

第05课

聊天栏里的字符串魔法

让名字、等级和消息各就各位



一条消息只要少一个空格，玩家名和正文就会挤成一团。String 让程序能够保存文字，也能寻找、比较和整理文字。这一课，我们从最小欢迎语开始，沿着索引找到每个位置，最后做出一台能把杂乱数据整理成清楚消息的聊天格式化器。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

使用 Java 11 或更新版本。故事与插画借用方块世界，本系列练习运行在普通 Java 控制台。

01 空格也是信息



一条清楚的消息，应该让人一眼分清谁在说话、说了什么。

广播上出现“欢迎Alex加入红队”。你大概能猜懂，可一旦名字和消息更长，猜测就不够了。电脑不会自动替我们补标点和空格。字符串里的每个字符，都参与了最后的表达。

这里的“聊天栏”是作品的故事主题。程序先在控制台输出消息；真正发送到游戏服务器，需要以后学习服务器或模组接口。先把文字组织正确，是值得单独完成的一步。

- 拼接时留意边界与空格。
- 截取前先标出索引。
- 比较内容与清理格式，选择不同的方法。

02 String 保存文字，也返回新文字



固定文字写在双引号中；变量名负责读取已有内容。

片段：放在 main 中，单独运行

```
String player = "Alex";  
String team = "Red";  
String message = "找到钻石! ";  
String welcome = "欢迎 " + player + "! ";  
System.out.println(welcome);  
System.out.println(team + ": " + message);
```

运行核对

```
欢迎 Alex!  
Red: 找到钻石!
```

String 的 S 要大写。"player" 是一段固定文字，player 则读取变量。"欢迎" + player 会得到 "欢迎Alex"；要留空格，就明确写 "欢迎 "。

String 对象的内容不能被原地改写。strip、replace 和大小写方法会返回结果；想让变量保存这个结果，需要重新赋值。方法调用不是一句自动生效的愿望。

03 + 会算，也会连接

组成一条有边界的消息



连接各部分时，空格和标点要明确写在字符串里。

片段：放在 main 中，单独运行

```
int level = 8;
System.out.println("等级=" + level + 1);
System.out.println("等级=" + (level + 1));
int diamonds = 3, bonus = 2;
System.out.println("总钻石=" + (diamonds + bonus));
String levelText = String.valueOf(level);
System.out.println("Lv." + levelText);
```

运行核对

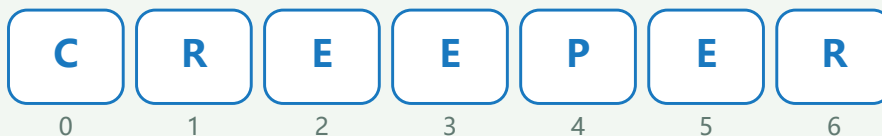
```
等级=81
等级=9
总钻石=5
Lv.8
```

第一行从左到右连接：先得到“等级=8”，再接上“1”。第二行的括号先算 $8+1$ ，再把9接进消息。看到数字和文字混在一起，先决定哪些部分是计算。

字符串拼接并不比加法“优先”。这里起作用的是表达式的分组、从左到右的结合顺序，以及当前操作数的类型。

04 索引从0开始，长度从数量出发

CREEPER 的七个位置



length() = 7; 合法索引为0到6。charAt(0)是C, charAt(6)是R。

片段：放在 main 中，单独运行

```
String mob = "CREEPER";  
System.out.println(mob.length());  
System.out.println(mob.charAt(0));  
System.out.println(mob.charAt(mob.length() - 1));
```

运行核对

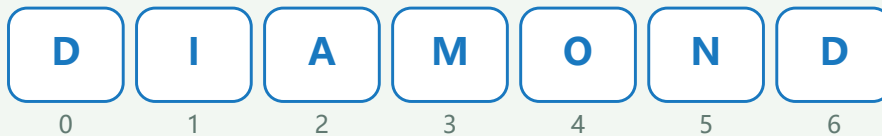
```
7  
C  
R
```

“有七个位置”和“最后的位置编号是六”并不矛盾：我们从0开始编号。charAt(7) 越过了末尾，空字符串也没有可以读取的第0个位置。

length() 统计 UTF-16 单元，本例的英文字母各占一个单元。某些表情符号会占两个单元，所以它不是所有文字场景里的“肉眼字符计数器”。

05 substring：左边拿，右边停

切下 DIAMOND 的前一段



substring(0, 3)拿索引0、1、2，得到DIA；索引3的M不包含。

片段：放在 main 中，单独运行

```
String ore = "DIAMOND";  
System.out.println(ore.substring(0, 3));  
System.out.println(ore.substring(3));  
System.out.println(ore.substring(ore.length() - 3));
```

运行核对

```
DIA  
MOND  
OND
```

两个参数分别是起点和结束边界。起点包含，结束边界不包含。只给一个起点时，就取到字符串末尾。先画编号，通常比反复试数字更快。

对本例，substring(1, 4) 得到 IAM。需要满足 $0 \leq \text{begin} \leq \text{end} \leq \text{length}()$ ；begin 与 end 相等时，结果可以是空字符串。

06 搜索：有没有，在哪里？

四个问题，四种返回结果

方法	所问的问题	返回
contains	有没有这段文字？	boolean
indexOf	第一次出现在哪里？	int, 未找到是-1
startsWith	开头是不是这段文字？	boolean
endsWith	结尾是不是这段文字？	boolean

片段：放在 main 中，单独运行

```
String msg = "I found diamond!";
System.out.println(msg.contains("diamond"));
System.out.println(msg.indexOf("diamond"));
System.out.println(msg.indexOf("emerald"));
System.out.println("/home now".startsWith("/"));
System.out.println(msg.endsWith("!"));
```

运行核对

```
true
8
-1
true
true
```

indexOf 的-1不是合法索引，别直接交给 charAt 或 substring。这些搜索方法区分大小写；"DIAMOND" 与 "diamond" 是不同内容。

07 内容相同，未必是同一个对象



两个箱子的内容可以一样，箱子却仍然是两个。

片段：放在 `main` 中，单独运行

```
String a = new String("diamond");
String b = new String("diamond");
System.out.println(a.equals(b));
System.out.println(a == b);
String role = null;
System.out.println("admin".equals(role));
```

运行核对

```
true
false
false
```

`equals` 比较 `String` 的内容；这里的 `==` 比较两个引用是否指向同一个对象。这个例子明确创建了两个对象，所以内容相同，引用比较仍为 `false`。

有时两个字面量的 `==` 恰好为 `true`，也不能把它当成内容比较规则。常量放左侧，如 `"admin".equals(role)`，在 `role` 为 `null` 时会返回 `false`，而不是调用 `null` 上的方法。

08 大小写转换：接住返回的新结果

归一化后再比较



片段：放在 main 中，单独运行

```
String command = "/Home";
String normalized = command.toLowerCase(java.util.Locale.ROOT);
System.out.println(normalized);
System.out.println(command.equalsIgnoreCase("/HOME"));
String team = "blue";
System.out.println(team.toUpperCase(java.util.Locale.ROOT));
System.out.println(command);
```

运行核对

```
/home
true
BLUE
/Home
```

最后一行仍是 /Home，因为我们没有给 command 重新赋值。对于固定格式的英文命令或标签，Locale.ROOT 让转换不依赖电脑当前的语言设置；暂时把它当成一个明确的转换选项。

只想比较英文内容是否忽略大小写相同，可以用 equalsIgnoreCase，通常不用先改原字符串。

09 两端空白与空内容，是两回事

空白清理站：逐项核对

数据	检查	结果
" Alex "	length()	10
strip 后的 "Alex"	length()	4
""	isEmpty()	true
" "	isEmpty()	false
" "	isBlank()	true

片段：放在 main 中，单独运行

```
String rawName = "  Alex  ";
String cleanName = rawName.strip();
System.out.println(rawName.length());
System.out.println(cleanName.length());
System.out.println("<" + cleanName + ">");
System.out.println("  ".isEmpty());
System.out.println("  ".isBlank());
```

运行核对

10
4
<Alex>
false
true

strip() 从 Java 11 起可用，按 Java 的空白字符定义清理两端；不会删除 "Hi Alex" 中间的空格。isEmpty 问长度是不是0；isBlank 还接受只有空白的字符串。

10 转义与 null：看清隐藏的规则



片段：放在 main 中，单独运行

```
String quote = "Alex 说: \"Hi!\"";  
String twoLines = "第一行\n第二行";  
String path = "C:\\Minecraft";  
System.out.println(quote);  
System.out.println(twoLines);  
System.out.println(path);  
System.out.println("bad word".replace("bad", "good"));
```

运行核对

```
Alex 说: "Hi!"  
第一行  
第二行  
C:\\Minecraft  
good word
```

反斜杠让某些字符进入字符串：\" 表示引号，\n 表示换行，\\ 表示一个反斜杠。它们在源代码里的样子，与运行后的结果不同。

片段：放在 main 中，单独运行

```
String player = null;  
boolean valid = player != null && !player.isBlank();  
System.out.println(valid);
```

运行核对

```
false
```

null 不是空字符串，也没有可调用的字符串方法。&& 的短路在这里守住右侧调用：左侧已为 false，就不再执行 player.isBlank()。

11 先设计聊天格式，再写代码



格式明确后，读者不用猜哪一段是名字，哪一段是消息。

格式化器的工作路线



本作品先使用写在变量里的文字。键盘读取要到下一课才学习，所以现在只需修改这些变量，再运行核对。队伍、姓名、消息清理后都应非空；等级使用1到30的练习范围。

- 空白只在两端清理，保留消息内部的空格。
- 不对整条消息强制改大写，否则正文也会变。
- 输出格式固定，便于用样例逐字核对。

12 完整作品：清理、检查、组织

完整程序 • ChatFormatter.java

```
import java.util.Locale;

public class ChatFormatter {
    public static void main(String[] args) {
        String team = " red ", player = " Alex ";
        String message = " /home ";
        int level = 8;
        team = team.strip().toUpperCase(Locale.ROOT);
        player = player.strip();
        message = message.strip();
        boolean valid = !team.isBlank() && !player.isBlank()
            && !message.isBlank() && level >= 1 && level <= 30;
        if (!valid) {
            System.out.println("请检查队伍、名字、消息和等级");
            return;
        }
        String chat = "[" + team + "] Lv." + level
            + " " + player + ": " + message;
        System.out.println(chat);
        System.out.println("命令=" + message.startsWith("/"));
    }
}
```

运行核对

```
[RED] Lv.8 Alex: /home
命令=true
```

if 继续充当借来的守门员，第7课会详细学习条件判断。这里的重点是追踪整理前后的文字，并看清最终消息怎样逐段拼成。

13 让测试覆盖不同边界

聊天格式化器的测试表

情况	修改项	主要结果	命令检查
正常	原始参考数据	[RED] Lv.8 Alex: /home	true
普通消息	message = " Hi! "	[RED] Lv.8 Alex: Hi!	false
大小写	team = " ReD "	仍是 [RED]	不变
空名字	player = " "	只输出检查提示	不执行
越界等级	level = 31	只输出检查提示	不执行

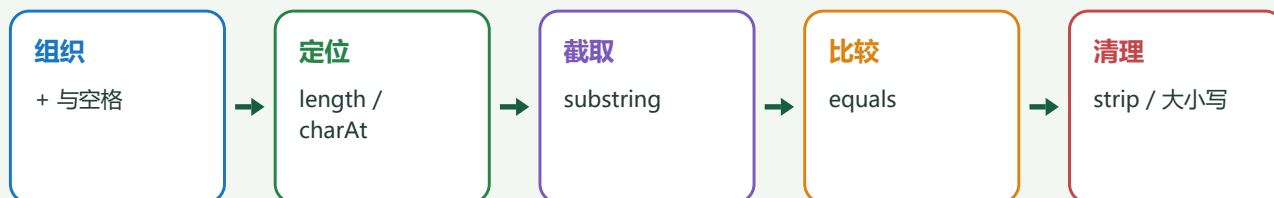
一张测试表同时写明“改什么”和“预期什么”，才能告诉你差异来自哪里。先只改一个字段，确认结果，再组合多个变化。空名字被拒绝时，不应继续读取它的首字符。

- 升级：输出 message.contains("diamond") 的结果，再测试大小写差异。
- 升级：为公告增加标题行，用 \n 将两行连起来。
- 升级：用 replace 替换一个明确的词，观察返回的新字符串。

replace 不是完整的内容审核系统，也不会自动合并所有重复标点。给每项升级写出具体规则，再挑合适的方法。

14 把方法与意图连起来

第五课的能力地图



- 基础：用三个字符串变量做两行告示牌，明确写出换行和空格。
- 进阶：清理两端带空格的名字，核对原长度与新长度；非空时再取首尾字符。
- 创造：为三条探险事件写消息模板，包含玩家、地点和事件，再做一次搜索或替换。

30秒自测

① substring(1, 4) 包含哪些索引？ ② " ".isEmpty() 与 isBlank() 相同吗？ ③ 运行 name.strip() 后，name 会自动改变吗？

答案：① 1、2、3。② 不同，前者false，后者true。③ 不会，需要接住返回值。下一课《让玩家输入指令》把固定数据改成用户回答。

技术参考：Java SE 11 的 String API, docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html。