CS109: Introduction to Computer Programming

Yepang Liu (刘烨庞)

liuyp1@sustech.edu.cn

Lecture Instructor

- Dr. Yepang LIU (Associate Professor in CSE)
- Office: Room 609, CoE Building (South)
- Email: liuyp1@sustech.edu.cn
- Homepage: https://yepangliu.github.io/
- Office hour: Monday 3:00 PM – 4:00 PM

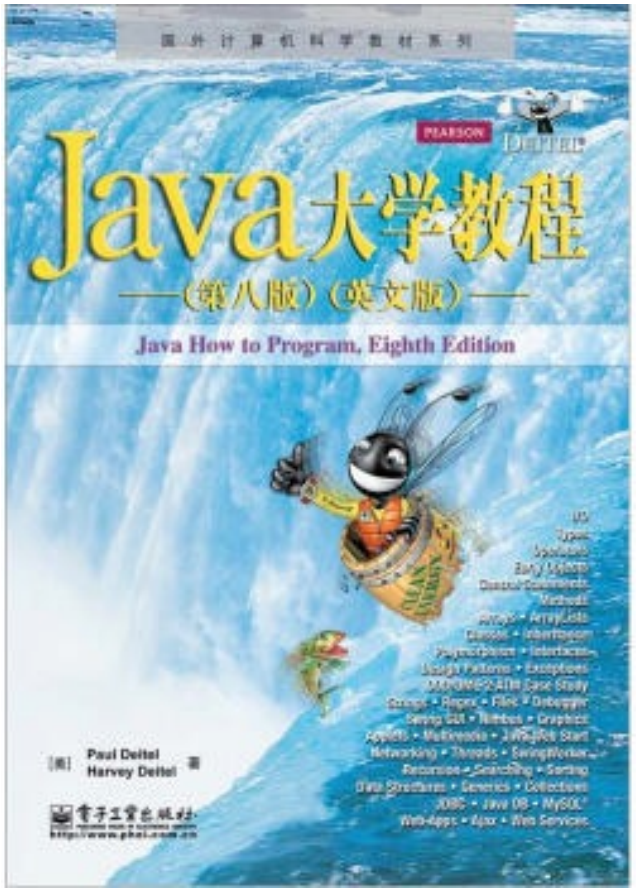
  (you can drop by my office; no appointment is needed)

Course Website & WeChat Group

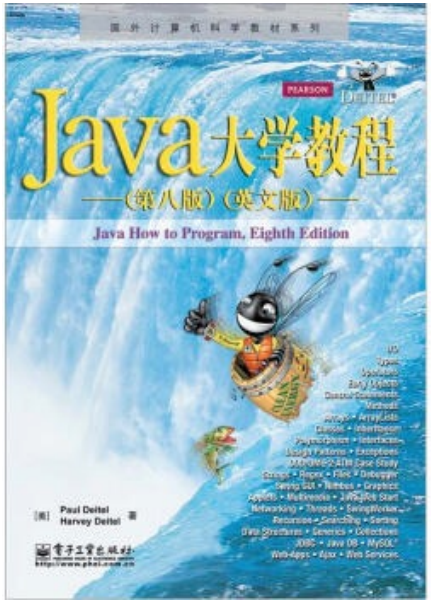
- Course materials will be available at the **Blackboard course site**: https://bb.sustech.edu.cn/

Textbook

- Main textbook:
  - P. Deitel, H. Deitel, **Java: How to Program（Java大学教程，第八版），电子工业出版社**
- Reference books:
  - Y. Daniel Liang. **Introduction to Java Programming**, 12e, Pearson, Prentice Hall, 2020.
  - Allen B. Downey and Chris Mayfield. **Think Java, How to Think Like a Computer Scientist**, O'Reilly, 2016.



Course Syllabus



- Introduction to Computers and Java Applications
- Primitive Data Types
- Control Statements and Structured Programming
- Array
- Procedural Programming: Methods and APIs
- Introduction to Classes, Objects, Methods
- Strings and Wrapper Classes
- Classes, Objects and Methods: A Deeper Look
- Object-Oriented Programming: Inheritance
- Object-Oriented Programming: Polymorphism
- Graphical User Interface (GUI)
- Generic Classes and Methods
- Exception Handling: A Deeper Look

Grading Scheme

- Final exam: 40%
- Project: 20%
- Lab participation and exercises: 5% (14 weeks)
- Programming assignments: 30%
  - 6 assignments, starting from week 2
- Lecture participation and quizzes: 5%

Programming!

You will pass the course if your overall grade >= 60

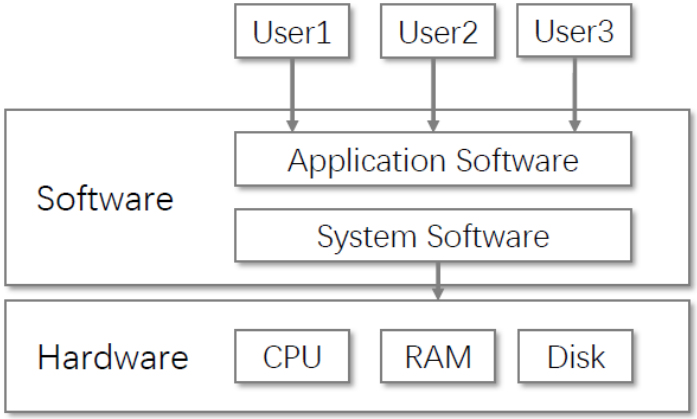
Let's Start & Have fun ☺

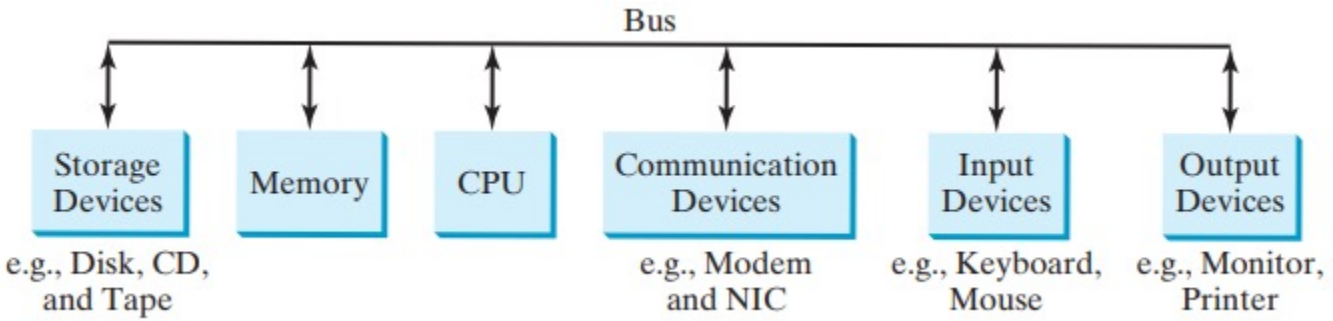
Practice Makes Perfect!

Chapter 0:
Introduction to Computers, Programs, and Java

What is a computer?

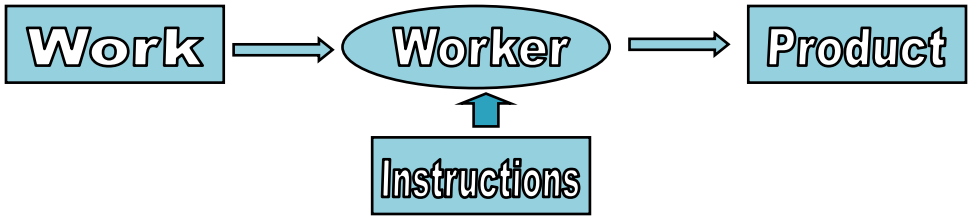
- **Software**: a set of programs, which could be viewed as a set of instructions
- **Hardware**: physical parts (e.g., keyboard, mouse, hard disk, memory, CPU). Hardware is directed by software to execute commands or instructions



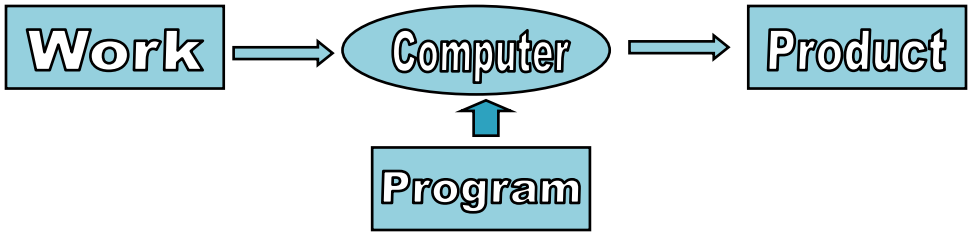


What is a computer program?

- Human work model

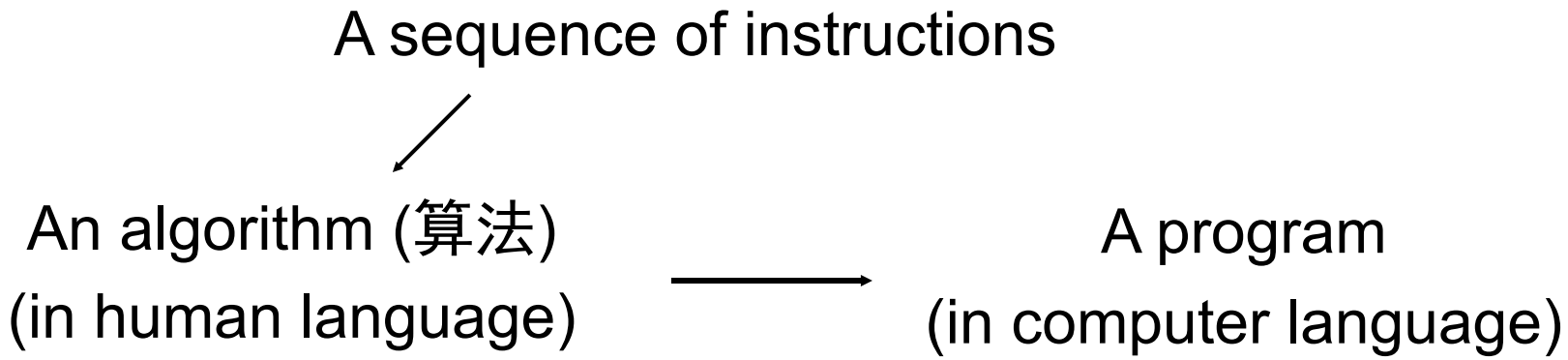


- Computer work model



- A computer program is a set of machine-readable instructions that tells a computer how to preform a specific task.

What is a (programming) language?



- Programs are written in programming languages
- There are many programming languages
  - Low-level (低级语言), understandable by a computer
  - High-level (高级语言), understandable by human

Can you understand this?

0000100100101110011001100110100101101100011001010000100100100010011011000110
0101011000110111010001110101011100100110010100110001001011100110001100100010
0000101001100111011000110110001100110010010111110110001101101111011011010110
0000110100101101100011001010110010000101110001110100000101000101110011100110
1100101011000110111010001101001011011110110111000001001001000100010111001110
1000110010101111000011101000010001000001010000010010010111001100001011011000
1101001011001110111011100010000000110100000010100000100100101110011001111011 01
1000110111101100010011000010110110000100000011011010110000010110100101101110 0
0001010000010010010111001110100011110010111000001100101000010010010000001101
1010110000101101001011011100010110000100011011001100111010101101110011000110
1110100011010010110111101101110000010100000100100101110011100000111001001101 1
1011000110000100100110000001101000000101001101101011000010110100101101110000
1110100000101000001001001000010010001101010000010100100100111101001100010010
1110100011101010101010001010010001100100000001100000000101000001001011100 11
0110000101110110 01

How about this?

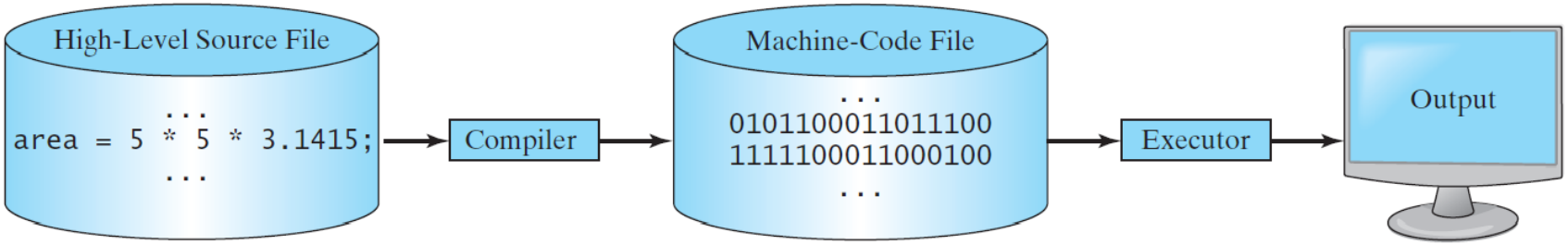
```
main:
        !#PROLOGUE# 0
        save %sp,-128,%sp
        !#PROLOGUE# 1
        mov 1,%o0
        st %o0,[%fp-20]
        mov 2,%o0
        st %o0,[%fp-24]
        ld [%fp-20],%o0
        ld [%fp-24],%o1
        add %o0,%o1,%o0
        st %o0,[%fp-28]
        mov 0,%i0
        nop
```

Is it better now?

```
int valueofz( )
{
        int x, y, z;
        x = 1;
        y = 2;
        z = x+y;
        return z;
}
```

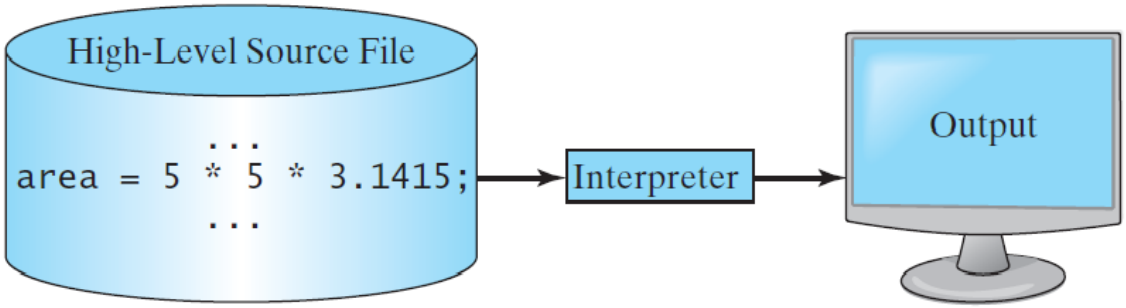
Compilation: from source to executables

- A complier (编译器) translates source programs written in high-level languages into machine codes that can run directly on the target computer.



Interpreter

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



A brief history of Java

- In 1991, Sun Microsystems (acquired by Oracle in 2009) funded an internal research project, aiming to achieve the goal of "write once, run anywhere". This resulted in a C++-based language named Java.
- Why called "Java"? Java is an island in Indonesia where the first coffee was produced (Java coffee) and "programmers drink a lot of java" (by Jim Waldo, a Harvard computer scientist who worked at Sun Microsystems)

The father of Java:
James Gosling

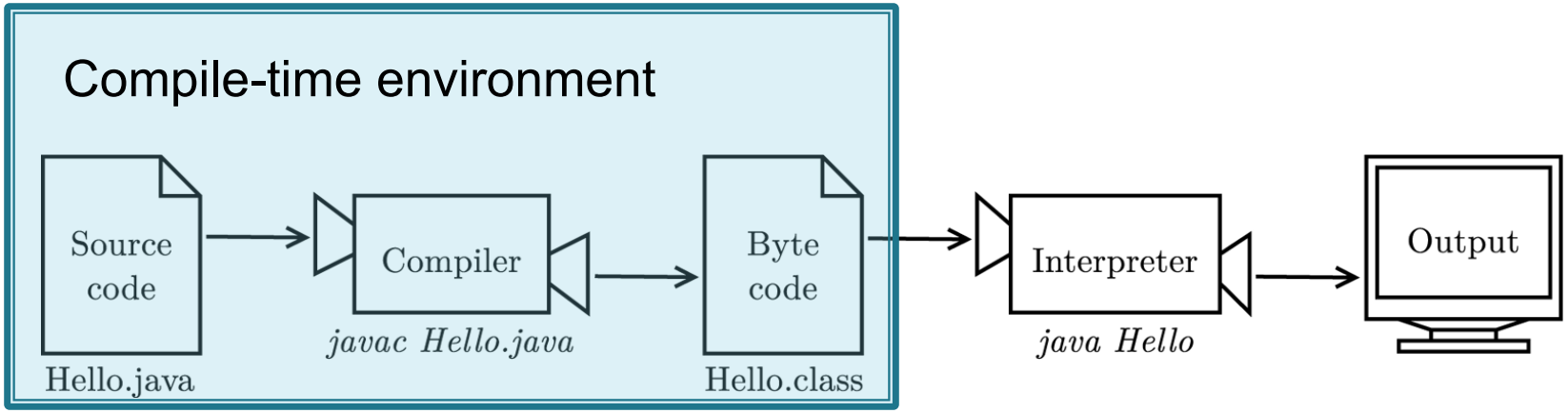
We learn Java, why?

- Java is a full-featured, general-purpose programming language.
- Can be used to develop applications across platforms on servers, desktop computers, and mobile devices.

https://www.oracle.com/a/ocom/docs/java-strength-in-numbers.pdf

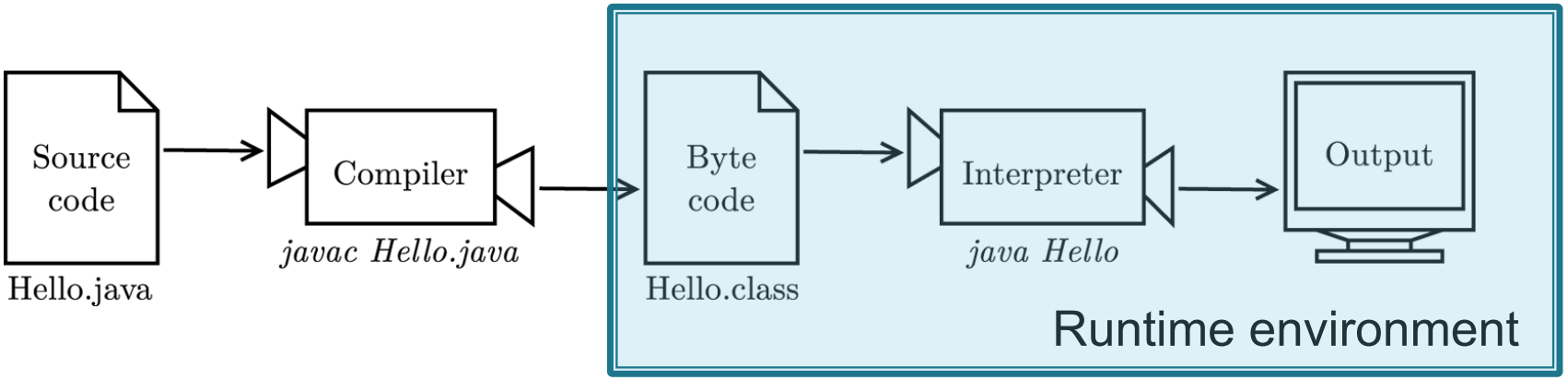
Java programming steps

- Step 1: Edit (write the program and store it in the disk `.java`)
- Step 2: Compile (create bytecode and store it in a file `.class`)

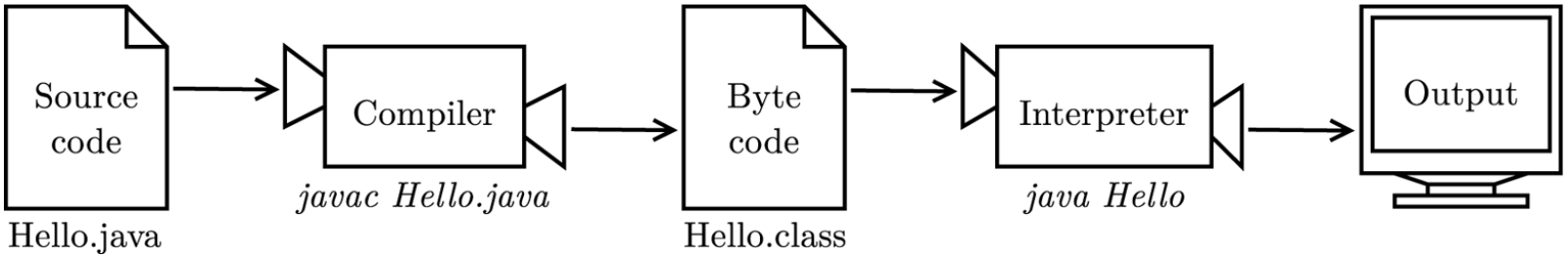


Java programming steps

- Step 3: the .class bytecode is read, verified, interpreted, and executed in JVM (Java Virtual Machine)

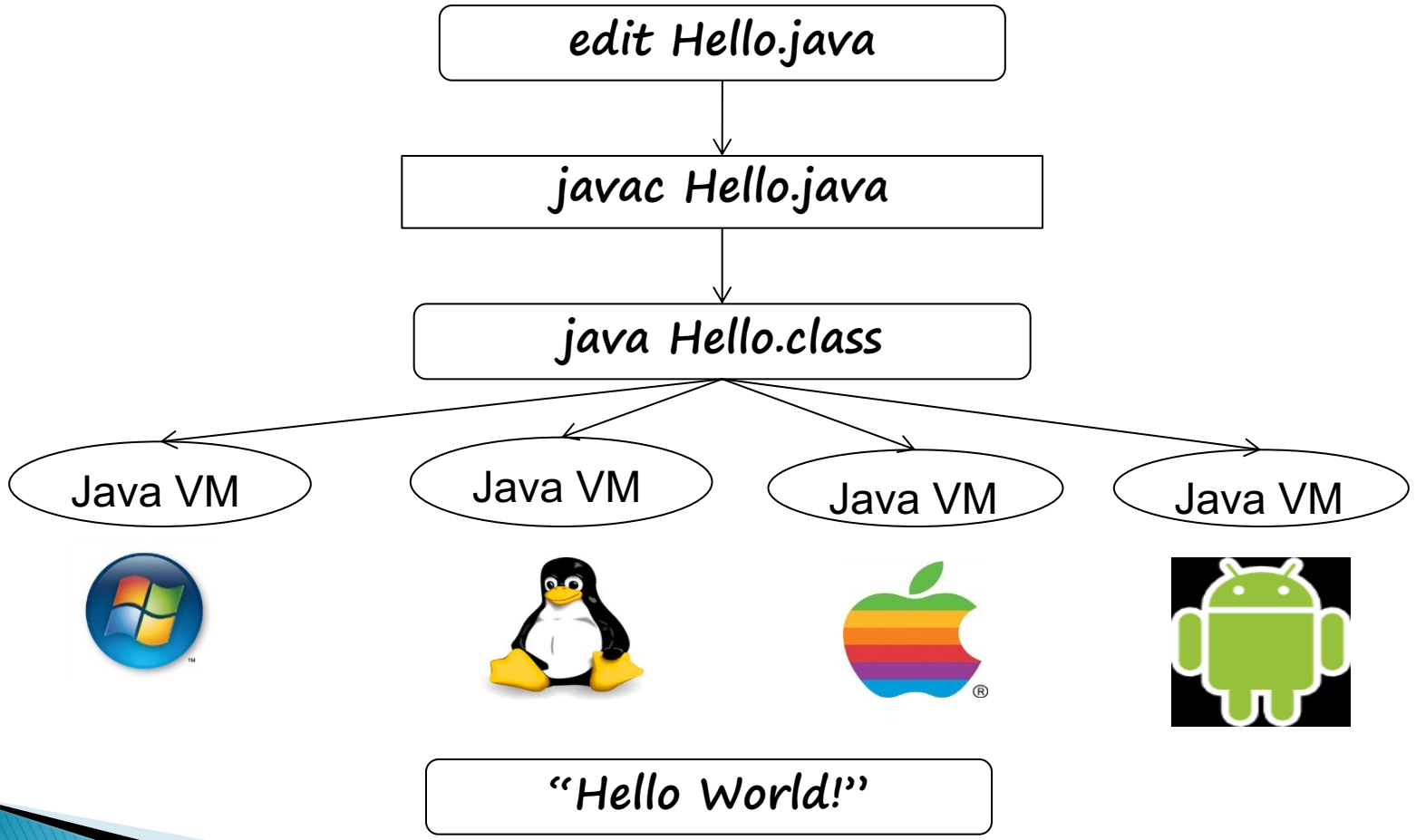


Java is both compiled and interpreted



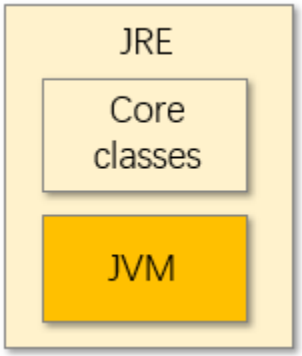
Write Once and Run Anywhere

Java is platform independent



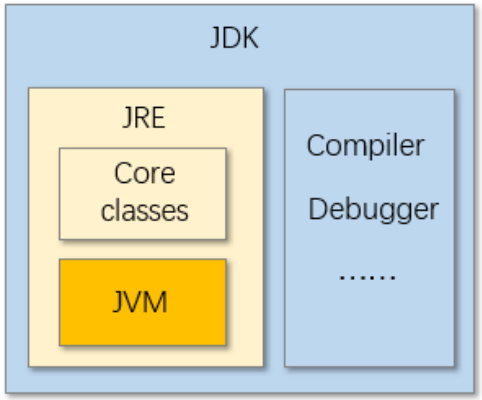
JRE and JVM

- A **Java Virtual Machine (JVM)** is an abstract computing machine that enables a computer to run a Java program.
- The **Java Runtime Environment (JRE)** provides the minimum requirements for executing a Java application. It consists of the Java Virtual Machine (JVM), core classes, and supporting files.
- In short, JRE = JVM + Library classes



JDK（开发套件）

- The **Java Development Kit (JDK)** is a software development environment for developing Java programs. It includes:
  - A Java Runtime Environment (JRE, 运行环境)
  - A compiler (javac)
  - An interpreter/loader (java)
  - An archiver (jar)
  - A documentation generator (javadoc)
  - Other tools needed in Java development.



- In short, JDK = JRE + Development tools

Our First Java Program

```
public class Welcome1 {
    // main method begins the execution of a Java application
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

`Welcome1` prints the following text in the command window (console):

```
Welcome to Java Programming!
```

Class Declaration

```
public class Welcome1
```

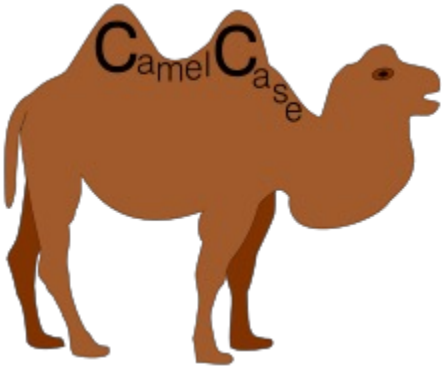
- Every Java program consists of at least one class (类) that you define
- The class keyword introduces a class declaration and is immediately followed by the class name
- Keywords are reserved for use by Java and are always spelled with all lowercase letters (we will see more later)

```
public class Welcome1 {
    // main method begins the execution of a Java application
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

Identifiers (标识符) `public class Welcome1`

- A name in a Java program is called an **identifier**, which is used for identification purpose.
  - "`Welcome1`" is an identifier. It is the name for the class we just defined.
- The only allowed characters in Java identifiers are **a to z, A to Z, 0 to 9, $** and **_** (underscore).
- Identifiers can't start with digits, e.g., **123name** is invalid. Java Keywords cannot be identifiers (can't compile).

Class Names



- By convention, class names begin with a capital letter and capitalize the first letter of each word they include (upper camel case, 大驼峰式命名规范)

- Java is case sensitive—uppercase and lowercase letters are distinct (not in comments). "`main`" and "`Main`" are different identifiers.

Comments (注释)

```java
public class Welcome1 {
    // main method begins the execution of a Java application
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

```
// This is a line comment (行注释)

/* This is a block comment (块注释或段注释). It
   can be spread over multiple lines */
```

- Comments help document programs to improve their readability.
- Compiler ignores comments.

Traditional vs. End-of-Line Comments

- Traditional comments do not nest (嵌套), the first */ after the first /* will terminate the comment

```
/*
  /* comment 1 */
     comment 2 */
```

Syntax Error (语法错误)

- End-of-line comments can contain anything

```
// /* this comment is okay */
```

Method declaration

- Java class declarations normally contain one or more methods
  - A method is a block of code, which only runs when it is called; They are used to perform certain actions; In some languages, such blocks are called functions.
- The main method is the starting point of Java applications

```
public class Welcome1 {
    // main method begins the execution of a Java application
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

Braces (花括号)

- A pair of curly braces { } in a program forms a block (块) that groups the program's components.
- A left brace { begins the declaration of every class and method
- A corresponding right brace } ends the declaration of each class and method

```
public class Welcome1 {
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

The `main` method body

- A method is a construct that contains statements (语句).
- The System.out.println statement displays the input string on the console
- String is a programming term meaning a sequence of characters. A string must be enclosed in double quotation marks (“ xxx ”).
- Every statement in Java ends with a semicolon (;),

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

The System.out Object

```
System.out.println("Welcome to Java Programming!");
```

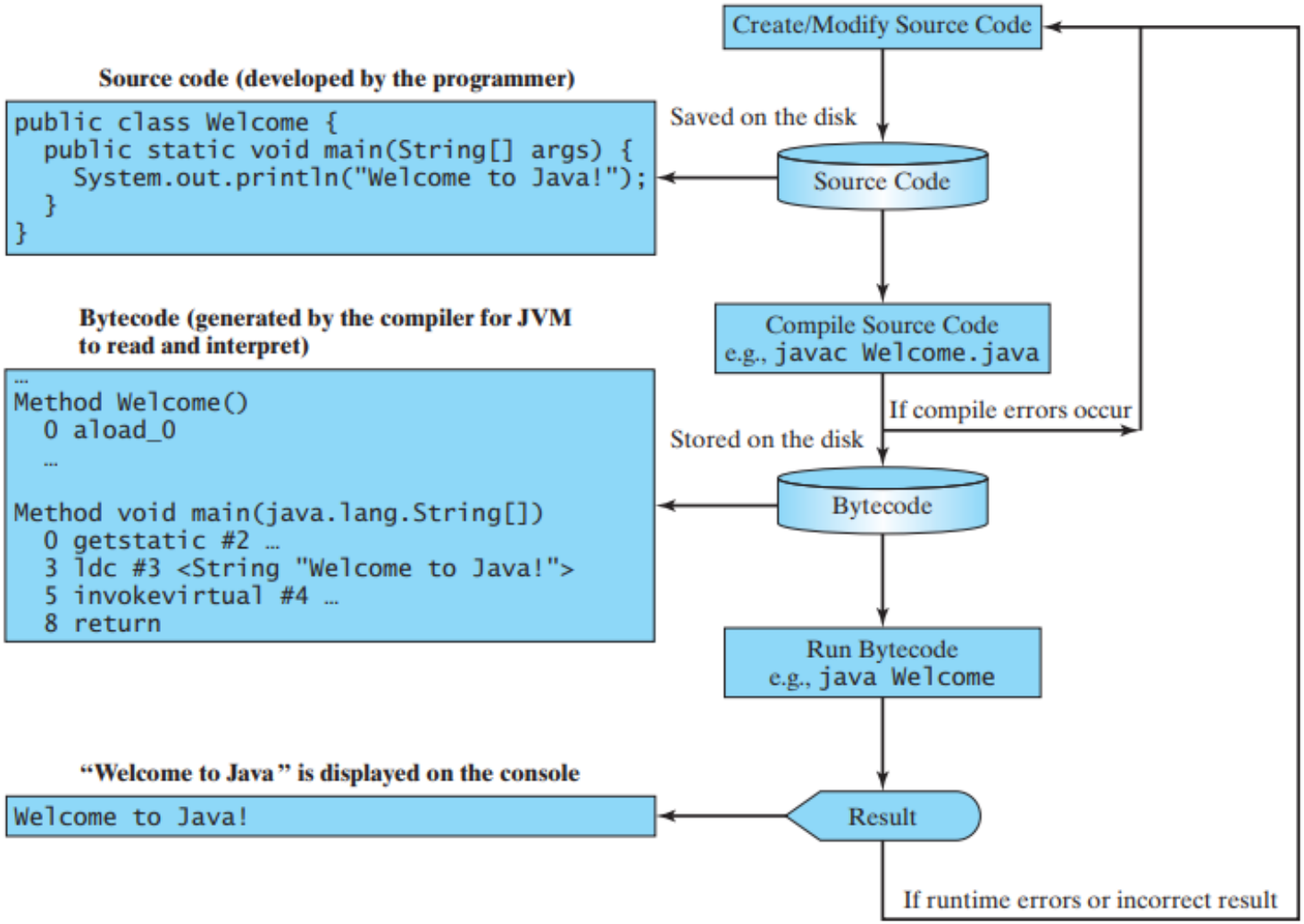
- System.out is the standard output object that allows Java applications to display strings in the command window
- `System.out.println` method `public void println(String x)`
  - Displays (or prints) a line of text in the command window
  - The string in the parentheses is the actual argument (实际参数) to the method
  - Positions the output cursor at the beginning of the next line in the command window

Indentation (缩进)

- Code between braces should be indented (good practice to make program structure clear)
- Indentation doesn't affect the compilation and execution of code in Java
- But for some programming languages, indentation matters a lot (e.g., Python uses indentation to indicate a block of code, all the statements with the same space to the right belong to the same code block)

```
public class Welcome1 {
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming!");
    }
}
```

Compile & Execute `Welcome1.java`



Modifying Welcome1.java

```
// Print a line of text with multiple statements
public class Welcome2 {
    public static void main(String[] args) {
        System.out.print("Welcome to ");
        System.out.print("Java Programming!");
    }
}
```

Class Welcome2 uses two statements to produce the same output as class Welcome1

```
Welcome to Java Programming!
```

The System.out.print() method

- System.out's method print displays a string

- Unlike the method println, print does not position the output cursor at the beginning of the next line in the command window (it simply prints the string)

```
System.out.print("Welcome to ");
System.out.print("Java Programming!");
```

Modify the code

```
// Print multiple lines of text using a single statement
public class Welcome3 {
    public static void main(String[] args) {
        System.out.println("Welcome\nto\nJava\nProgramming!");
    }
}
```

`Welcome3` prints the following text on the console:

```
Welcome
to
Java
Programming!
```

The newline character \n

- Newline characters (换行符) instruct `System.out`'s `println` method to position the output cursor at the beginning of the next line in the command window

- Newline characters are white-space characters, which represent horizontal or vertical space in typography and do not correspond to visible marks (辅助排版)

```
System.out.println("Welcome\nto\nJava\nProgramming!");
```

Escape character

- The backslash (\) is an escape character (转义字符, a case of metacharacters), which invokes an alternative interpretation on subsequent characters (转换意义)

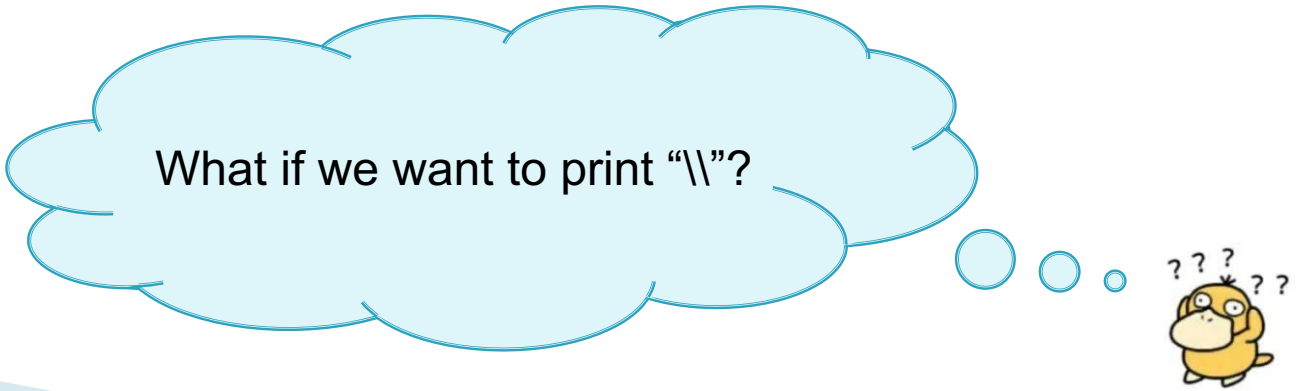
- Backslash \ is combined with the next character to form an escape sequence (转义序列)

- The escape sequence \n represents the newline character

Common Escape Sequences

Sequence	Description
`\n`	Newline (换行符). Position the cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the cursor to the next tab stop.
`\"`	Used to print a double-quote character. `System.out.println("\"in quotes\"");` displays `"in quotes"`

More at: https://docs.oracle.com/javase/tutorial/java/data/characters.html



What is debugging?

- The process of tracking down and correcting bugs in your programs
  - Compile-time Errors（编译错误）: Syntactical problems due to incorrect use of Java syntax, which can be caught by compilers (e.g., missing a semicolon at the end of a statement). Java programs cannot be executed before fixing compile-time errors. => most easy bugs
  - Runtime Errors (运行时错误, 异常): Runtime errors occur when the byte code is running in JVM, but something goes wrong and cannot be fixed by the interpreter, e.g., division by zero causes a ArithmeticException. Programs will be abruptly terminated when runtime errors occur.
  - Logical Errors (逻辑错误): Logical errors occur when the programmer uses incorrect logic or formula during coding (the program runs but yields an unexpected result). => most critical and difficult bugs