

第29课

保存世界

用 UTF-8 保存数据，跨过程序重启

相对路径从程序工作目录出发

Path

描述位置，不创建文件

相对路径

由启动工作目录决定

绝对路径

帮助找到真实存档

程序退出，内存里的分数也会消失。给它一份中文可读的文本存档，明确覆盖与追加，逐行检查损坏数据，用往返测试证明保存可靠。

前置背包：Map、字符串拆分和异常处理。今天让内存里的钻石不再随着程序关闭掉进岩浆。

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：先预测，再让程序回答

本章使用 JDK 21，无需预览功能。故事借用方块世界，示例运行在普通 Java 控制台；数值和规则若是课程模型，都会明确说明。

每个普通示例都是完整独立文件，按框上标注的文件名保存。各示例放在不同文件夹，避免辅助类重名。完整项目分段时，按顺序接在同一个文件中。

终端命令：中文占位符换成实际文件名与类名

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

- 预测：先写预期输出，或画清引用和数据的变化。
- 验证：运行后逐行核对，再只改一个条件。
- 解释：说清结果来自 Java 规则、课程合同还是输入数据。

“故意错误”用于认识 Bug，不能与普通示例混用。文件实验使用专门临时目录；不要拿真实存档测试覆盖和删除。

01 Path 指向存档位置

Path 只描述位置，不代表文件已经存在。相对路径的实际位置取决于程序从哪里启动。

相对路径从程序工作目录出发

Path

描述位置，不创建文件

相对路径

由启动工作目录决定

绝对路径

帮助找到真实存档

保存完成时显示绝对路径，玩家才能找到文件。路径问题经常只是大家在不同箱子里找东西。

L29Example01.java • 完整独立示例

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;
public class L29Example01 {
    public static void main(String[] args) {
        Path path = Path.of("saves", "lesson29-scores.txt");
        System.out.println(path.getFileName());
        System.out.println(path.toAbsolutePath().isAbsolute());
    }
}
```

运行核对

```
lesson29-scores.txt
true
```

亲自再打印 `path.toAbsolutePath()`，把真实地址抄进记录。此处核对只列与电脑无关的文件名和真假值；`Path.of` 不会创建目录或文件。

02 UTF 8 保存中文

编码是字符与字节之间的规则。保存和加载使用同一编码，中文名字才不会变成问号方块。

写入和读取使用同一字符集

字符

例如小马=12

UTF-8

写入和读取明确相同编码

结果核对

中文内容保持一致

不要依赖不同电脑可能不同的默认编码。课程存档统一写出 UTF-8。

L29Example02.java • 完整独立示例

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;
public class L29Example02 {
    public static void main(String[] args) throws IOException {
        Path folder = Files.createTempDirectory("java-utf8-");
        Path file = folder.resolve("scores.txt");
        try {
            Files.writeString(file, "小马=12\n", StandardCharsets.UTF_8);
            System.out.print(Files.readString(file, StandardCharsets.UTF_8));
        } finally { Files.deleteIfExists(file); Files.deleteIfExists(folder); }
    }
}
```

运行核对

小马=12

03 覆盖 追加与目录

完整计分板通常每次覆盖保存，聊天日志可能追加。选择错误会导致旧记录重复或新记录消失。

默认写入会覆盖旧内容

覆盖

文件只含新内容

追加

保留原文后添加

目录

createDirectories 创建父目录

每次写文件前先说出是覆盖还是追加。不要等玩家发现存档长到十万行。

验收对照表

任务	方式	结果
完整存档	覆盖	文件只含最新状态
事件日志	追加	新内容放到末尾
首次创建目录	createDirectories	已有目录也安全

03 覆盖 追加与目录（续）

L29Example03.java • 完整独立示例

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;
public class L29Example03 {
    public static void main(String[] args) throws IOException {
        Path folder = Files.createTempDirectory("java-write-");
        Path file = folder.resolve("log.txt");
        try {
            Files.writeString(file, "旧\n", StandardCharsets.UTF_8);
            Files.writeString(file, "新\n", StandardCharsets.UTF_8);
            Files.writeString(file, "追加\n", StandardCharsets.UTF_8,
                StandardOpenOption.APPEND);
            System.out.print(Files.readString(file, StandardCharsets.UTF_8));
        } finally { Files.deleteIfExists(file); Files.deleteIfExists(folder); }
    }
}
```

运行核对

新
追加

默认写入会创建文件或截断旧文件。APPEND 仅追加，这里文件已存在；若要支持首次创建，同时传 CREATE 和 APPEND。只在专门测试目录中做覆盖实验。

04 每条记录占一行

文本存档先从可读格式开始。一行一条记录，等号左边是玩家名，右边是整数分数。

简单格式让人和程序都能检查

名字

非空，不含等号和换行

分数

非负 int 整数

一行记录

名字=分数

存档名字不含等号、CR、LF或首尾空白，中间空格可以保留。分数为非负 int；拆成两段后仍要验证名字与数字。

L29Example04.java • 完整独立示例

```
public class L29Example04 {
    static boolean validName(String name) {
        return name != null && !name.isBlank() && name.equals(name.strip())
            && !name.contains("=") && !name.contains("\n") && !name.contains("\r");
    }
    public static void main(String[] args) {
        System.out.println(validName("小马"));
        System.out.println(validName("A=B"));
        System.out.println(validName(" Alex "));
        String[] parts = "小马=12".split("=", 2);
        System.out.println(parts[0]);
        System.out.println(Integer.parseInt(parts[1]));
    }
}
```

运行核对

```
true
false
false
小马
12
```

本格式允许名字中间有空格，拒绝首尾空白、空名字、等号、CR与LF。分数是0到 int 上限的整数。split 的2限制拆分数量，不会替你校验第二段是否仍有等号。

05 逐行解析并拒绝坏数据

加载器不能假装文件永远完美。每一行都要检查分隔符、名字和数字。

split limit 2 保留右侧剩余内容

格式检查

分隔符与合法名字

数字检查

解析、非负、整数上限

重复名字

第一条合法记录保留

一行损坏不必放弃整个世界。记录行号并跳过坏行，其他有效玩家仍可加载。

重复名字保留第一条合法记录并报告行号。本章为了集中看解析规则，把读写方法放在同一个完整文件中；复制全部代码段，不能只复制 main。

L29Example05.java • 第1/3段，顺序接在同一文件

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;

public class L29Example05 {
    static boolean validName(String name) {
        return name != null && !name.isBlank() && name.equals(name.strip())
            && !name.contains("=") && !name.contains("\n") && !name.contains("\r");
    }

    static void save(Path path, Map<String, Integer> scores) throws IOException {
        if (path == null || scores == null)
            throw new IllegalArgumentException("路径和计分板不能为null");
        List<String> lines = new ArrayList<>();
        for (var entry : scores.entrySet()) {
            if (!validName(entry.getKey()) || entry.getValue() == null
                || entry.getValue() < 0)
                throw new IllegalArgumentException("存档记录不合法");
        }
        for (var entry : new TreeMap<>(scores).entrySet())
            lines.add(entry.getKey() + "=" + entry.getValue());
        Path parent = path.getParent();
        if (parent != null) Files.createDirectories(parent);
        Files.write(path, lines, StandardCharsets.UTF_8);
    }

    static Map<String, Integer> load(Path path) throws IOException {
        if (path == null) throw new IllegalArgumentException("路径不能为null");
        List<String> lines;
        try { lines = Files.readAllLines(path, StandardCharsets.UTF_8); }
```

05 逐行解析并拒绝坏数据（续）

L29Example05.java • 第2/3段，顺序接在同一文件

```
catch (NoSuchFileException missing) { return new HashMap<>(); }
Map<String, Integer> scores = new HashMap<>();
int lineNumber = 0;
for (String line : lines) {
    lineNumber++;
    String[] parts = line.split("=", 2);
    if (parts.length != 2 || !validName(parts[0])) {
        System.out.println("跳过坏行=" + lineNumber);
        continue;
    }
    try {
        int score = Integer.parseInt(parts[1].strip());
        if (score < 0) throw new NumberFormatException("负分数");
        if (scores.containsKey(parts[0])) {
            System.out.println("跳过重复名字=" + lineNumber);
            continue;
        }
        scores.put(parts[0], score);
    } catch (NumberFormatException error) {
        System.out.println("跳过坏分数=" + lineNumber);
    }
}
return scores;
}

public static void main(String[] args) throws IOException {
    Path folder = Files.createTempDirectory("java-load-");
    Path file = folder.resolve("scores.txt");
    try {
```

L29Example05.java • 第3/3段，顺序接在同一文件

```
Files.write(file, List.of("Alex=12", "bad line", "Mia=很多",
    "Alex=99", "小马=8", "Zero=-1"), StandardCharsets.UTF_8);
Map<String, Integer> loaded = load(file);
System.out.println(loaded.get("Alex"));
System.out.println(loaded.get("小马"));
System.out.println(loaded.size());
} finally { Files.deleteIfExists(file); Files.deleteIfExists(folder); }
}
```

运行核对

跳过坏行=2
 跳过坏分数=3
 跳过重复名字=4
 跳过坏分数=6
 12
 8
 2

06 缺失文件与 IOException

存档不存在可能只是第一次运行，不一定是错误。权限不足或磁盘故障才是无法完成读写。

首次启动返回空计分板

文件缺失

返回空计分板

其他 IOException

继续传播并说明路径

坏行

报行号后跳过

IOException 若无法恢复，就显示路径和原因。不要返回空数据却告诉玩家加载成功。

L29Example06.java • 第1/2段，顺序接在同一文件

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;

public class L29Example06 {
    static boolean validName(String name) {
        return name != null && !name.isBlank() && name.equals(name.strip())
            && !name.contains("=") && !name.contains("\n") && !name.contains("\r");
    }

    static void save(Path path, Map<String, Integer> scores) throws IOException {
        if (path == null || scores == null)
            throw new IllegalArgumentException("路径和计分板不能为null");
        List<String> lines = new ArrayList<>();
        for (var entry : scores.entrySet()) {
            if (!validName(entry.getKey()) || entry.getValue() == null
                || entry.getValue() < 0)
                throw new IllegalArgumentException("存档记录不合法");
        }
        for (var entry : new TreeMap<>(scores).entrySet())
            lines.add(entry.getKey() + "=" + entry.getValue());
        Path parent = path.getParent();
        if (parent != null) Files.createDirectories(parent);
        Files.write(path, lines, StandardCharsets.UTF_8);
    }

    static Map<String, Integer> load(Path path) throws IOException {
        if (path == null) throw new IllegalArgumentException("路径不能为null");
        List<String> lines;
        try { lines = Files.readAllLines(path, StandardCharsets.UTF_8); }
    }
}
```

06 缺失文件与 IOException (续)

L29Example06.java • 第2/2段, 顺序接在同一文件

```

        catch (NoSuchFileException missing) { return new HashMap<>(); }
        Map<String, Integer> scores = new HashMap<>();
        int lineNumber = 0;
        for (String line : lines) {
            lineNumber++;
            String[] parts = line.split("=", 2);
            if (parts.length != 2 || !validName(parts[0])) {
                System.out.println("跳过坏行=" + lineNumber);
                continue;
            }
            try {
                int score = Integer.parseInt(parts[1].strip());
                if (score < 0) throw new NumberFormatException("负分数");
                if (scores.containsKey(parts[0])) {
                    System.out.println("跳过重复名字=" + lineNumber);
                    continue;
                }
                scores.put(parts[0], score);
            } catch (NumberFormatException error) {
                System.out.println("跳过坏分数=" + lineNumber);
            }
        }
        return scores;
    }

    public static void main(String[] args) throws IOException {
        Path folder = Files.createTempDirectory("java-missing-");
        try { System.out.println(load(folder.resolve("missing.txt")).size()); }
        finally { Files.deleteIfExists(folder); }
    }
}

```

运行核对

0

只有 `NoSuchFileException` 被视为初次启动的缺失存档。权限、目录当文件、编码损坏等其他 `IOException` 会向上传播；不能伪装成成功加载的空世界。

07 往返测试验证存档

文件功能最重要的测试是 round trip：准备数据，保存，加载，再比较内容。



自动测试使用专门临时目录，结束后删除测试文件，不要碰玩家真实世界。

验收对照表

场景	操作	预期
正常	保存两名玩家再加载	Map内容相同
边界	保存空Map	加载为空
字符	玩家名含中文	UTF-8保持不变
错误	坏数字行	提示并跳过
缺失	读取不存在文件	返回空Map

07 往返测试验证存档（续）

L29Example07.java • 第1/3段，顺序接在同一文件

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;
public class L29Example07 {
    static boolean validName(String name) {
        return name != null && !name.isBlank() && name.equals(name.strip())
            && !name.contains("=") && !name.contains("\n") && !name.contains("\r");
    }
    static void save(Path path, Map<String, Integer> scores) throws IOException {
        if (path == null || scores == null)
            throw new IllegalArgumentException("路径和计分板不能为null");
        List<String> lines = new ArrayList<>();
        for (var entry : scores.entrySet()) {
            if (!validName(entry.getKey()) || entry.getValue() == null
                || entry.getValue() < 0)
                throw new IllegalArgumentException("存档记录不合法");
        }
        for (var entry : new TreeMap<>(scores).entrySet())
            lines.add(entry.getKey() + "=" + entry.getValue());
        Path parent = path.getParent();
        if (parent != null) Files.createDirectories(parent);
        Files.write(path, lines, StandardCharsets.UTF_8);
    }
    static Map<String, Integer> load(Path path) throws IOException {
        if (path == null) throw new IllegalArgumentException("路径不能为null");
        List<String> lines;
        try { lines = Files.readAllLines(path, StandardCharsets.UTF_8); }
```

07 往返测试验证存档（续）

L29Example07.java • 第2/3段，顺序接在同一文件

```
catch (NoSuchFileException missing) { return new HashMap<>(); }
Map<String, Integer> scores = new HashMap<>();
int lineNumber = 0;
for (String line : lines) {
    lineNumber++;
    String[] parts = line.split("=", 2);
    if (parts.length != 2 || !validName(parts[0])) {
        System.out.println("跳过坏行=" + lineNumber);
        continue;
    }
    try {
        int score = Integer.parseInt(parts[1].strip());
        if (score < 0) throw new NumberFormatException("负分数");
        if (scores.containsKey(parts[0])) {
            System.out.println("跳过重复名字=" + lineNumber);
            continue;
        }
        scores.put(parts[0], score);
    } catch (NumberFormatException error) {
        System.out.println("跳过坏分数=" + lineNumber);
    }
}
return scores;
}

public static void main(String[] args) throws IOException {
    Path folder = Files.createTempDirectory("java-roundtrip-");
    Path file = folder.resolve("scores.txt");
    Map<String, Integer> original = Map.of("小马", 12, "Alex", 0);
```

L29Example07.java • 第3/3段，顺序接在同一文件

```
try {
    save(file, original);
    System.out.println(original.equals(load(file)));
    save(file, Map.of());
    System.out.println(load(file).isEmpty());
} finally { Files.deleteIfExists(file); Files.deleteIfExists(folder); }
}
```

运行核对

```
true
true
```

08 挑战 蓝图计分板存档

挑战 程序叫 WorldSave。它把 Map 保存为 UTF-8 文本，再加载并验证两名玩家和总数量。

按键排序保存 逐行安全加载

稳定保存

TreeMap 按名字排序

安全入口

先验证全部记录

已有存档

主程序只加载

save 方法明确覆盖文件；主程序第一次创建演示档，后续只加载已有文件。按键排序便于查看，不代表已经实现备份或原子写入。

验收顺序

- 保存Alex=12和Steve=8。
- 文件使用UTF-8且每人一行。
- 加载后按名字查询得到相同分数。
- 缺失文件返回空Map。
- 坏行打印提示并继续读取。

09 世界存档程序

新建 WorldSave.java，使用 Java 21 编译运行。

完整保存与加载流程

首次运行

创建演示文件

再次运行

不覆盖已有数据

失败显示

路径与异常类型

手动在文件末尾加入bad line和Mia=很多，再加载；应提示两条坏行，Alex与Steve仍然存在。

WorldSave.java • 第1/3段，顺序接在同一文件

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;

public class WorldSave {
    static boolean validName(String name) {
        return name != null && !name.isBlank() && name.equals(name.strip())
            && !name.contains("=") && !name.contains("\n") && !name.contains("\r");
    }
    static void save(Path path, Map<String, Integer> scores) throws IOException {
        if (path == null || scores == null)
            throw new IllegalArgumentException("路径和计分板不能为null");
        List<String> lines = new ArrayList<>();
        for (var entry : scores.entrySet()) {
            if (!validName(entry.getKey()) || entry.getValue() == null
                || entry.getValue() < 0)
                throw new IllegalArgumentException("存档记录不合法");
        }
        for (var entry : new TreeMap<>(scores).entrySet())
            lines.add(entry.getKey() + "=" + entry.getValue());
        Path parent = path.getParent();
        if (parent != null) Files.createDirectories(parent);
        Files.write(path, lines, StandardCharsets.UTF_8);
    }
    static Map<String, Integer> load(Path path) throws IOException {
        if (path == null) throw new IllegalArgumentException("路径不能为null");
        List<String> lines;
        try { lines = Files.readAllLines(path, StandardCharsets.UTF_8); }
    }
```

09 世界存档程序（续）

WorldSave.java • 第2/3段，顺序接在同一文件

```
catch (NoSuchFileException missing) { return new HashMap<>(); }
Map<String, Integer> scores = new HashMap<>();
int lineNumber = 0;
for (String line : lines) {
    lineNumber++;
    String[] parts = line.split("=", 2);
    if (parts.length != 2 || !validName(parts[0])) {
        System.out.println("跳过坏行=" + lineNumber);
        continue;
    }
    try {
        int score = Integer.parseInt(parts[1].strip());
        if (score < 0) throw new NumberFormatException("负分数");
        if (scores.containsKey(parts[0])) {
            System.out.println("跳过重复名字=" + lineNumber);
            continue;
        }
        scores.put(parts[0], score);
    } catch (NumberFormatException error) {
        System.out.println("跳过坏分数=" + lineNumber);
    }
}
return scores;
}

public static void main(String[] args) {
    Path path = args.length == 0
        ? Path.of("saves", "lesson29-scores.txt") : Path.of(args[0]);
    try {
```

09 世界存档程序（续）

WorldSave.java • 第3/3段，顺序接在同一文件

```
        if (Files.notExists(path)) {
            save(path, Map.of("Alex", 12, "Steve", 8));
            System.out.println("首次运行：已建立演示存档");
        } else {
            System.out.println("已有存档：本次只加载");
        }
        Map<String, Integer> loaded = load(path);
        System.out.println("文件=" + path.getFileName());
        System.out.println("Alex=" + loaded.get("Alex"));
        System.out.println("Steve=" + loaded.get("Steve"));
        System.out.println("玩家数=" + loaded.size());
    } catch (IOException error) {
        System.out.println("读写失败：" + path.toAbsolutePath());
        System.out.println(error.getClass().getSimpleName());
    }
}
```

运行核对

首次运行：已建立演示存档
文件=lesson29-scores.txt
Alex=12
Steve=8
玩家数=2

以上是专用目录里第一次运行的输出。再次运行，第一行改为“已有存档：本次只加载”；程序不会覆盖手改数据。save 本身明确执行覆盖，不是自动备份、原子写入或多人服务器存档系统。

10 拓展任务与通关复盘

作业必须标出文件实际路径、编码、覆盖或追加方式，并提交一次往返测试。

三档任务继续扩建存档系统

编码证据

明确 UTF-8

路径证据

记录绝对地址

读写证据

往返测试和坏行测试

第30课把挖矿记录送上数据流水线，用 filter、map、sorted 和 sum 生成统计简报。

三档作业

- 木镐：把一句中文告示保存为UTF-8，再读取显示。
- 铁镐：保存多个玩家分数，加载时跳过坏行。
- 钻石镐：在首行加入版本号，版本不支持时拒绝加载并给出提示。

验收对照表

自评	能	还要练
我会使用Path和Files	☐	☐
我能明确使用UTF-8	☐	☐
我会做往返测试	☐	☐

自己的验收记录

漂亮的输出是起点，能解释边界才算掌握。把正常、边界与错误各选一个案例，运行前写预期，运行后留下证据。

为判断留下依据

类别	你选择的操作	核对重点
正常	一个常用输入或一次成功动作	结果是否逐行一致
边界	范围端点与相邻值	有没有多一步、少一项
错误	非法数量、缺失对象或坏数据	是否明确拒绝，状态是否保持

- Bug 记录：触发条件、实际现象、原因、最小修复。
- 图示记录：为每条箭头说明起点、终点和含义。
- 改编记录：只改一个规则，同时修改预期和测试。

技术核对：Java SE 21 语言规范与标准库。类、接口、类型与表达式规则见语言规范；集合、Random、Files 与 Stream 的行为见对应 API。
<https://docs.oracle.com/javase/specs/jls/se21/html/index.html>
<https://docs.oracle.com/en/java/javase/21/docs/api/index.html>