福建師範大學
FUJIAN NORMAL UNIVERSITY

Java语言

学　　期：2025-2026-1

教　　师：赵珊珊

第1章 Java简介

主要内容

- 1.1Java概述
- 1.2Java的运行机制
- 1.3JDK的使用
- 1.4Java开发工具Eclipse

1.1Java概述

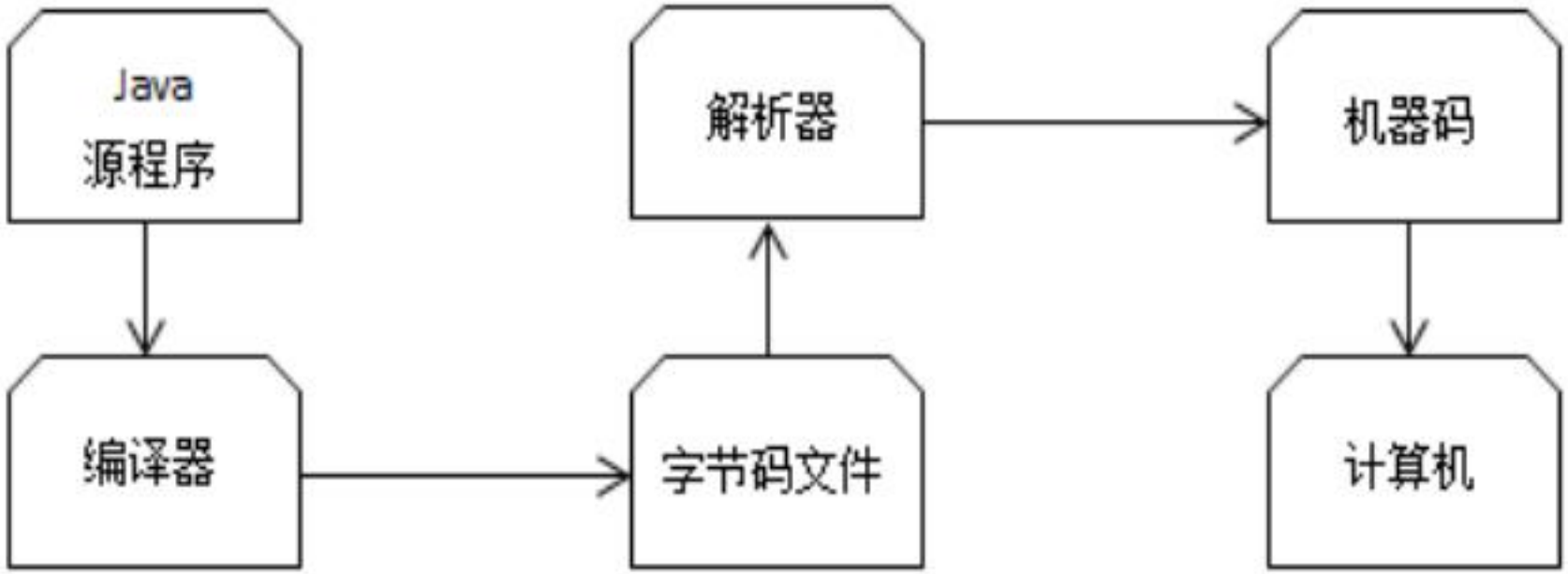
- 1.1.1 Java主要应用方向
  - 1. Java SE（Java Platform Standard Edition）
  - 2. Java EE（Java Platform，Enterprise Edition）
  - 3. Java ME（Java Platform，Micro Edition）

1.1Java概述

- 1.1.2 Java语言的特点
  - 1.跨平台性
  - 2.面向对象
  - 3.安全性
  - 4.多线程和同步机制
  - 5.简单易用

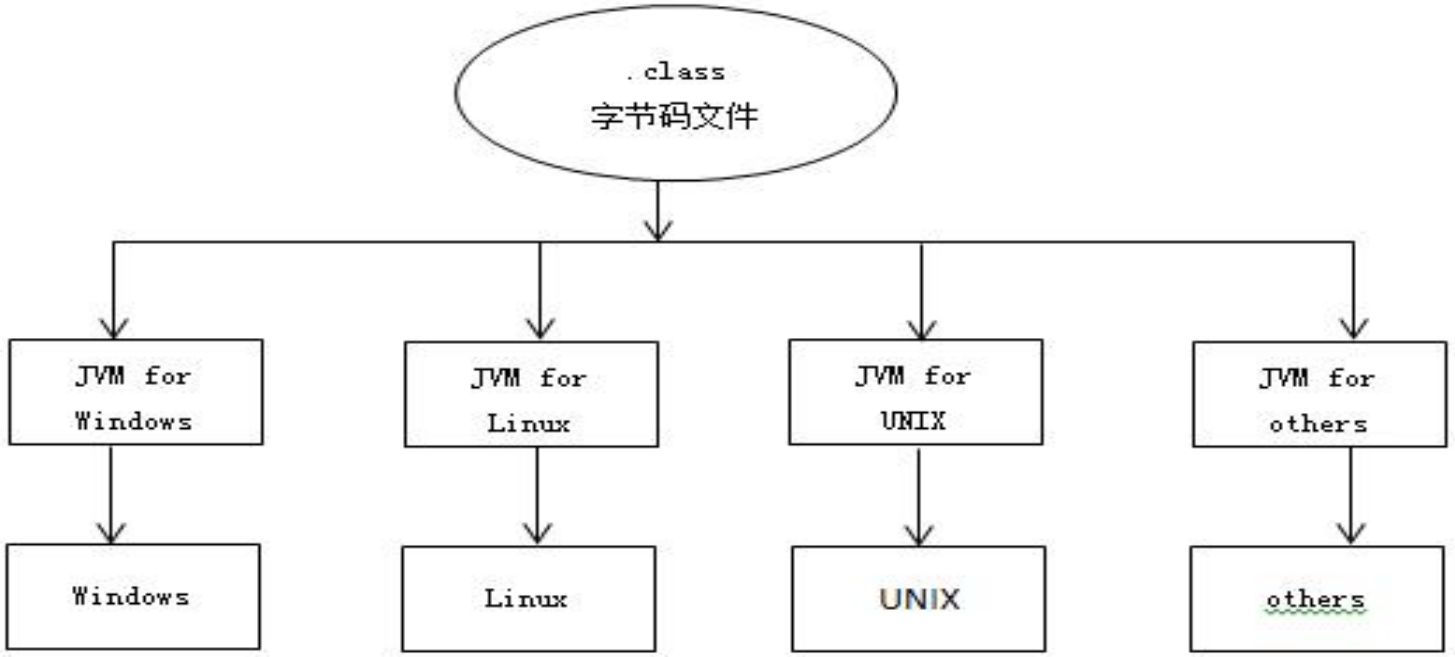
1.2 Java的运行机制

- Java程序运行时，必须经过编译和运行两个步骤。



1.2 Java的运行机制

- Java的class文件是在Java虚拟机（Java Virtual Machine，JVM）上运行的。JVM是在一台计算机上由软件或硬件模拟的计算机，JVM可以实现Java程序的跨平台运行，即运行的操作平台各不相同。



1.2 Java的运行机制

- 1.2.1什么是JDK
- JDK（Java Development Kit） 称为Java开发包或Java开发工具，是一个写JavaApplet小程序和Application应用程序的开发环境。JDK 是整个Java的核心，包括了Java运行环境（Java Runtime Envirnment），一些Java工具和Java的核心类库（Java API）。

Java Applet由于其安全性和兼容性问题，已经被逐步淘汰。

1.安全性问题：

- Java Applet运行在客户端，这使得它们容易被恶意软件利用，因为它们可以直接访问用户的系统资源。
- Oracle在2015年4月停止了对Java Applet的安全更新，这意味着一旦发现安全漏洞，没有官方的修复途径。

2.兼容性问题：

- 随着Web技术的发展，许多现代浏览器默认禁用或不再支持Java插件，这导致了大量依赖于Java Applet的旧网站出现问题。

2023年最新统计数据显示，目前全球仅有不到0.1%的网站还在使用JavaApplets技术，这个数字还在持续下降中

1.HTML5和JavaScript

- 利用HTML5和JavaScript可以实现几乎所有Java Applet能做的功能，且无需依赖外部插件。

1.React, Angular, Vue等现代前端框架

- 这些框架结合使用HTML5、CSS3和JavaScript，提供了构建复杂、交互式Web应用的能力。

1.2 Java的运行机制

- 1.2.2什么是JRE
- JRE（Java Runtime Environment）是Java运行环境，是运行Java程序所必需的环境集合，包含JVM标准实现及Java核心类库。JRE不包含开发工具，如编译器、调试器和其他工具。

1.3 JDK的使用

- 1.3.1 安装JDK
- https://www.oracle.com/java/technologies/downloads/#java17
- 注：教材介绍Java8，教学选择Java 17，因为Java17是新的长期支持（LTS）版本。

特性维度	Java 8 (2014)	Java 11 (2018)	Java 17 (2021)	Java 19 (2022)
LTS支持	是 (但官方主流支持已终止)	**是** (支持至2023年9月，部分供应商提供更久支持)	**是 (官方支持至2029年，当前推荐)**	否 (非LTS版本，每6个月迭代)
关键新特性	Lambda表达式、Stream API、默认方法、新日期时间API	var局部变量类型推断、HTTP Client API、ZGC	密封类、模式匹配、记录类	虚拟线程（预览）、结构化并发（孵化）、记录模式（预览）
教学适用性	经典、稳定、资源丰富，但特性较老，缺乏现代语言特性	承上启下，引入了重要新特性，但部分特性在后续版本更成熟	**特性丰富且稳定，兼顾传统与现代，适合系统学习**	特性最新且前沿，但多为预览版，不适合教学稳定性要求
性能与稳定性	久经考验，但性能优化不及新版本	性能优化（如ZGC），更稳定	**性能进一步提升（ZGC/Shenandoah GC优化），非常稳定**	性能有潜力，但作为非LTS版本，稳定性不及LTS版本
社区与生态	曾经是绝对主流，现有存量项目多，但新项目较少采用	企业过渡版本，生态丰富	**新项目首选LTS，社区活跃，生态日益完善**	社区关注度高，但生产环境采用率低

1.3 JDK的使用

- 1.3.1 安装JDK
- 下载完成后，运行安装程序。
- Windows：双击下载的 .exe或 .msi安装文件，按照向导提示操作，一般使用默认设置即可

1.3 JDK的使用

- 1.3.2系统环境变量
  - Java_home
  - Path

设置 JAVA_HOME：

•右键点击“此电脑” -> “属性” -> “高级系统设置” -> “环境变量”。

•在“系统变量”部分点击“新建”。

•变量名：JAVA_HOME

•变量值：JDK 17安装路径（默认C:\Program Files\Java\jdk-17）

编辑 Path变量：

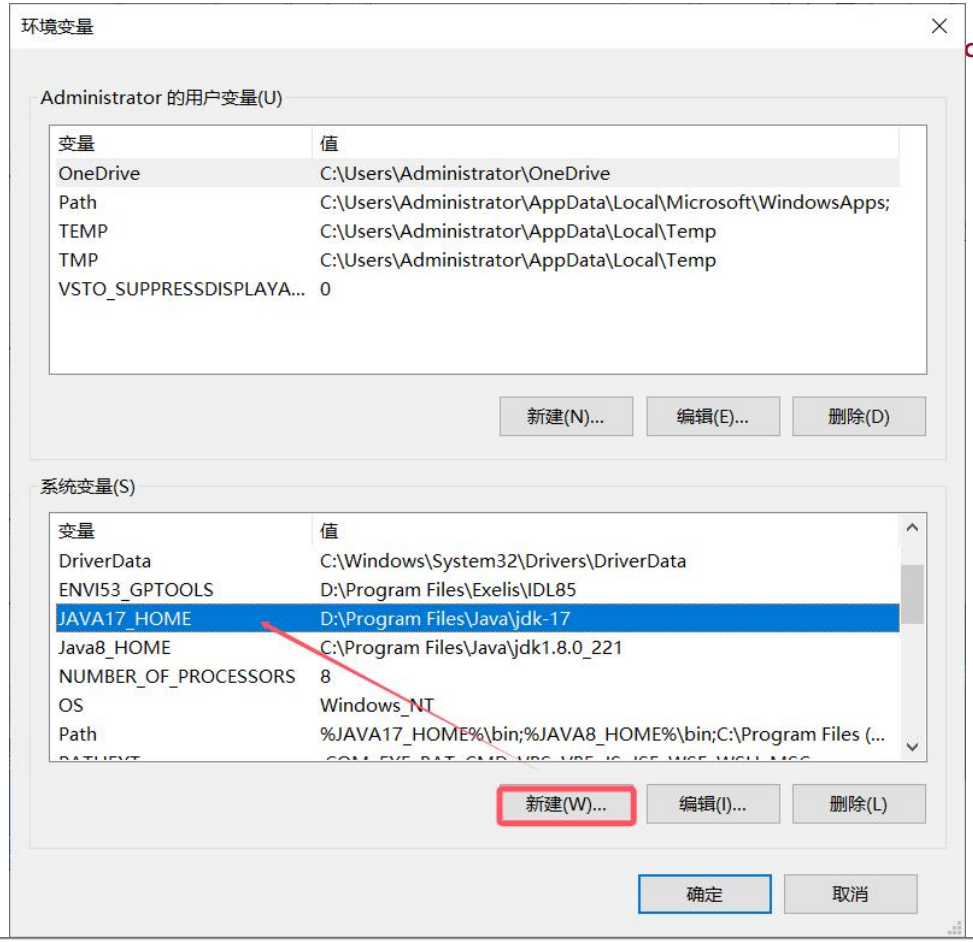
•在“系统变量”中找到并选中 Path变量，点击“编辑”。

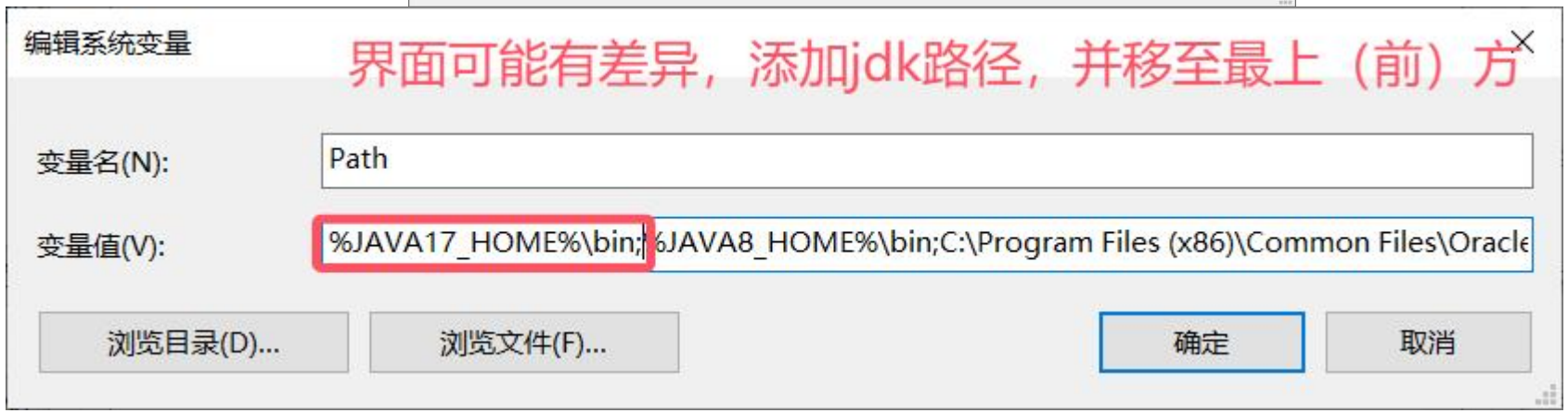
•点击“新建”，添加 %JAVA_HOME%\bin。

•在所有窗口中点击“确定”保存更改。

1.3 JDK的使用

- 1.3.2系统环境变量
  - Java_home
  - Path





1.4 Java开发工具Eclipse

- 1.4.1 Eclipse简介
- 1.4.2 Eclipse的安装与启动（解压即可，需要先设置Project目录）
- 1.4.3 Java注释
  - 单行注释
  - 多行注释
  - 文档注释

第2章Java语法基础

第2章Java语法基础

- 2.1Java语法
- 2.2 基本数据类型
- 2.3 变量和常量
- 2.4 运算符和表达式
- 2.5程序流程控制

2.1Java语法

- 2.1.1基本语句
- 2.1.2 基本格式
- 2.1.3 Java中的标识符和关键字
  - 1.Java标识符
  - 2.Java关键字

2.2 基本数据类型

- 在Java编程语言中，主要有两种类型的数据，基本数据类型和引用数据类型。
- 基本数据类型是由一组简单数据组成的数据类型，其数据是不可分解的。
- Java的引用数据类型包括数组、类和接口。
- 数组型变量本身不存储实际的值，而是代表了指向内存中存放实际数据的位置

2.2 基本数据类型

- 2.2.1整数类型

类型	空间大小	取值范围	默认值
byte	8位（1字节）	$-2^7 \sim 2^7-1$	(byte)0
short	16位（2字节）	$-2^{15} \sim 2^{15}-1$	(short)0
int	32位（4字节）	$-2^{31} \sim 2^{31}-1$	0
long	64位（8字节）	$-2^{63} \sim 2^{63}-1$	0L

2.2 基本数据类型

- 2.2.2浮点类型

类型	空间大小	取值范围	默认值
float	32位（4字节）	-3.4×10^{38}~3.4×10^{38}	0.0f
double	64位（8字节）	-1.79×10^{308}~1.79×10^{308}	0.0d

2.2 基本数据类型

- 2.2.3布尔类型
- 布尔类型只有两种值：真和假，通常用关键字true和false来表示。与C++语言不同的是，Java的布尔类型只能是真或假，不能代表整数（0或1）。Java的布尔类型用boolean表示，占用1个字节内存空间，布尔类型的默认值是“false”。

2.2 基本数据类型

- 2.2.4字符类型
- 字符型变量用来存储单个字符，字符型值必须使用英文半角格式的单引号引起来，如’a’、’b’。Java语言使用char表示字符型，占用2个字节内存空间，取值范围为0~65535之间的整数，默认值是’\n0000’。

2.3 变量和常量

- 2.3.1常量
  - 常量的值是固定的，不可改变的。在Java中利用关键字final声明常量。

2.3 变量和常量

- 2.3.2变量
  - 在程序运行过程中，空间内的值是变化的，这个内存空间就称为变量，为了便于操作，给这个空间取个名字，称为变量名。

2.3 变量和常量

- 2.3.3数据类型之间的相互转换
- 按照优先关系，转换分为两种，自动类型转换和强制类型转换。

2.3 变量和常量

- 1.自动类型转换
- 按照优先关系，低级数据要转换成高级数据时，进行自动类型转换。

操作数1类型	操作数2类型	转换后的类型
byte或short	int	int
byte或short或int	long	long
byte或short或int或long	float	float
byte或short或int或long或float	double	double
char	int	int

2.3 变量和常量

- 2.不兼容强制类型转换
- 允许的转换包括
- byte→ short→ int→ long→ float→ double 以及char→ int

2.4 运算符和表达式

- 2.4.1算术运算符和算术表达式
- 2.4.2 赋值运算符和表达式
- 2.4.3关系运算符和表达式
- 2.4.4逻辑运算符和表达式
- 2.4.5位运算符和表达式
- 2.4.6 条件运算符和表达式
- 2.4.7运算符的优先级

```java
// 传统if-else if链：冗长且易漏写break
String describe(Object obj) {
    if (obj instanceof Integer) {
        return "整数: " + obj;
    } else if (obj instanceof String) {
        return "字符串长度: " + ((String) obj).length();
    } else {
        return "未知类型";
    }
}

// Java 17 switch模式匹配：简洁、安全、可读性强
String describe(Object obj) {
    return switch (obj) {
        case Integer i -> "整数: " + i;
        case String s -> "字符串长度: " + s.length();
        case null -> "空对象"; // 显式处理null
        default -> "未知类型";
    };
}
```

2.5程序流程控制

- 2.5.1顺序结构
- 结构化程序中最简单的结构就是顺序结构。顺序结构是按照程序语句出现的先后顺序去执行，直到程序结束。

2.5程序流程控制

- 2.5.2选择结构
  - 1.if语句
  - 2.if-else语句
  - 3.if-else if语句

2.5程序流程控制

- 2.5.2选择结构
- 4.switch语句
  - （1）switch语句的判断条件只能是byte、short、char和int四种基本类型，JDK5.0开始支持枚举类型，JDK7.0开始支持String类型，不能是boolean类型。
  - （2）常量1~常量N必须与判断条件类型相同，且为常量表达式，不能是变量。
  - （3）case子句后面可以有多条语句，这些语句可以使用大括号括起来。
  - （4）程序将从第一个匹配的case子句处开始执行后面的所有代码（包括后面case子句中的代码）。可以使用break跳出switch语句。如果没有break语句，当程序执行完匹配的case语句序列后，后面的case子句起不到跳出switch结构的作用，程序还会继续执行后面的case语句序列。因此在每个case中，用break语句终止后面的case分支语句的执行。
  - （5）default语句是可选的，当所有case子句条件都不满足时执行。

2.5程序流程控制

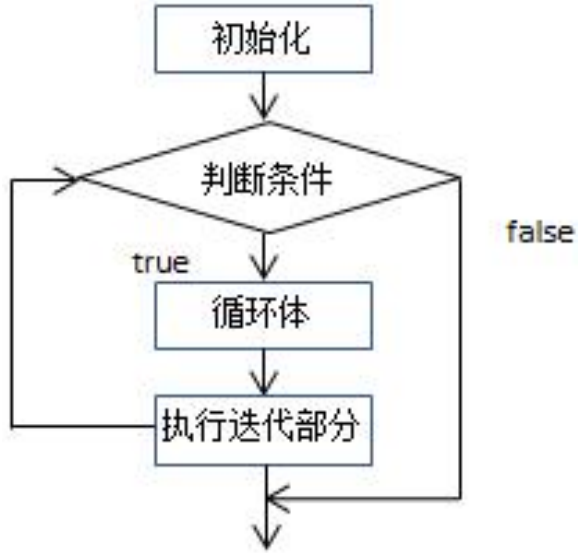
- 2.5.3循环结构
- 循环结构是程序中的另一种重要结构。
- 一个循环结构一般包含以下几部分：
  - （1）初始部分：用来设置循环控制的初始化条件，如设置计数器。
  - （2）循环体部分：反复执行的一段代码。
  - （3）迭代部分：用来修改循环控制条件，常常在本次循环结束，下一次开始前执行。
  - （4）判断部分：也称终止部分，是一个关系表达式或布尔逻辑表达式，其值用来判断是否满足循环终止条件。每执行一次循环都要对该表达式求值。

2.5程序流程控制

- 2.5.3循环结构
  - while 循环语句
  - do-while循环语句
  - for循环语句

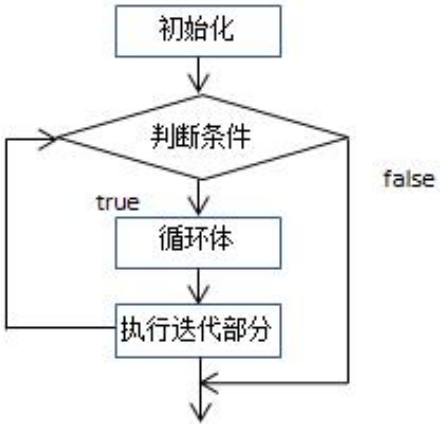
2.5程序流程控制

- for循环语句



2.5程序流程控制

- 2.while语句
- 当不清楚循环会被重复执行多少次时，可以选择while循环和do-while循环



2.5程序流程控制

- 3.do-while语句
- do-while语句与while语句类似，它们之间的区别在于：while语句是先判断循环条件的真假，再决定是否执行循环体，而do-while语句则先执行循环体，然后再判断循环条件的真假，因此do-while循环体至少被执行一次。

2.5程序流程控制

- 2.5.4跳转语句
- 可以使用break、continue中断语句来实现循环执行过程中程序流程的跳转

第3章数组

第3章数组

- 3.1一维数组
- 3.2 二维数组
- 3.3数组作为方法的参数

3.1一维数组

- 数组是一种数据结构，是按一定顺序排列的相同类型的元素集合。数组实际上就是一连串类型相同的变量，这些变量用一个名字命名，即数组名，并用索引区分它们。使用数组时，可以通过索引来访问数组元素，如数组元素的赋值和取值。

3.1一维数组

- 3.1.1数组的声明
- 数组的声明包含两个部分：数组类型和数组的名称，定义数组的语法格式如下：

  type[] 数组名　　或　　type 数组名[];

3.1一维数组

- 3.1.2数组的初始化
- 1.静态初始化

静态初始化是指由程序员在初始化数组时为数组每个元素赋值，由系统决定数组的长度。数组的静态初始化有两种方式，具体示例如下：

```
int[] x;
x=new int[]{10,20,30,40,50};        或
int x[]=new int[]{10,20,30,40,50}
```

对于数组的静态初始化也可简写，具体示例如下：

```
int x[]={10,20,30,40,50}
```

3.1一维数组

- 3.1.2数组的初始化
- 2.动态初始化

动态初始化是指由程序员在初始化数组时指定数组的长度，由系统为数组元素分配初始值。数组动态初始化的具体示例如下：

```
int a[]=new int[10];
```

3.1一维数组

- 3.1.3数组的操作
- 1.访问数组
  - 在Java中，数组对象有一个length属性，用于表示数组的长度，所有类型的数组都是如此。
  - 数组中的变量又称为元素，每个元素都有下标（索引），下标从0开始

3.1一维数组

- 3.1.3数组的操作
- 2.数组遍历
  - 数组的遍历是指依次访问数组中的每个元素。

3.1一维数组

- 3.1.3数组的操作
- 3.数组排序
  - 数组排序是指数组元素按照特定的顺序排列。在实际应用中，经常需要对数据排序。数组排序有多种算法，本文介绍一种简单的排序算法—冒泡排序。这种算法是不断地比较相邻的两个元素，较小的向上冒，较大的向下沉，排序过程如同水中气泡上升，即两两比较相邻元素，反序则交换，直到没有反序的元素为止。

3.1一维数组

- 3.1.4数组的内存机制
- 数组是引用数据类型，因此数组变量就是一个引用变量，通常被存储在栈（Stack）内存中。数组初始化后，数组对象被存储在堆（Heap）内存中的连续内存空间，而数组变量存储了数组对象的首地址，指向堆内存中的数组对象。

3.2二维数组

- 二维数组可以看成以数组为元素的数组，常用来表示表格或矩形。
- 二维数组的声明，示例如下：

```
int[][] a;
int a[][];
```

3.3数组作为方法的参数

- 在Java中，可以使用数组作为方法的参数来传递数据。在使用数组参数时，应注意以下事项：
  - （1）在形参列表中，数组名后的括号不能省略，括号的个数和数组的维数要相同，但在括号中可以不给出数组元素的个数；
  - （2）在实参列表中，数组名后不需要括号；
  - （3）数组名作为实参时，传递的是地址，而不是具体的数组元素值，即实参和形参具有相同的存储单元。

地理目标匹配

- 高德地图获取66个POI目标，其中宿舍楼坐标不明。
- 展示GIS班的学生信息txt和POI目标txt。
- 启动eclipse，运行MatchObject，随机匹配学生与目标。

1-3章作业

- 见学习通课后作业与平时作业。