

第06课

让玩家输入指令

让程序等待回答，也学会检查回答



以前，程序只会说我们提前写好的话。现在给它接上一条输入通道：它问名字，你来回答；它问等级，你来决定。不过，用户也可能输错、留空，或者给出一个大得离谱的数字。会读取只是开始，能解释和处理这些情况，程序才更像一个可靠的伙伴。

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 从独白到对话



程序先提问，再等待完整回答，交互才有清楚的顺序。

问“玩家名是什么？”之后，程序需要停下来等待。读取结束，回答才会成为变量的值。一次可靠的交互应该让人知道：现在要回答什么、答案有什么要求、没有通过时该怎样修改。

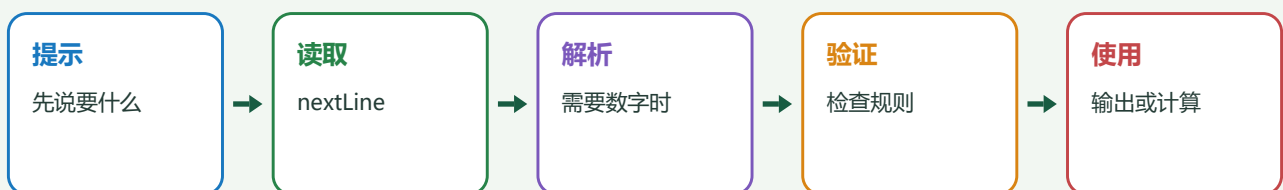
- 读取：把输入通道中的内容带进程序。
- 解析：把数字文字变成数值。
- 验证：检查数据是否符合作品的规则。

这三件事不能互相代替。读到“999999999999”不表示它装得进 int；解析成整数，也不表示它在1到30的等级范围内。

02 Scanner 接通输入通道



输入是一条完整流水线



完整程序 • FirstInput.java

```
import java.util.Scanner;

public class FirstInput {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("玩家名: ");
        String name = in.nextLine();
        System.out.println("收到: " + name);
    }
}
```

运行核对

玩家名: 收到: Red Stone

输入 Red Stone 再按回车，会读到含空格的完整名字。运行核对只列程序输出；你手动输入时，终端通常还会显示刚敲进去的文字。一个小程序里使用一个 Scanner 读取 System.in。

03 nextLine、next、nextInt 各读什么？

同一条通道，不同读取边界

方法	输入示例	带走的内容
nextLine()	Red Stone + 回车	整行 Red Stone
next()	Red Stone + 回车	第一个记号 Red
nextInt()	64 + 回车	整数记号64，剩余行内容仍在

片段：放在 main 中，单独运行

```
Scanner in = new Scanner(System.in);
String first = in.next();
String rest = in.nextLine();
System.out.println("词=<" + first + ">");
System.out.println("余下=<" + rest + ">");
```

运行核对

```
词=<Red>
余下=< Stone>
```

默认情况下，next 按空白分隔读取记号；nextLine 读取当前行剩余内容，不带行末分隔符。它们不是缩写关系。nextInt 还会按整数格式读取，错误输入可能触发异常。

本课主路线统一用 nextLine 读文字，再根据需要解析。先把边界弄清，再考虑混用其他读取方法。

04 解析：文字变数值，格式要明确

数字文字进入解析步骤



片段：放在 main 中，单独运行

```
String text = "64";
int diamonds = Integer.parseInt(text);
double x = Double.parseDouble("-12.5");
int level = Integer.parseInt(" 8 ".strip());
System.out.println(diamonds + 1);
System.out.println(x);
System.out.println(level);
```

运行核对

```
65
-12.5
8
```

parseInt 不负责猜单位，也不负责把小数四舍五入成整数。"64 blocks"、"8.5"、空字符串或超过 int 范围的值，都会解析失败。先 strip 两端空白，再解析，是清楚的一条路线。

parseDouble 还接受 NaN 和 Infinity 等特殊值。若作品要的是有限坐标，可以在解析后用 Double.isFinite(x) 检查，再核对你设定的坐标范围。

05 验证：外形通过，不等于数值合格



城门检查有顺序：先问“像不像”，再问“能不能解析”和“在不在范围内”。

片段：放在 main 中，单独运行

```
String text = "64";
boolean digitsOnly = text.matches("[0-9]+");
boolean customName = "Alex_7".matches("[A-Za-z0-9_]{3,16}");
int level = 8;
boolean levelOK = level >= 1 && level <= 30;
System.out.println(digitsOnly);
System.out.println(customName);
System.out.println(levelOK);
```

运行核对

```
true
true
true
```

matches 检查整个字符串。[0-9]+ 表示一个或多个本例的数字字符；名字表达式是某个练习服务器自定的3到16位规则，不是所有玩家名字的通用规定。

一长串数字也可能超过 int 的范围。外形检查不取代 try/catch，范围判断则必须在得到数值后进行。完整作品只要求名字清理后非空，不使用本页的自定名字规则。

06 为什么名字像被跳过了?



读整数只带走一个记号；剩下的行尾还在输入通道里。

实验：输入12、回车，再输入Alex

```
Scanner in = new Scanner(System.in);
int age = in.nextInt();
String name = in.nextLine();
System.out.println("年龄=" + age);
System.out.println("名字=<" + name + ">");
```

运行核对

年龄=12
名字=<>

当12后面紧接回车，nextInt 带走12，nextLine 随即读到当前行剩余的空内容。不是电脑不喜欢Alex，而是你读到了另一个边界。

修法之一：明确丢弃整数后的剩余整行

```
Scanner in = new Scanner(System.in);
int age = in.nextInt();
in.nextLine();
String name = in.nextLine();
System.out.println(age + " / " + name);
```

运行核对

12 / Alex

若年龄后面本来还有别的文字，上面的额外 nextLine 会一起丢弃。更统一的修法是 nextLine 读取年龄文字，再 parseInt；完整作品使用这条路线。

07 异常处理：数字太大，也要说明白

等级验证的五种实验

类型	输入	应有结果
正常	8	接受
边界	1、30	接受
越界	0、31	提示1到30
格式	abc、8.5、空内容	拒绝整数格式
过大	999999999999	捕获解析异常

片段：放在 main 中，单独运行

```
String text = "999999999999";
try {
    int level = Integer.parseInt(text);
    System.out.println("等级=" + level);
} catch (NumberFormatException e) {
    System.out.println("请输入 int 范围内的整数");
}
```

运行核对

请输入 int 范围内的整数

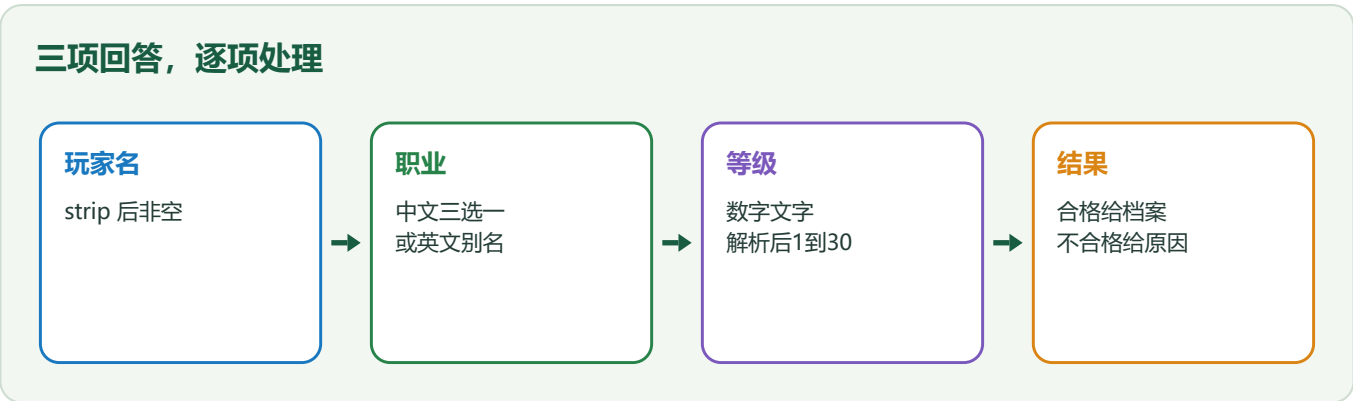
先借用 try/catch 做一层护盾：try 中尝试解析，发生对应异常时，catch 输出明确提示。它把解析失败变成具体说明，让用户知道该怎样修改输入。异常会在后续课程详细展开。

完整作品先检查等级文字只含数字，再解析，最后检查1到30的范围。三道检查各有任务。

08 三问式角色创建器：先写规则



一份角色档案的可靠程度，取决于每个字段是否经过清楚检查。



玩家名允许中文，清理两端空白后不能为空。职业接受矿工、建造师、探险家，也接受不区分大小写的 miner、builder、explorer。等级只接受数字字符组成的整数文字，解析后的值必须是1到30。

下面的完整程序借用 if 和 try/catch 把规则落实。你先看输入怎样经过各道检查；第7课再仔细拆解条件判断。程序只尝试一次，发现错误就提示并结束。

09 完整作品：角色创建器

完整程序 • RoleCreator.java

```
import java.util.Scanner;
import java.util.Locale;

public class RoleCreator {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("玩家名: ");
        String name = in.nextLine().strip();
        System.out.println("职业: 矿工 / 建造师 / 探险家");
        String job = in.nextLine().strip();
        System.out.println("初始等级: 1到30");
        String text = in.nextLine().strip();
        String alias = job.toLowerCase(Locale.ROOT);
        if (alias.equals("miner")) job = "矿工";
        if (alias.equals("builder")) job = "建造师";
        if (alias.equals("explorer")) job = "探险家";
        boolean nameOK = !name.isBlank();
        boolean jobOK = job.equals("矿工")
            || job.equals("建造师") || job.equals("探险家");
        int level = 0;
        String levelError = "";
        if (!text.matches("[0-9]+")) {
            levelError = "等级只能输入数字组成的整数";
        } else {
            try {
                level = Integer.parseInt(text);
                if (level < 1 || level > 30)
                    levelError = "等级必须在1到30之间";
            } catch (NumberFormatException e) {
                levelError = "等级数字太大, 请输入1到30";
            }
        }
        if (!nameOK) System.out.println("玩家名不能为空");
        if (!jobOK) System.out.println("职业请选择三种之一");
        if (!levelError.isEmpty()) System.out.println(levelError);
        if (nameOK && jobOK && levelError.isEmpty()) {
            System.out.println("[角色] " + name + " | " + job
                + " | Lv." + level);
        }
    }
}
```

运行核对

玩家名:
职业: 矿工 / 建造师 / 探险家
初始等级: 1到30
[角色] Alex | 矿工 | Lv.8

10 用测试表检查每一道门

角色创建器的测试矩阵

情况	名字	职业	等级	预期
正常	Alex	矿工	8	角色档案
边界	A	builder	1 / 30	接受两端边界
英文	Steve	MINER	8	职业显示矿工
空白	仅空格	矿工	8	名字不能为空
格式	Alex	矿工	8.5	等级只接受数字
过大	Alex	矿工	很多个9	数字太大提示
越界	Alex	矿工	31	1到30的范围提示
职业	Alex	骑士	8	职业选择提示

提示文字也是作品的一部分。“输入错误”太笼统，用户不知道该改哪项。参考程序分别检查名字、职业和等级；多项有问题时，会给出多项提示。

- 输入 Red Stone：名字中的内部空格保留。
- 输入等级 008：解析后显示 Lv.8。
- 输入等级 +8 或 -1：按本作品的数字外形规则拒绝。
- 换成自己的验证规则时，同步修改提示和测试表。

文件中的输出核对只列程序输出，不包含终端回显。若输入在第二行就结束，nextLine 仍需要第三项数据；完整交互必须提供三行。

11 从“能读到”走向“能解释”

第六课的能力地图



- 基础：读取名字和年龄，先预测两行输入分别被哪个变量接住。
- 进阶：读取两个坐标，解析为 double，检查有限性后计算差值。
- 创造：制作交易摘要，询问物品、数量和单价，明确验证规则。

30秒自测

① nextInt 为什么不能替代 nextLine? ② 一长串数字通过 matches，就一定能装进 int 吗? ③ 读到整数31，是否已经满足等级规则?

答案：① 读取边界和返回类型不同。② 不能，仍可能超出范围。③ 不能，还需范围验证。下一课《天黑了，if 还是 else》会让程序根据条件选择行动。

技术参考：Java SE 11 的 Scanner、Integer、Double 和 String API。官方文档：
docs.oracle.com/en/java/javase/11/docs/api/。