Mini-OJ ·Java 考试速成

拟合试卷·只看考试向

winbeau ·**Mini-OJ** 教学项目

古法手撸·教学优先·2026-06

考试向（拟合试卷，先上手）

工程向（项目深度，选学）

★★★ 考试相关度（历年真题）

目录

Mini-OJ ·Java 考试速成

> 本册只覆盖考试向 (每个里程碑的「第一步」): 按真题难度/编码风格, 用我们 Mini-OJ 的功能练考点。工程化部分 (反射工厂、外部 C++ 判题机、泛型架构、数据库工程、MVC 解耦) 不在本册——要做项目深度看《完整版》。题型 (新疆大学《面向对象程序设计 B》): 单选 + 判断 + 读程序 + 程序填空 + 编程, 各约 20 分。全程命令行 `javac` / `java`, 不用 make。

考点速查 (★ = 历年真题强相关)

模块 (章)	★	真题题型	本册小节
类与对象/构造/封装/static(Ch4)	★★★	编程大题、读程序、选择	M2
继承/重写/多态/抽象类/接口/异常 (Ch5-7)	★★★	编程大题、读程序、选择	M3
Swing GUI + 事件 (Ch9)	★★★	编程 + 填空双大题、选择	M5c
集合 ArrayList/LinkedList/HashMap(Ch15)	★★	读程序、填空	M5a
JDBC 查询/更新 (Ch11)	★★	填空、原题	M5b
多线程 Thread/join/synchronized(Ch12)	★★	填空、选择	M6a

模块 (章)	★	真题题型	本册小节
String/StringBuffer/Random/Math(Ch8)	★★	读程序、选择	M4
数组/运算符/流程 (Ch2-3)	★	选择、读程序	M1
异常 try-catch(Ch7)	★	选择、判断	M3
文件 File/流/序列化 (Ch10)	★	少量选择/判断	(并入各章)
环境/编译运行 (Ch1)	☆	少量选择	M0

> 复习顺序建议: 先把编程大题练熟 (M2 类、M3 抽象类 + 多态、M5c 窗口——必手写), 再过填空/读程序 (M5a 集合、M5b JDBC、M6a 线程、M4 String), 选择/判断靠平时概念。

M0 ·环境与编译运行 (Ch1)☆

```
sudo apt install openjdk-21-jdk      # 装 JDK
nvim Main.java                       # 写一个打印 "Mini-OJ ready" 的 Main 类
javac Main.java                      # 编译 -> Main.class
java Main                            # 运行
javap -c Main                        # 反编译看字节码 (理解 .class)
```

考点:.java 编译成几个 .class(每个类一个)、运行用 java 类名 (不带 .class)、平台无关。

M1 ·数组与流程: 单题判题器 (Ch2-3)★

考点: 数组遍历、if/switch、for/while、++i、关系/逻辑运算。

Main.java(单文件、纯 static)

```
public class Main {
    static int solve(int a, int b) { return a + b; }          // 模拟用户解法
    public static void main(String[] args) {
        int[][] in = { {1,2}, {10,20}, {0,0} };               // 内置用例
        int[] expected = { 3, 30, 0 };
        int passed = 0;
        for (int i = 0; i < in.length; i++) {                 // 数组遍历
            int actual = solve(in[i][0], in[i][1]);
            if (actual == expected[i]) passed++;
            else System.out.println("case#" + i + " 期望 " + expected[i] + " 实际 " +
             ↪  actual);
        }
```

```
        System.out.println("A+B: " + (passed == in.length ? "AC" : "WA") + " " + passed
        ↪  + "/" + in.length);
    }
}
```

```
javac Main.java && java Main      # A+B: AC 3/3
```

M2 ·类与对象 (Ch4)★★★

考点: 属性 + 构造方法 + get/set、封装 (private)、static、this、方法重载。编程大题常考「定义一个类」。

src/oj/core/TestCase.java

```
package oj.core;
public class TestCase {
    private String input, expected;
    public TestCase(String input, String expected) { this.input = input; this.expected
    ↪  = expected; }
    public String getInput()    { return input; }
    public String getExpected() { return expected; }
}
```

src/oj/core/Problem.java

```
package oj.core;
public class Problem {
    private int id; private String title; private TestCase[] cases;
    public Problem(int id, String title, TestCase[] cases) { this.id = id; this.title
    ↪  = title; this.cases = cases; }
    public int getId()           { return id; }
    public String getTitle()     { return title; }
    public TestCase[] getCases() { return cases; }
}
```

src/oj/core/JudgeResult.java

```
package oj.core;
public class JudgeResult {
    private String status; private int passed, total;
    public JudgeResult(String status, int passed, int total) { this.status = status;
    ↪  this.passed = passed; this.total = total; }
    @Override public String toString() { return status + " " + passed + "/" + total; }
```

```
}
```

src/oj/Demo.java

```
package oj;
import oj.core.*;
public class Demo {
    public static void main(String[] args) {
        Problem p = new Problem(1, "A+B", new TestCase[]{ new TestCase("1 2","3"), new
 ↪  TestCase("10 20","30") });
        System.out.println(p.getTitle() + " 有 " + p.getCases().length + " 组用例");
        System.out.println(new JudgeResult("AC", 2, 2));
    }
}
```

```
javac -d build src/oj/core/*.java src/oj/Demo.java
java -cp build oj.Demo            # A+B 有 2 组用例 / AC 2/2
```

M3 ·继承/多态/抽象/接口/异常 (Ch5-7)★★★

考点: 抽象基类-> 子类重写、多态、接口、try-catch、成员变量隐藏。编程大题常考「抽象类 + 子类重写 + 多态」。

src/oj/judge/Solution.java(接口, 可用 Lambda 实现)

```
package oj.judge;
public interface Solution { String solve(String input); }
```

src/oj/judge/SimpleJudge.java

```
package oj.judge;
import oj.core.*;
public class SimpleJudge {
    public JudgeResult judge(Problem p, Solution s) {
        TestCase[] cs = p.getCases();
        int passed = 0;
        try {
            for (TestCase c : cs)
                if (s.solve(c.getInput()).trim().equals(c.getExpected().trim()))
                 ↪  passed++;
        } catch (RuntimeException e) {
            return new JudgeResult("RE", passed, cs.length);   // 崩溃 -> RE
```

```
        }
        return new JudgeResult(passed == cs.length ? "AC" : "WA", passed, cs.length);
    }
}
```

src/oj/Demo3.java

```
package oj;
import oj.core.*;
import oj.judge.*;
public class Demo3 {
    public static void main(String[] args) {
        Problem p = new Problem(1, "A+B", new TestCase[]{ new TestCase("1 2","3"), new
 ↪  TestCase("10 20","30") });
        Solution good = in -> { String[] t = in.split("\\s+"); return "" +
 ↪  (Integer.parseInt(t[0]) + Integer.parseInt(t[1])); };
        Solution bad  = in -> { throw new RuntimeException("boom"); };
        System.out.println(new SimpleJudge().judge(p, good));   // AC 2/2
        System.out.println(new SimpleJudge().judge(p, bad));    // RE 0/2
    }
}
```

```
javac -d build src/oj/core/*.java src/oj/judge/*.java src/oj/Demo3.java
java -cp build oj.Demo3
```

抽象类写法 (编程题模板):abstract class Shape{ abstract double area(); } -> class Circle extends Shape{ double area(){...} }, 父类引用指子类对象即多态。

M4 ·String 与类库 (Ch8)★★

考点:String 的 == vs equals(字符串常量池)、trim/split、StringBuffer/StringBuilder、Random/Math。

src/oj/judge/OutputChecker.java

```
package oj.judge;
public class OutputChecker {
    public static boolean same(String expected, String actual) { return
     ↪  norm(expected).equals(norm(actual)); }
    private static String norm(String s) {
        StringBuilder sb = new StringBuilder();
        for (String line : s.split("\n", -1)) sb.append(line.trim()).append('\n');
        return sb.toString().trim();
```

```
    }
}
```

```
javac -d build src/oj/judge/OutputChecker.java
# 比对忽略行尾空白:same("3\n","  3  ") -> true
```

读程序常考:"abc"=="abc" 为 true(常量池),new String("abc")==.. 为 false,equals 始终比内容。

M5a ·集合框架 (Ch15)★★

考点:ArrayList/LinkedList(有序可重复)、HashSet(去重)、HashMap(键值)、Iterator 遍历、Stream。

src/oj/io/Bank.java

```
package oj.io;
import oj.core.*;
import java.util.*;
public class Bank {
    private Map<Integer, Problem> problems = new HashMap<>();
    public void add(Problem p) { problems.put(p.getId(), p); }
    public Problem get(int id) { return problems.get(id); }
    public List<Integer> ids() { return new ArrayList<>(problems.keySet()); }
}
```

src/oj/Demo5a.java

```
package oj;
import oj.core.*;
import oj.io.Bank;
public class Demo5a {
    public static void main(String[] args) {
        Bank bank = new Bank();
        bank.add(new Problem(1, "A+B", new TestCase[]{ new TestCase("1 2","3") }));
        bank.add(new Problem(2, " 平方", new TestCase[]{ new TestCase("3","9"), new
 ↪  TestCase("4","16") }));
        for (int id : bank.ids())                                    // 遍历
            System.out.println(id + " => " + bank.get(id).getTitle());
        long n = bank.ids().stream().mapToInt(id ->
         ↪  bank.get(id).getCases().length).sum();  // Stream
        System.out.println(" 总用例: " + n);
    }
}
```

```
javac -d build src/oj/core/*.java src/oj/io/Bank.java src/oj/Demo5a.java
java -cp build oj.Demo5a
```

M5b ·JDBC(Ch11)★★

考点:DriverManager.getConnection -> Statement -> executeQuery/executeUpdate -> ResultSet.next();PreparedStatement(? 占位) 防 SQL 注入。填空常考。

src/oj/db/SimpleDb.java(需 MySQL + lib/ 里放驱动 jar)

```
package oj.db;
import java.sql.*;
import java.util.*;
public class SimpleDb {
    static final String URL = "jdbc:mysql://localhost:3306/mini_oj?serverTimezone=UTC";
    public static void main(String[] args) throws Exception {
        try (Connection c = DriverManager.getConnection(URL, "root", "root");
             Statement st = c.createStatement()) {
            st.executeUpdate("CREATE TABLE IF NOT EXISTS sub(id INT, name VARCHAR(32),
 ↪  status VARCHAR(8))");
            st.executeUpdate("INSERT INTO sub VALUES(1,'alice','AC')");
            ResultSet rs = st.executeQuery("SELECT id,name,status FROM sub");
            List<String> rows = new ArrayList<>();
            while (rs.next()) rows.add(rs.getInt("id") + " " + rs.getString("name") + "
             ↪  " + rs.getString("status"));
            rows.forEach(System.out::println);
        }
    }
}
```

```
javac -cp "lib/*" -d build src/oj/db/SimpleDb.java
java -cp "build:lib/*" oj.db.SimpleDb            # 1 alice AC
```

M5c ·Swing GUI(Ch9)★★★

考点:JFrame、布局 (FlowLayout/BorderLayout/GridLayout)、JButton/JTextField/JLabel/JComboBox、ActionListener(Lambda)、JOptionPane。编程 + 填空双大题。

src/oj/gui/SimpleOj.java

```
package oj.gui;
import javax.swing.*;
```

```
import java.awt.*;
public class SimpleOj {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {                    // 在 EDT 上建界面
            JFrame f = new JFrame("Mini-OJ 提交");
            f.setLayout(new FlowLayout());
            f.setSize(380, 150);
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JComboBox<String> prob = new JComboBox<>(new String[]{ "1 A+B", "2 平方" });
            JTextField input = new JTextField(12);
            JButton submit = new JButton(" 提交");
            JLabel result = new JLabel(" 等待提交");
            submit.addActionListener(e ->                     // Lambda 监听
                result.setText(" 已提交 [" + prob.getSelectedItem() + "] 输出 =" +
 ↪  input.getText()));
            f.add(new JLabel(" 题目")); f.add(prob);
            f.add(new JLabel(" 输出")); f.add(input);
            f.add(submit); f.add(result);
            f.setVisible(true);
        });
    }
}
```

```
javac -d build src/oj/gui/SimpleOj.java
java -cp build oj.gui.SimpleOj                    # 需图形界面
```

填空常考骨架:new JFrame() -> setLayout(new GridLayout(行, 列)) -> for(...) add(new JButton(...)) -> setVisible(true)。

M6a ·多线程 (Ch12)★★

考点:Thread/Runnable、start()(非 run())、join()、synchronized。填空常考线程类。

src/oj/Demo6a.java

```
package oj;
public class Demo6a {
    static int done = 0;
    static synchronized void finish(String who) { done++; System.out.println(who + " 判
     ↪  完, 累计 " + done); }
    public static void main(String[] args) throws InterruptedException {
```

```
        Runnable job = () -> { try { Thread.sleep(50); } catch (InterruptedException
        ↪  e) {} finish(Thread.currentThread().getName()); };
        Thread t1 = new Thread(job, "worker-1");
        Thread t2 = new Thread(job, "worker-2");
        t1.start(); t2.start();                                  // 并发 (不是 run())
        t1.join();  t2.join();                                   // 主线程等它们
        System.out.println(" 全部完成: " + done);
    }
}
```

```
javac -d build src/oj/Demo6a.java
java -cp build oj.Demo6a
```

填 空 常 考:class T extends Thread{ public T(String n){ super(n); } public void run(){...} }; 主线程 t.start(); 启动。

想看「这些功能怎么长成工业级判题系统」(反射、外部 C++ 判题机、数据库工程、多线程队列、MVC)——见完整版:https://winbeau.github.io/Mini-OJ-docs/mini-oj-tutorial.pdf