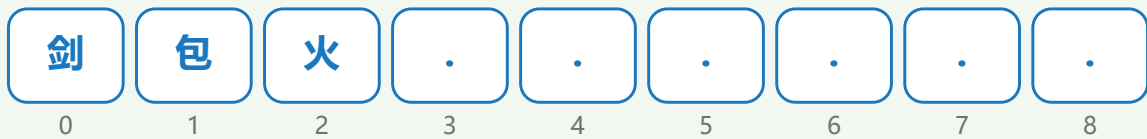


第15课

快捷栏数组

数组 索引与遍历

九格快捷栏，从0编号



长度9，合法索引0到8。一个点表示一个暂时没有物品的格子。

会创建和初始化数组；能用0到 `length-1` 的索引读取与替换元素；会选择普通 `for` 或增强 `for` 遍历；最终实现九格快捷栏的显示、替换、查找与空位统计。

前置背包：变量、循环、方法和作用域。数组像一排同类型箱子，每个格子都有从0开始的门牌号。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

本章使用 JDK 21，无需预览功能。故事与插画借用方块世界，练习运行在普通 Java 控制台。

开始前：把示例放对地方

本章使用 JDK 21，不需要预览功能。每个普通示例都给出完整程序，按标注的文件名保存。多个示例各放在自己的文件夹，避免辅助类同名。

终端命令：把中文占位符替换成实际名称

```
javac -encoding UTF-8 示例文件名.java  
java 示例类名
```

类名区分大小写。完整挑战代码若分成多段，请按顺序接在同一个文件中，不要为续段另建类。运行核对只列程序输出；需要输入的地方另给输入序列。

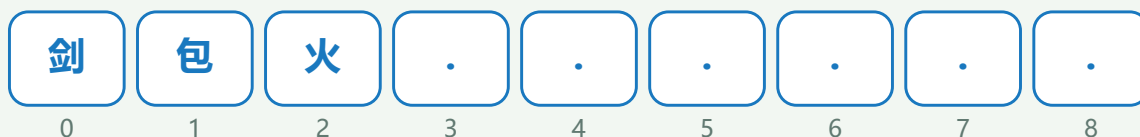
学习动作：先写下你的预测，运行后比较，再只改一个条件。标有“故意错误”的实验用于认识 Bug；它们与可直接运行的示例有清楚区分。

01 九格快捷栏从零开始

管理员喊：“把火把放第一格！”程序员把它放进 `slots[0]`。管理员沉默三秒，决定先学习数组。

数组把一组相同类型的元素放在按索引排列的格子中。数组长度固定，每个元素通过索引访问。

九格快捷栏：最后一格的索引是 8



这里用单个符号表示物品；共9格，索引0到8，长度为9。

带着四个问题出发

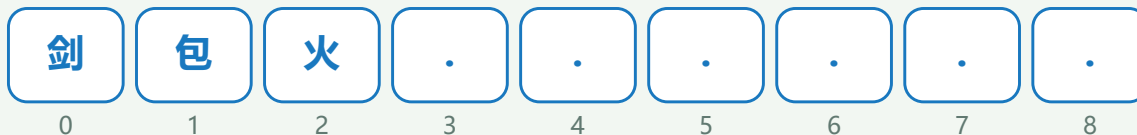
- 长度为9的数组，合法索引为什么是0到8？
- 新建 `String` 数组后，未放物品的格子是什么？
- 普通 `for` 和增强 `for` 分别适合什么任务？
- 怎样避免越界和 `null` 带来的错误？

本课 挑战：完成九格快捷栏：显示全部格子，替换指定位置，查找物品，并统计空位。

02 数组是一排同类型箱子

`String[] slots` 表示一组字符串格子，`new String[9]` 创建九个位置。数组变量保存的是整排格子的引用。

九格快捷栏：最后一格的索引是 8



这里用单个符号表示物品；共9格，索引0到8，长度为9。

L15Example02.java • 完整独立示例

```
public class L15Example02 {  
    public static void main(String[] args) {  
        String[] slots = {"剑", "面包", "火把"};  
        System.out.println(slots.length);  
        System.out.println(slots[0]);  
        System.out.println(slots[2]);  
    }  
}
```

运行核对

```
3  
剑  
火把
```

长度固定：数组创建后不能把9格变成10格；可以替换格子中的元素，但不能修改 `length`。

03 创建数组和默认值

Java 会给新数组填默认值。String[] 的空格子是 null，它不是空字符串，也不是写着“空气”的字符串。

新箱子先装着默认值

int元素

默认0

double元素

默认0.0

boolean元素

默认false

引用元素

默认null

L15Example03.java • 完整独立示例

```
public class L15Example03 {  
    public static void main(String[] args) {  
        int[] counts = new int[2];  
        boolean[] locks = new boolean[2];  
        String[] slots = new String[9];  
        System.out.println(counts[0]);  
        System.out.println(locks[0]);  
        System.out.println(slots[0]);  
    }  
}
```

运行核对

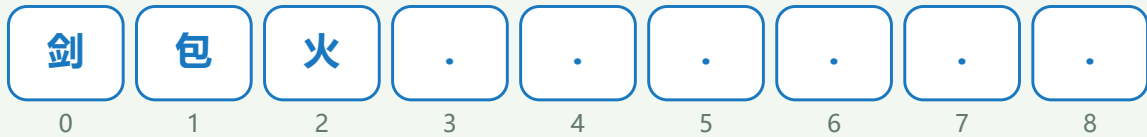
```
0  
false  
null
```

选择一种空位表示：可以用 null，也可以统一用“空气”。两种都行，但整个程序必须一致，并在使用前检查。

04 用索引读取和替换

索引放在方括号中。读取不会删除物品，赋值会替换该位置原来的元素。

九格快捷栏：最后一格的索引是 8



这里用单个符号表示物品；共9格，索引0到8，长度为9。

L15Example04.java • 完整独立示例

```
public class L15Example04 {  
    public static void main(String[] args) {  
        String[] slots = new String[9];  
        slots[1] = "面包";  
        String before = slots[1];  
        slots[1] = "胡萝卜";  
        System.out.println(before + " -> " + slots[1]);  
    }  
}
```

运行核对

面包 -> 胡萝卜

索引边界：合法索引满足 $\text{index} \geq 0 \ \&\& \ \text{index} < \text{slots.length}$ 。slots[9] 已经跑出九格快捷栏。

05 普通 for 逐格巡检

需要索引时使用普通 for。循环条件写 $i < \text{length}$ ，因为最后一个合法位置是 $\text{length} - 1$ 。

索引循环同时拿到位置和物品

开始

i 等于 0

条件

i 小于 length

读取

访问 $\text{slots}[i]$

更新

i 加 1，继续巡检

L15Example05.java • 完整独立示例

```
public class L15Example05 {  
    public static void main(String[] args) {  
        String[] slots = {"剑", null, "火把"};  
        for (int i = 0; i < slots.length; i++) {  
            String text = slots[i] == null ? "空位" : slots[i];  
            System.out.println(i + ": " + text);  
        }  
    }  
}
```

运行核对

0: 剑
1: 空位
2: 火把

循环契约：初始化从 0 开始，条件是 $i < \text{length}$ ，循环体只用合法的 $\text{slots}[i]$ ，每轮 i 增加 1。

06 增强 for 只看物品

只关心元素、不需要位置时，增强 for 更简洁。循环变量得到元素值，但不能用它直接替换数组格子。

增强 for 依次取出每个元素

item

本轮元素值的副本

读取

适合逐件统计

重新赋值

不会替换数组格子

替换元素

需要slots[i]赋值

L15Example06.java • 完整独立示例

```
public class L15Example06 {  
    public static void main(String[] args) {  
        String[] slots = {"面包", "火把"};  
        for (String item : slots) { item = "钻石"; }  
        for (String item : slots) { System.out.println(item); }  
    }  
}
```

运行核对

面包
火把

item 是本轮接收元素值的局部变量。给它重新赋值，不会替换数组格子；要替换格子，使用 slots[i] = ...。第18课再研究修改元素所指对象的情况。

如何选择

- 需要索引或要替换格子：普通 for。
- 只读取每个元素：增强 for。
- 需要同时访问前后元素：普通 for 更合适。

常见误会：给增强 for 的循环变量重新赋值，只改变这次循环的局部变量，不会替换原数组元素。

07 统计总数平均值最大值

数组与循环组合后，可以一次处理整组数据。最大值算法先用第一个元素初始化，避免随便猜一个起点。

数字数组进入统计工作台

total

各格数量相加

average

转为小数再除长度

maximum

从第一个元素开始

空数组

先检查再访问与除法

L15Example07.java • 完整独立示例

```
public class L15Example07 {  
    public static void main(String[] args) {  
        int[] counts = {3, 12, 7, 18, 5};  
        int total = 0, maximum = counts[0];  
        for (int count : counts) {  
            total += count;  
            maximum = Math.max(maximum, count);  
        }  
        double average = (double) total / counts.length;  
        System.out.println(total + "," + average + "," + maximum);  
    }  
}
```

运行核对

45,9.0,18

这个示例的数组已知非空。一般方法应先处理 `length == 0`，才能读 `counts[0]` 或计算平均值；空数组没有最大元素，平均值也没有定义。

空数组警报：访问 `ores[0]` 前先确认 `length > 0`。空数组没有第一个元素，也不能直接计算平均值。

08 搜索物品和计算空位

搜索数组时，先处理 null，再调用字符串方法。找到后是否继续，取决于你要找第一处还是统计全部。

巡检每个格子并记录发现

find

返回首个索引或-1

空位

本课程统一使用null

比较

非null目标调用
equals

空位数

逐格计数

L15Example08.java • 完整独立示例

```
public class L15Example08 {
    static int find(String[] slots, String target) {
        if (target == null) return -1;
        for (int i = 0; i < slots.length; i++)
            if (target.equals(slots[i])) return i;
        return -1;
    }
    static int emptyCount(String[] slots) {
        int result = 0;
        for (String item : slots) if (item == null) result++;
        return result;
    }
    public static void main(String[] args) {
        String[] slots = {"剑", null, "火把", null};
        System.out.println(find(slots, "火把"));
        System.out.println(find(slots, "钻石"));
        System.out.println(emptyCount(slots));
    }
}
```

运行核对

```
2
-1
2
```

安全比较：写 "钻石剑".equals(slots[i])，即使该格是 null 也不会抛出 NullPointerException。

09 数组故障诊所

`ArrayIndexOutOfBoundsException` 通常表示索引越界；`NullPointerException` 常表示对 `null` 调用了方法。两者修法不同。

越界和 `null` 是两只常见苦力怕

越界

索引length不合法

错误边界

小于等于会多走一格

`null`数组

不能读取length

空数组

长度0，但不是`null`

故障实验：索引3超出合法范围0到2，会发生 `ArrayIndexOutOfBoundsException`。下面故意保留错误，请先预测，再尝试修复。

故障实验：slots 还没有指向数组，会发生 `NullPointerException`。下面故意保留错误，请先预测，再尝试修复。

一次修一只：不要同时改循环条件和数组长度。先证明是哪一轮出错，再做最小修改。

BadIndex.java • 故意错误示例

```
public class BadIndex {
    public static void main(String[] args) {
        int[] counts = new int[3];
        System.out.println(counts[counts.length]);
    }
}
```

BadArrayNull.java • 故意错误示例

```
public class BadArrayNull {
    public static void main(String[] args) {
        String[] slots = null;
        System.out.println(slots.length);
    }
}
```

09 数组故障诊所（续）

急诊顺序

- 读异常名称和第一条属于自己代码的行。
- 打印 index、length 和当前元素。
- 索引用 $< \text{length}$ ，引用使用前检查 null。

10 挑战 蓝图 九格快捷栏

挑战 类叫 HotbarManager。快捷栏固定九格，初始物品放在0、1、2和8号位置。

显示 替换 查找 统计四个功能

show
逐格显示

replace
先检查索引和物品

find
查找首个匹配

emptyCount
统计null格子

验收顺序

- 先显示初始快捷栏。
- 把1号位从面包替换成金胡萝卜。
- 查找火把，应返回8。
- 统计空位，应返回5。
- 尝试索引9，程序不得崩溃。

验收表

功能	输入	输出或效果
show	数组	逐格显示0到8
replace	数组 索引 新物品	合法时替换并返回true
find	数组 物品名	返回首个索引或-1
emptyCount	数组	返回 null 格子数

最小版本：先完成 show 和 replace，再增加 find 与 emptyCount。每加一个方法就运行一次。

11 挑战 实现 显示替换统计

新建 HotbarManager.java。完整程序使用普通 for 显示索引，使用增强 for 统计空位。

四个数组方法协同管理快捷栏

初始状态

显示全部九格

替换

将1号格换为胡萝卜

查找

火把在索引2

空位

共有5个

验收：运行后检查0到8共九行；确认1号位已替换；再测试-1、0、8、9四个索引。

11 挑战 实现 显示替换统计 (续)

HotbarManager.java • 完整独立示例

```
public class HotbarManager {
    static void show(String[] slots) {
        for (int i = 0; i < slots.length; i++) {
            String text = slots[i] == null ? "空位" : slots[i];
            System.out.println(i + ": " + text);
        }
    }
    static boolean replace(String[] slots, int index, String item) {
        if (index < 0 || index >= slots.length
            || item == null || item.isBlank()) return false;
        slots[index] = item;
        return true;
    }
    static int find(String[] slots, String target) {
        if (target == null) return -1;
        for (int i = 0; i < slots.length; i++)
            if (target.equals(slots[i])) return i;
        return -1;
    }
    static int emptyCount(String[] slots) {
        int result = 0;
        for (String item : slots) if (item == null) result++;
        return result;
    }
    public static void main(String[] args) {
        String[] slots = new String[9];
        slots[0] = "铁剑"; slots[1] = "面包";
        slots[2] = "火把"; slots[8] = "地图";
        show(slots);
        System.out.println(replace(slots, 1, "胡萝卜"));
        System.out.println("修改后的1号格: " + slots[1]);
        System.out.println("火把索引: " + find(slots, "火把"));
        System.out.println("空位: " + emptyCount(slots));
    }
}
```

运行核对

0: 铁剑
 1: 面包
 2: 火把
 3: 空位
 4: 空位
 5: 空位
 6: 空位
 7: 空位
 8: 地图
 true
 修改后的1号格: 胡萝卜
 火把索引: 2
 空位: 5

12 拓展任务与阶段复盘

这是基础语法阶段最后一课。第16课会把变量、条件、循环、方法和数组合成“背包管理员”项目。

三档任务把数组用进真实数据

木镐

矿石数组逐格显示

铁镐

统计一周采集量

钻石镐

实现循环移位

复盘

测试长度0与1

三档作业

- 木镐：把六种矿石放进 String 数组并逐项打印索引和名称。
- 铁镐：用 int 数组记录一周七天的挖矿数量，输出总数、平均值和最大值。
- 钻石镐：实现快捷栏向右轮换一格，最后一格回到0号位。

验收表

自评	能	还要练
我能说出合法索引范围	☐	☐
我会选择两种 for	☐	☐
我能处理越界和 null	☐	☐

下一课：第16课是阶段项目“背包管理员”。你会把输入、判断、循环、方法和数组装进同一个可运行程序。

自己的验收记录

运行之前写预期，运行之后记录实际结果。正常案例说明程序会做事；边界案例说明它知道何时停；错误案例说明它能拒绝不合理的请求。

把证据写进记录

类别	你选择的输入	预期与实际
正常	一个常用值或一次成功操作	是否逐行一致？
边界	最小值、最大值及相邻值	有没有少一格、多一步？
错误	非法数量、缺失对象或错误语法	是否明确拒绝？状态是否保持？

- 把一个 Bug 写成：触发条件、现象、原因、最小修复。
- 整理一个值得记住的规则，用自己的话解释，不能只复制定义。
- 改变一个示例，保留测试记录，让结果证明你的理解。

技术核对依据：Java SE 21 语言规范，作用域与访问控制（第6章）、类与构造器（第8章）、方法调用（第15章）。 <https://docs.oracle.com/javase/specs/jls/se21/html/index.html>