Tour of Java

科协智能体部 计06 徐晨曦

PREFACE

Introduction to Java Platform

What is Java

- Java是一门*高层次，强类型，编译型，基于类的面向对象*的*通用*编程语言
- Java保证*一次编写，到处运行 (write once, run anywhere)*，即Java编译出的字节码文件可以保证在任何系统上的Java运行时环境运行
- Java保证向前兼容性
- Java是一门*类C语言*
- 从1995年开始，历史悠久的Java平台积累了兼具广度和可靠性的生态系统和开发者社区
- Java至今仍然是最流行、最知名、使用最广的编程语言之一

History of Java

- 1991, *James Gosling*, C++ and Oak
- 1995, *Sun Microsystems* and Java 1.0
- 1996, Dynamic Web & Java Applets
- 2006, Java EE, Java SE, Java ME
- 2007, Java visual machine is opened under GPL
- 2009, Sunset of the *Sun*
- 2014, Java 8 reached General Availability
- 2018, Java 10 started to adopt Time-Based Release Versioning
- Today, Java 17 is current LTS, Java 18 GA, Java 19 RDP 2

Why Java

- Easy!
- Cross-Platform
- Rich and Reliable Ecosystem
- Forward evolution with backwards compatibility
- Performance

Why NOT Java

- Performance ?
- 相对贫弱的语法特性削弱了对程序员的吸引力
- JavaScript, Python等语言的流行和其生态系统的逐步完善
- Go, Rust等新星语言的涌现
- Kotlin, Scala, Clojure, Groovy等JVM平台语言提供了利用Java生态系统的其他选项
  - Kotlin获得了Google的青睐和Android平台的优先支持
  - Scala极为灵活的语法和强大的表达能力长期以来收到科学计算领域和分布式数据处理领域的欢迎

Before we start,

Let's recognize some terms of Java

- JDK, Java Develop Kit, Java开发工具包
- JRE, Java Runtime Environment, Java运行时环境
- JVM, Java Virtual Machine, Java虚拟机
- OpenJDK, 开源Java平台实现的合作组织
- LTS，Long Term Support, 长期支持版本
- IDE, Integrated Development Environment, 集成开发环境

Chapter 0

Installation of JDK & IntelliJ IDEA

Eclipse Adoptium

- **TLDR**: Eclipse Adoptium是一个提供一大堆平台和一大堆不同版本的JDK的网站
- **太长不看版**
  - OpenJDK顾名思义，是开源的，因此很多公司和组织都提供了自己的构建版本
  - 2017年启动的AdoptOpenJDK项目是第一个提供一大堆平台和一大堆不同版本的JDK的网站，同时也提供了多种平台的构建和测试平台
  - Eclipse Adoptium是AdoptOpenJDK的继任者
  - 它们提供的OpenJDK分发版称为Eclipse Temurin

Install JDK *(Eclipse Temurin)*

- 进网站，点下载，双击安装包
- macOS & Homebrew

```
$ brew install --cask temurin       // Install latest JDK18
$ brew tap homebrew/cask-versions
$ brew install --cask temurin17
```

- Linux

```
$ apt install temurin-17-jdk
$ yum install temurin-17-jdk
$ zypper install temurin-17-jdk
```

Other OpenJDK Distribution ?

- Azul Zulu
- BellSoft Liberica JDK
  - 上面两个在业界均有一定应用率，可以信赖
  - 均提供JDK8在ARM Mac的构建版本
  - 均提供JavaFX的支持
- GraalVM
  - Oracle开发的下一代JDK和全新的多语言虚拟机
  - 将Java编译为本地代码以消除大部分根本上的性能问题
  - 提供高性能的Python, R, Ruby等语言的运行时

First Command

```
$ java --version
openjdk 17.0.3 2022-04-19
OpenJDK Runtime Environment Temurin-17.0.3+7 (build 17.0.3+7)
OpenJDK 64-Bit Server VM Temurin-17.0.3+7 (build 17.0.3+7, mixed mode)
```

CodeLab 0-1: helloworld

Wait... What is *CodeLab*?

- 编程小练习，仅此而已
- 我会带大家在课上都写一遍
- GitHub仓库里提供了未完成供大家联系，也同时提供了完成的版本供大家参考
- `tourofjava/chN/labM`

CodeLab 0-1: helloworld

`Helloworld.java`

```java
package tourofjava.ch0.lab1;

public class Helloworld {
    public static void main(String[] args) {
        System.out.println("helloworld");
    }
}
```

```
tour-of-java/codelab $ javac tourofjava/ch0/lab1/Helloworld.java
tour-of-java/codelab $ java tourofjava.ch0.lab1.Helloworld
helloworld
```

CodeLab 0-1 helloworld

[JEP330, Java11] Launch Single-File Source-Code Programs

```
tour-of-java/codelab $ java tourofjava/ch0/lab1/Helloworld.java
helloworld
```

CodeLab 0-1 helloworld

[JEP222, Java9] `jshell`: The Java Shell (Read-Eval-Print Loop)

```
$ jshell
jshell
|  欢迎使用 JShell -- 版本 17.0.3
|  要大致了解该版本, 请键入: /help intro

jshell> System.out.println("helloworld")
helloworld
```

JetBrains IntelliJ IDEA

- https://www.jetbrains.com/idea/
- 目前最强大、使用最广泛的Java IDE
- IntelliJ IDEA Community Edition是免费开源版本，其包含了基础的Java语言支持、分析、构建与版本管理和Docker支持
- IntelliJ IDEA Ultimate是收费版本，支持性能剖析，企业级框架支持，Web技术支持，数据库工具，远程协作开发支持
  - 使用 `@mails.thu.edu.cn` 可以申请JetBrains学生包以获取全部开发工具

Chapter 1

Basic language structure

Java is a *C-style* language

Comments

```
// This is a one-line comment

/*
 * This is a multi-line comment
 */

/**
 * This is JavaDoc
 */
```

Java is a *C-style* language

Variable Declare & Assignment / Data types

```
jshell> int a;
a ==> 0
jshell> a = 1;
a ==> 1
jshell> long b = 1L
b ==> 1
jshell> char e = 'E'
e ==> 'E'
jshell> String f = "java"
f ==> "java"
jshell> boolean g = true
g ==> true
```

Java is a *C-style* language

Variable Declare & Assignment / Data types

```
jshell> float c = 1.0
|  错误:
|  不兼容的类型: 从double转换到float可能会有损失
|  float c = 1.0;
|            ^-^
jshell> float c = 1.0f
c ==> 1.0
jshell> c = (float) 2.0
c ==> 2.0
jshell> double d = 1.0
d ==> 1.0
```

Java is a *C-style* language

Operators: Arithmetic

```
jshell> 1 + 1
$2 ==> 2
jshell> 2 * 2
$3 ==> 4
jshell> 3 / 2
$4 ==> 1
jshell> 3.0 / 2
$5 ==> 1.5
jshell> 3 % 2
$6 ==> 1
jshell> -3 % 2
$6 ==> -1
```

Java is a *C-style* language

Operators: `++` `--`

```
jshell> int a = 1;
a ==> 1

jshell> ++a
$29 ==> 2
jshell> --a
$30 ==> 1

jshell> a++
$31 ==> 2
jshell> a--
$32 ==> 1
```

Java is a *C-style* language

Operators: Bitwise

```
jshell> 1 & 2
$11 ==> 0

jshell> 1 | 2
$12 ==> 3

jshell> 1 ^ 2
$13 ==> 3

jshell> ~1
$14 ==> -2
```

Java is a *C-style* language

Operators: Logic

```
jshell> true && false
$15 ==> false

jshell> true || false
$16 ==> true

jshell> !true
$17 ==> false
```

Java is a *C-style* language

Operators: String

```
jshell> "hello" + "world"
$10 ==> "helloworld"
```

Java is a *C-style* language

Function / Method Syntax

```
jshell> int mul(int a, int b) {
   ...>     return a * b;
   ...> }
|  已创建 方法 mul(int,int)

jshell> mul(2, 3)
$20 ==> 6
```

Java is a *C-style* language

Control Flow: `if-else`

```
jshell> boolean canIGetA(double score) {
   ...>     if (score >= 90) {
   ...>         System.out.println("Good Job! You got a 4.0!");
   ...>         return true;
   ...>     } else if (score >= 70) {
   ...>         System.out.println("Keep Going! You can do better!");
   ...>         return false;
   ...>     } else {
   ...>         System.out.println("Emm... What happends?");
   ...>         return false;
   ...>     }
   ...> }
|  已创建 方法 canIGetA(double)
```

Java is a *C-style* language

Control Flow: `if-else`

```
jshell> canIGetA(100)
Good Job! You got a 4.0!
$2 ==> true

jshell> canIGetA(80)
Keep Going! You can do better!
$3 ==> false

jshell> canIGetA(0)
Emm... What happends?
$4 ==> false
```

Java is a *C-style* language

Control Flow: `?:`

```
jshell> int gcd(int a, int b) {
   ...>     return b == 0 ? a : gcd(b, a % b);
   ...> }
|  已创建 方法 gcd(int,int)

jshell> gcd(48, 18)
$26 ==> 6
```

Java is a *C-style* language

Control Flow: `switch`

```
jshell> int calculate(int a, int b, char op) {
   ...>     int res;
   ...>     switch(op) {
   ...>         case '+': res = a + b; break;
   ...>         case '-': res = a - b; break;
   ...>         case '*': res = a * b; break;
   ...>         case '/': res = a / b; break;
   ...>         default: res = -1;
   ...>     }
   ...>     return res;
   ...> }
|  已创建 方法 calculate(int,int,char)
```

Java is a *C-style* language

Control Flow: `switch`

```
jshell> calculate(2, 3, '+')
$6 ==> 5
jshell> calculate(2, 3, '-')
$7 ==> -1
jshell> calculate(2, 3, '*')
$8 ==> 6
jshell> calculate(2, 3, '/')
$9 ==> 0
jshell> calculate(2, 3, '^')
$10 ==> -1
```

Java is a *C-style* language

Control Flow: `for`

```
jshell> boolean isPrime(int a) {
   ...>     if (a <= 1) return false;
   ...>     for (int i = 2; i * i <= a; i++) {
   ...>         if(a % i == 0) return false;
   ...>     }
   ...>     return true;
   ...> }
|  已创建 方法 isPrime(int)
```

Java is a *C-style* language

Control Flow: `for`

```
jshell> for (int a = 1; a <= 100; a++) {
   ...>     if (isPrime(a)) System.out.print(a + " ");
   ...> }
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Java is a *C-style* language

Control Flow: `while`

```
jshell> import java.lang.Math;

jshell> double x;
x ==> 0.0

jshell> while( (x = Math.random()) < 0.8) {
   ...>     System.out.println(x + " < 0.8");
   ...> }
0.17636000596270907 < 0.8
0.5029857602823097 < 0.8
0.3349284868587292 < 0.8
0.3650514335114553 < 0.8
```

Java is a *C-style* language

Control Flow: `do-while`

```
jshell> do {
   ...>     x = Math.random();
   ...> } while(x < 0.8);

jshell> x
x ==> 0.8346328176085303
```

Other basic syntax & API

[JEP286, Java10] `var` type inference

```
jshell> var mapOfMap = new HashMap<String, Map<String, String>>()
mapOfMap ==> {}
jshell> mapOfMap.put("Java", Map.of("Type", "Strong-Typed", "Paradigm", "OOP"))
$111 ==> null
jshell> mapOfMap
mapOfMap ==> {Java={Paradigm=OOP, Type=Strong-Typed}}

jshell> mapOfMap.forEach((var key, var val) -> System.out.println(key + " : " + val))
Java : {Paradigm=OOP, Type=Strong-Typed}
```

Other basic syntax & API

[JEP378, Java15] Text Blocks

```
jshell> """
   ...> Line 1
   ...> Line 2
   ...> Line 3
   ...> Line 4
   ...> """
$4 ==> "Line 1\nLine 2\nLine 3\nLine 4\n"
```

Other basic syntax & API

Array

```
jshell> int[] intArr = {2, 0, 1, 9, 0, 1, 0, 8, 9, 5};
intArr ==> int[10] { 2, 0, 1, 9, 0, 1, 0, 8, 9, 5 }
jshell> intArr[1]
$52 ==> 0
jshell> intArr[20]
|  异常错误 java.lang.ArrayIndexOutOfBoundsException: Index 20 out of bounds for length 10
|        at (#53:1)
jshell> for (int i : intArr) System.out.print(i)
2019010895
jshell> intArr.length
$53 ==> 10

jshell> int[] intArr2 = new int[10];
intArr2 ==> int[10] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
```

Other basic syntax & API

Array

```
jshell> var matrix = new int[5][5];
matrix ==> int[5][] { int[5] { 0, 0, 0, 0, 0 }, int[5] { 0,  ... int[5] { 0, 0, 0, 0, 0 } }
jshell> for (int i = 0; i<5; i++) {
   ...>     for(int j = 0; j<5; j++) {
   ...>         matrix[i][j] = i * j;
   ...>     }
   ...> }
jshell> matrix
matrix ==> int[5][] {
    int[5] { 0, 0, 0, 0, 0 },
    int[5] { 0, 1, 2, 3, 4 },
    int[5] { 0, 2, 4, 6, 8 },
    int[5] { 0, 3, 6, 9, 12 },
    int[5] { 0, 4, 8, 12, 16 } }
```

Other basic syntax & API

Array Utility

```
jshell> import java.util.Arrays;
jshell> Arrays.sort(intArr)
jshell> intArr
intArr ==> int[10] { 0, 0, 0, 1, 1, 2, 5, 8, 9, 9 }
jshell> Arrays.binarySearch(intArr, 5)
$64 ==> 6

jshell> System.out.println(intArr)
[I@2d6eabae
jshell> System.out.println(Arrays.toString(intArr))
[0, 0, 0, 1, 1, 2, 5, 8, 9, 9]
jshell> System.out.println(Arrays.deepToString(matrix))
[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8], [0, 3, 6, 9, 12], [0, 4, 8, 12, 16]]
```

Other basic syntax & API

String

```
jshell> String s = "I Love Java"
s ==> "I Love Java"
jshell> s.length()
$37 ==> 11
jshell> s.split(" ")
$38 ==> String[3] { "I", "Love", "Java" }
jshell> s.concat(" and Rust")
$39 ==> "I Love Java and Rust"
jshell> s.contains("C++")
$40 ==> false
jshell> s.charAt(2)
$41 ==> 'L'
```

Other basic syntax & API

String

```
jshell> s.replace("Java", "C++")
$42 ==> "I Love C++"
jshell> s.toUpperCase()
$46 ==> "I LOVE JAVA"
jshell> s.toLowerCase()
$47 ==> "i love java"
jshell> s.startsWith("I")
$48 ==> true
jshell> s.endsWith("Java")
$49 ==> true
jshell> s
s ==> "I Love Java"
```

Other basic syntax & API

String

```
jshell> s == "I Love Java"
$43 ==> true
jshell> s == new String("I Love Java")
$44 ==> false
jshell> s.equals(new String("I Love Java"))
$45 ==> true
jshell> s.equalsIgnoreCase("i lOve jAVA")
$50 ==> true
```

Other basic syntax & API

Console IO

```
jshell> System.out.println("I Love Java")
I Love Java

jshell> System.out.print("I Love Java")
I Love Java
jshell> System.out.printf("I Love %s\n", "Java")
I Love Java
$82 ==> java.io.PrintStream@4d95d2a2
```

Other basic syntax & API

Console IO

```
jshell> import java.util.Scanner;
jshell> var input = new Scanner(System.in);
jshell> input.hasNextInt()
100
$86 ==> true
jshell> input.nextInt()
$87 ==> 100
jshell> input.hasNextBoolean()
100
$89 ==> false
```

Other basic syntax & API

BigInteger **BigDecimal**

```
jshell> import java.math.BigInteger;

jshell> 1000000000000000 + 1
|  错误:
|  整数太大
|  1000000000000000 + 1
|  ^
jshell> new BigInteger("1000000000000000").add(BigInteger.ONE)
$92 ==> 1000000000000001

jshell> new BigInteger("19260817").isProbablePrime(10)
$93 ==> true
```

Other basic syntax & API

BigInteger **BigDecimal**

```
jshell> import java.math.BigInteger;

jshell> 0.1 + 0.2
$94 ==> 0.30000000000000004

jshell> new BigDecimal("0.1").add(new BigDecimal("0.2"))
$95 ==> 0.3

jshell> new BigDecimal("0.000000001").toEngineeringString()
$100 ==> "1E-9"
```

Other basic syntax & API

Math

```
jshell> Math.
E                   IEEEremainder(      PI                  abs(                absExact(
acos(               addExact(           asin(               atan(               atan2(
cbrt(               ceil(               class               copySign(           cos(
cosh(               decrementExact(     exp(                expm1(              floor(
floorDiv(           floorMod(           fma(                getExponent(        hypot(
incrementExact(     log(                log10(              log1p(              max(
min(                multiplyExact(      multiplyFull(       multiplyHigh(       negateExact(
nextAfter(          nextDown(           nextUp(             pow(                random()
rint(               round(              scalb(              signum(             sin(
sinh(               sqrt(               subtractExact(      tan(                tanh(
toDegrees(          toIntExact(         toRadians(          ulp(
```

Other basic syntax & API

Math

```
jshell> Math.sin(Math.PI / 4)
$103 ==> 0.7071067811865475
jshell> Math.floor(1.002)
$104 ==> 1.0
jshell> Math.log(Math.E)
$107 ==> 1.0
jshell> Math.pow(2, 1.5)
$108 ==> 2.82842712474619
jshell> Math.sqrt(2)
$109 ==> 1.4142135623730951
```

CodeLab 1-1 a +-*/^&| b

CodeLab 1-2 QuickSort