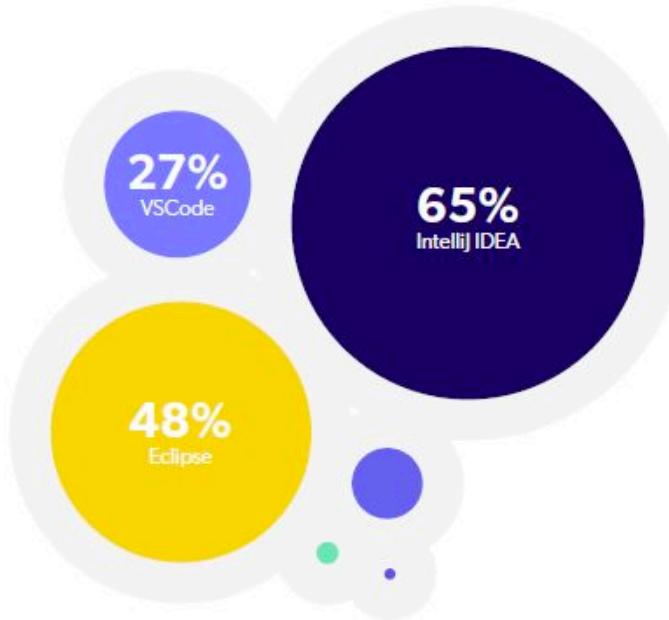
- ◆ 1) Java Core API
- ◆ 2) Java Standard Extension API (javax)
- ◆ 3) Provided by Vendors or Organizations (3<sup>rd</sup> parties)

# IDE选择

Integrated Development Environment

- 集成开发环境
- 推荐IDE
  - IntelliJ IDEA
  - Visual Studio Code
- 其他
  - Eclipse
  - NetBeans

What developer IDE do you use professionally?



# IntelliJ IDEA

[jetbrains.com](https://jetbrains.com)



- 下载链接
  - <https://www.jetbrains.com/idea/download/>
- 免费license申请
  - <https://www.jetbrains.com/community/education/#students>
- license适用于Jetbrains公司所有产品的Ultimate版本，1年后续即可

	IntelliJ IDEA Ultimate	IntelliJ IDEA Community Edition
Java, Kotlin, Groovy, Scala	✓	✓
Maven, Gradle, sbt	✓	✓
Git, GitHub, SVN, Mercurial, Perforce	✓	✓
Debugger	✓	✓
Docker	✓	✓
Profiling tools <small>i</small>	✓	
Spring, Jakarta EE, Java EE, Micronaut, Quarkus, Helidon, and more <small>i</small>	✓	
HTTP Client	✓	
JavaScript, TypeScript, HTML, CSS, Node.js, Angular, React, Vue.js	✓	
Database Tools, SQL	✓	
Remote Development (Beta)	✓	
Collaborative development	✓	✓

# 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



# Creating and Editing Using NotePad

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

```
..b/Java_Course (-zsh) ⌘1 ..b/Java_Course (-zsh)
(base) → Java_Course git:(master) ✘ pwd
/Users/zhenghu/Documents/Github/Java_Course
(base) → Java_Course git:(master) ✘ subl ch01/Welcome.java
(base) → Java_Course git:(master) ✘
```

```
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!"); //to do
9     }
10}
```

```

Welcome.java  x
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!"); //to do
9     }
10 }

```

**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)**

```

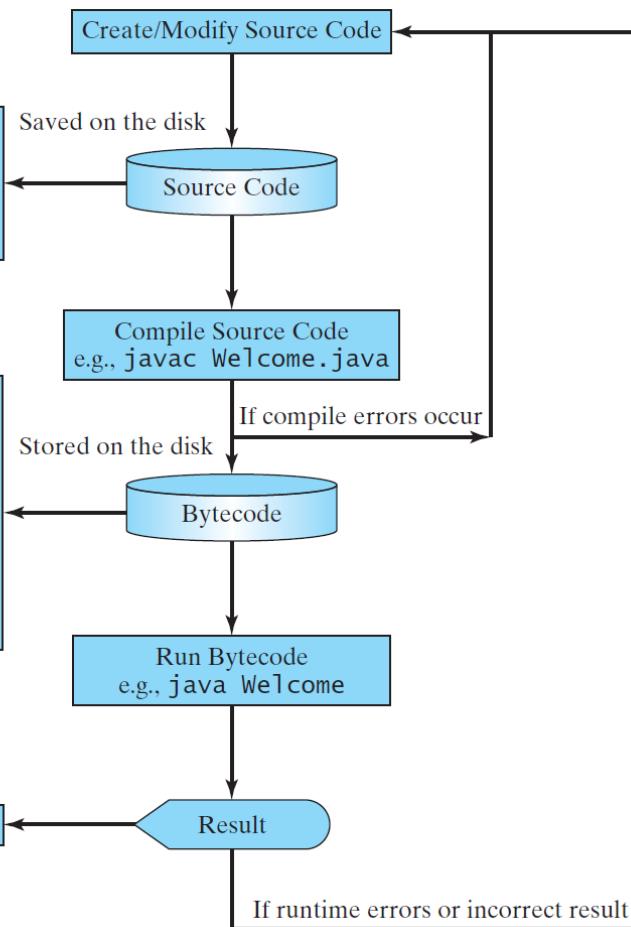
...
Method Welcome()
0  aload_0
...
Method void main(java.lang.String[])
0  getstatic #2 ...
3  ldc #3 <String "Welcome to Java!">
5  invokevirtual #4 ...
8  return

```

**“Welcome to Java” is displayed on the console**

Welcome to Java!

# Creating, Compiling, and Running Programs

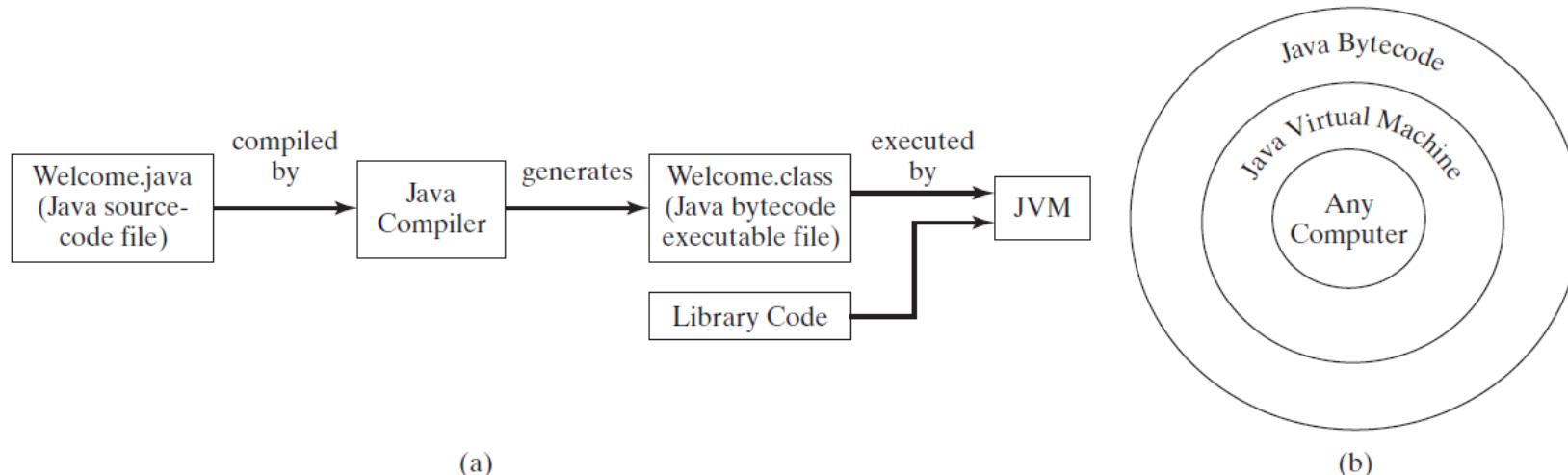


# Compiling Java Source Code

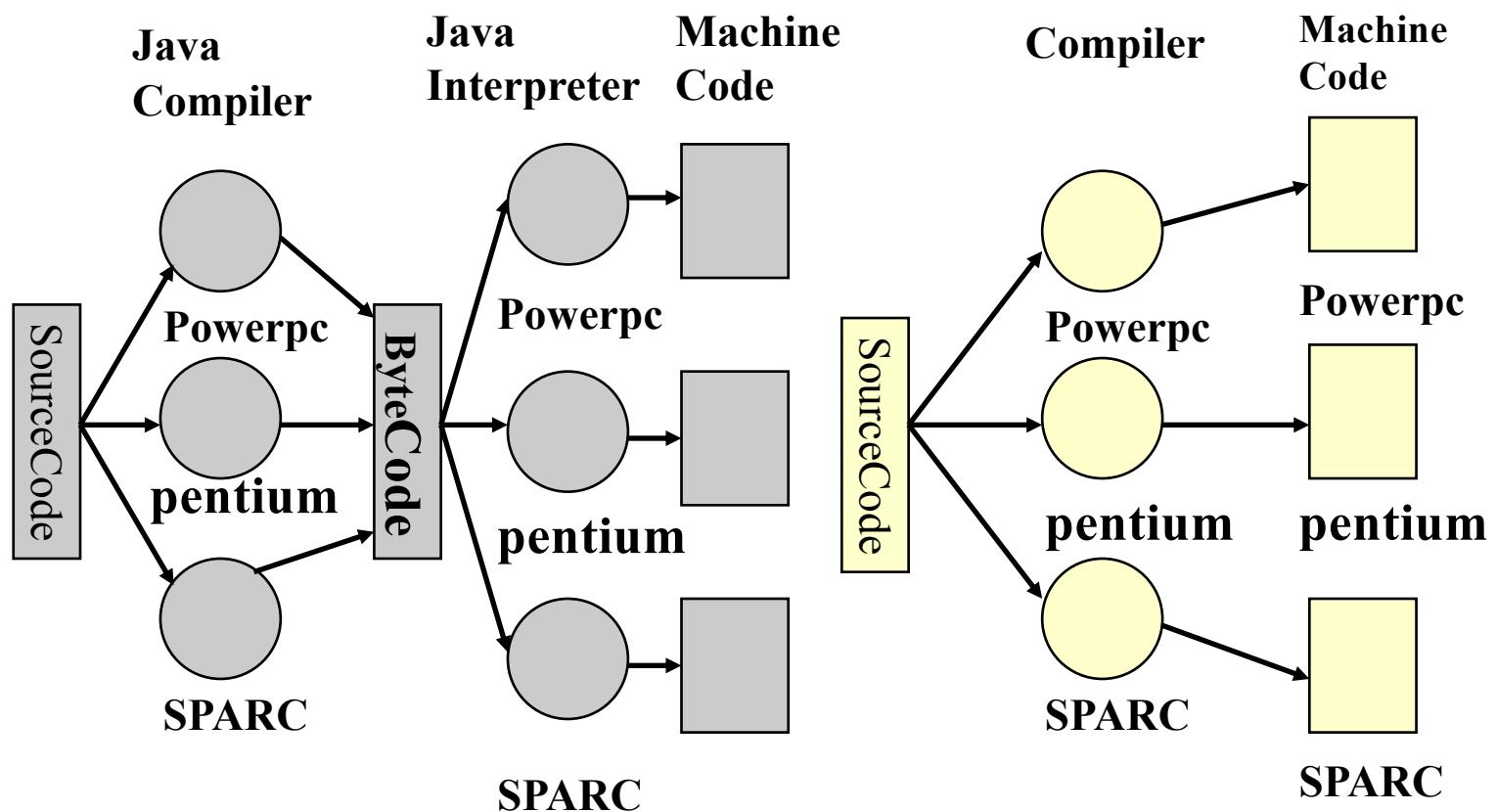
Write the program once, and compile the source program into a special type of object code, known as *bytecode*.

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

Java Virtual Machine is a software that interprets Java bytecode.



# Neutral/Platform Independent



animation

# 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!");
    }
}
```

animation

# 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!");
    }
}
```

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!");
    }
}
```

```
(base) → Java_Course git:(master) ✘ javac ch01/Welcome.java
(base) → Java_Course git:(master) ✘ ll ch01/Welcome.class
-rw-r--r-- 1 zhenghu staff 429 9 15 21:16 ch01/Welcome.class
(base) → Java_Course git:(master) ✘ java welcome
错误：找不到或无法加载主类 welcome
原因：java.lang.ClassNotFoundException: welcome
(base) → Java_Course git:(master) ✘ java ch01>Welcome.class
错误：找不到或无法加载主类 ch01>Welcome.class
原因：java.lang.ClassNotFoundException: ch01>Welcome.class
(base) → Java_Course git:(master) ✘ java ch01>Welcome
Welcome to Java!
```

print a message to the  
console

# 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.

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

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!");
    }
}
```

# 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);  
    }  
}
```