
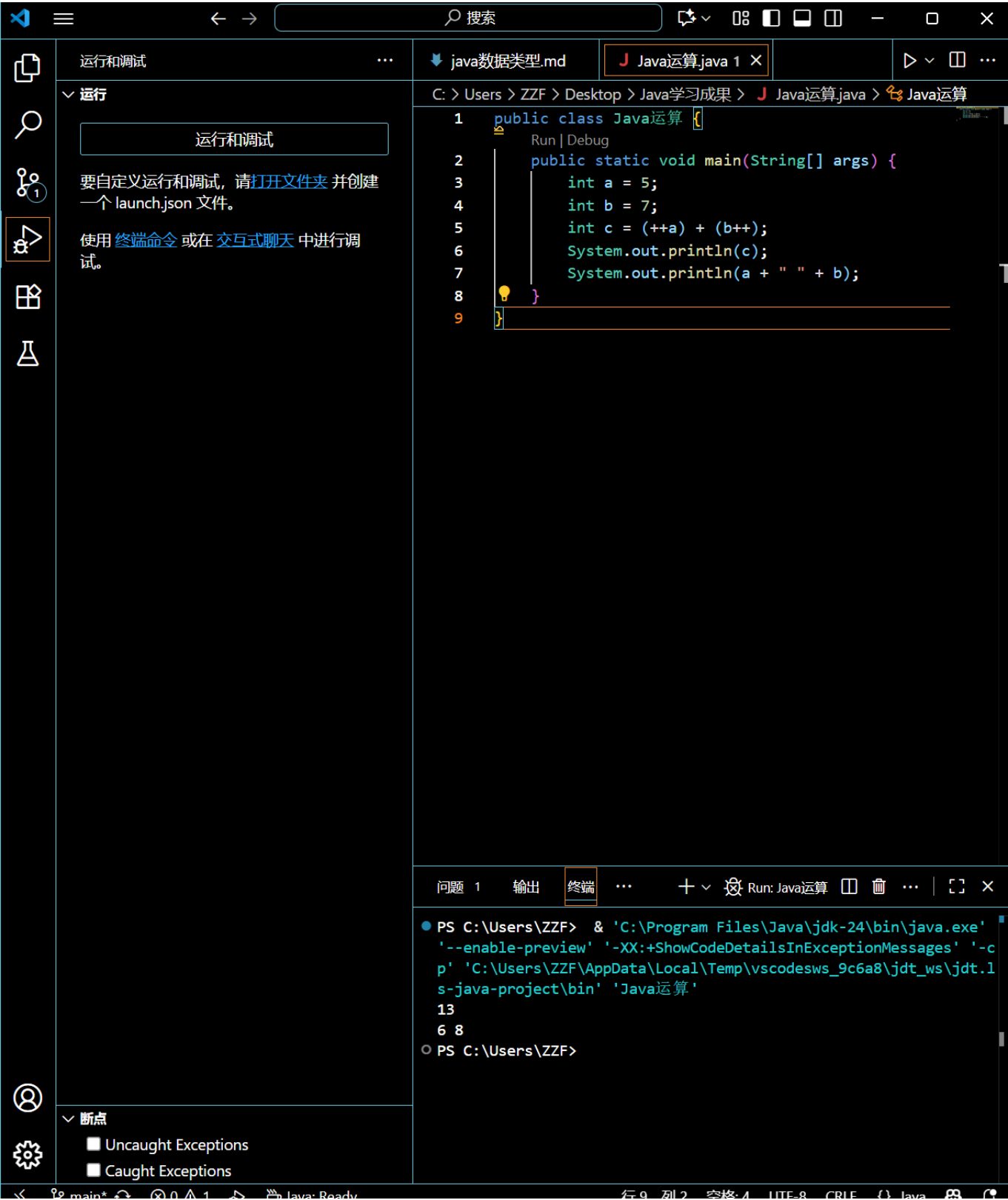
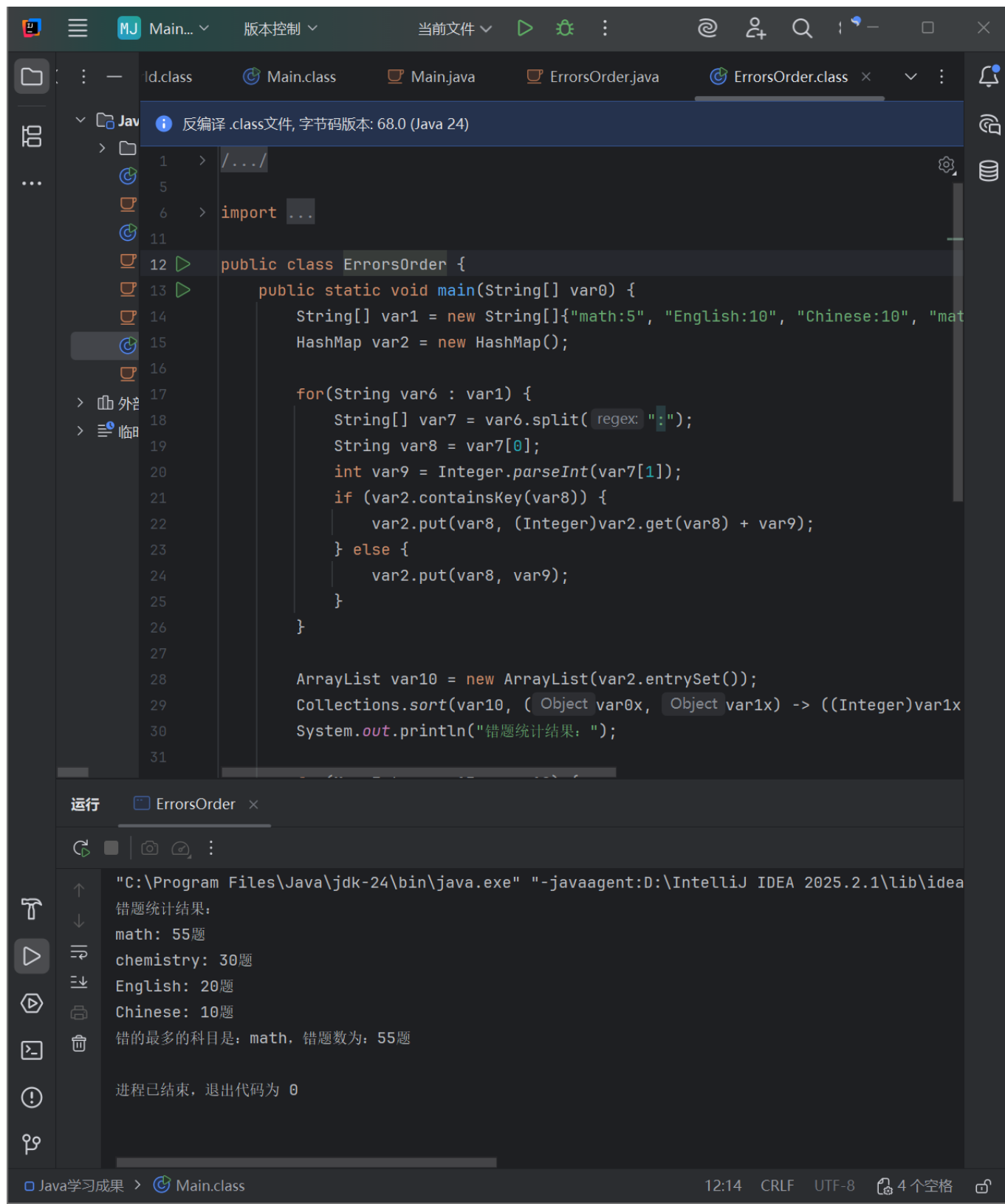


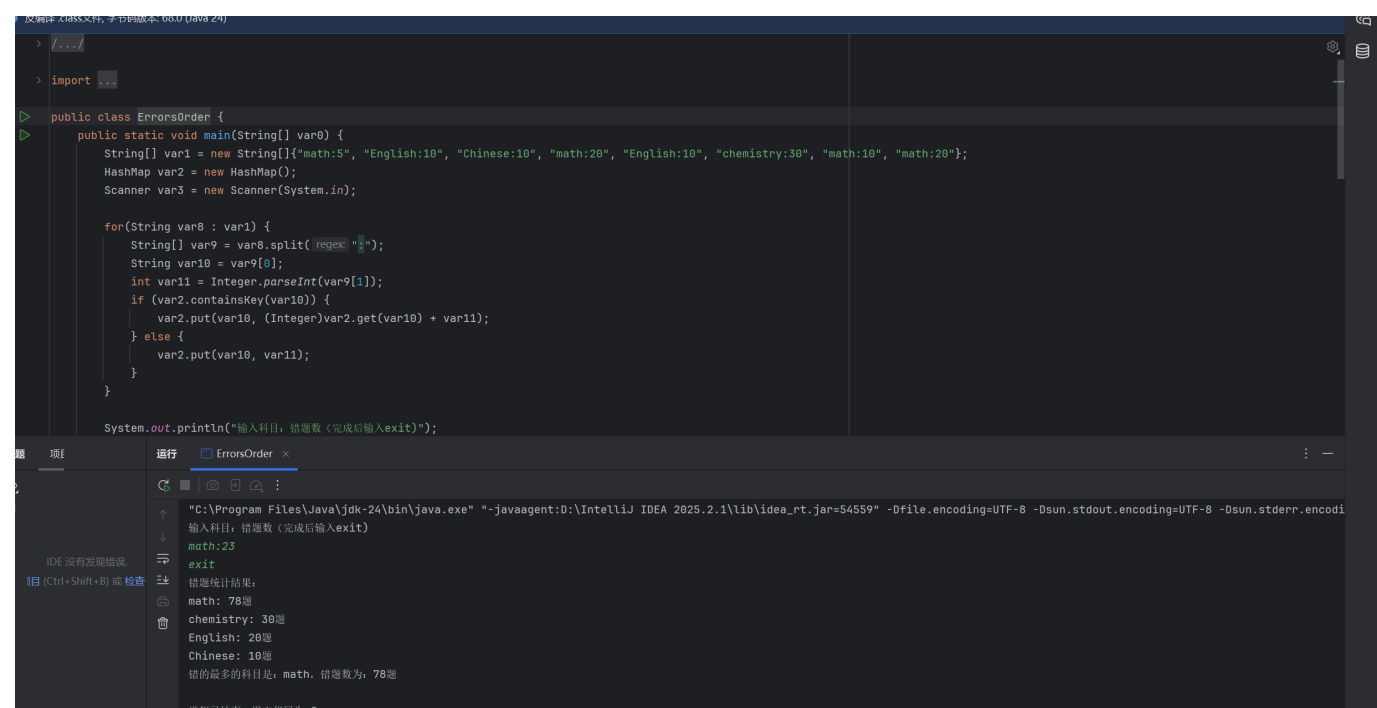
Task1 1) -整型 byte存储中等范围整数, 存储年龄等 short存储中等范围整数 int默认整型, 日常整数计算首选 long存储大范围整数 (需加 L 后缀) 如身份证号 -字符型 char存储单个字符, 本质是 Unicode 编码值 (十进制整数) -浮点型 float 单精度浮点数 (需加 f 后缀), 精度较低 double 双精度浮点数, 默认浮点类型, 精度更高 -布尔型 boolean存储逻辑值, 仅 true (真) 和 false (假) 两种取值 2) byte 占1字节 范围是-128~127 short 占2字节 范围是-32768到32767 int 占4字节 范围-2的31次方到2的31次方减一 long占8字节 范围-2的63次方到2的63次方减一 3) 隐式类型转换; b的值是52, 因为char类型的'0'在 ASCII 码中对应的十进制值是48, int类型的a是4, 在相加过程中, char类型会自动转换为int类型后再进行计算, 因此两者相加 $4+48=52$  4) 第一段代码输出false。因为x和y都是通过new Integer(18)创建的, new Integer() 会强制创建新对象, 导致x和y指向不同的内存地址, ==比较的是对象地址, 所以结果为false。第二段代码输出true。Integer.valueOf对-128到127之间的整数, 会直接从缓存池中获取对象, 而18在这个范围内, 所以z和k引用的是同一个对象, 所以==结果为true。第三段代码输出false。300超出了Integer的范围 (-128到127), 会创建新对象, 所以m和p指向不同对象, ==比较引用, 所以结果为false。 Task2 5) a取5, b取7, c取a自增1后 (6) 和b自增1之前的数 (7) 之和 (13), 输出c的取值13;从左到右执行, a+" " 得到"6 ", 再"6 "+b得到"6 8" alt text 6) int类型通常以二进制补码的形式存储 原码: 最高位为符号位 (0 表示正数, 1 表示负数), 其余位表示数值大小 补码: 正数的补码和原码相同, 负数的补码是其原码除符号位外, 其余各位取反, 然后末位加 1 float类型遵循IEEE 754 标准进行存储 符号位 (1 位): 用于表示数的正负, 0 表示正数, 1 表示负数。指数位 (8 位): 以偏移值 127 为基准, 存储实际指数加上偏移值后的结果。尾数位 (23 位): 存储小数部分, 默认最高位是 1 两个正数相加结果为负数是因为发生了溢出。在int的二进制补码运算中, 两个超出了 int 类型所能表示的范围的数相加, 会导致最高位 (符号位) 发生进位, 原本表示正数的符号位变为 1, 结果就成了负数。而对于float, 当运算结果超出了float 所能表示的最大范围时, 会得到 Infinity (正无穷), 如果是负数超出范围则得到 -Infinity (负无穷)。 Task3 1)



2)



3)



代码

```
import java.util.HashMap;
import java.util.Map;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class ErrorsOrder {
    public static void main(String[] args) {
        String[] rawDat = {"math:5", "English:10", "Chinese:10", "math:20",
            "English:10", "chemistry:30", "math:10", "math:20"};
        Map<String, Integer> map = new HashMap<>();
        Scanner sc = new Scanner(System.in);
        String input;

        for (String entry : rawDat) {
            String[] parts = entry.split(":");
            String subject = parts[0];
            int score = Integer.parseInt(parts[1]);
            if (map.containsKey(subject)) {
                map.put(subject, map.get(subject) + score);
            } else {
                map.put(subject, score);
            }
        }

        System.out.println("输入科目：错题数（完成后输入exit）");
        while (true) {
            input = sc.nextLine();
            if (input.equals("exit")) {
                break;
            }
            String[] parts = input.split(":");
            String subject = parts[0];
            int errors = Integer.parseInt(parts[1].trim());
            if (map.containsKey(subject)) {
                map.put(subject, map.get(subject) + errors);
            } else {
                map.put(subject, errors);
            }
        }
    }
}
```

4)

```

System.out.println("输入科目：错题数（完成后输入exit）");
while (true) {
    input = sc.nextLine();
    if (input.equals("exit")) {
        break;
    }
    String[] parts = input.split(":");
    String subject = parts[0];
    int errors = Integer.parseInt(parts[1].trim());
    if (map.containsKey(subject)) {
        map.put(subject, map.get(subject) + errors);
    } else {
        map.put(subject, errors);
    }
}

ArrayList<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());
Collections.sort(list, (Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) ->
    o2.getValue().compareTo(o1.getValue()));

System.out.println("错题统计结果：");
for (Map.Entry<String, Integer> entry : list) {
    System.out.println(entry.getKey() + "： " + entry.getValue() + "题");
}

if (!list.isEmpty()) {
    Map.Entry<String, Integer> maxEntry = list.get(0);
    System.out.println("错的最多的科目是： " + maxEntry.getKey() + "， 错题数为： " + maxEntry.getValue() + "题");
}

sc.close();

```

```

public class ErrorsOrder {
    private static void processData(String[] var0, Map<String, Integer> var1) {
        String var7 = var0[0];
        int var8 = Integer.parseInt(var0[1]);
        updateMap(var7, var8, var1);
    }

    private static void updateMap(String var0, int var1, Map<String, Integer> var2) {
        if (var2.containsKey(var0)) {
            var2.put(var0, (Integer)var2.get(var0) + var1);
        } else {
            var2.put(var0, var1);
        }
    }

    private static void displayStatistics(Map<String, Integer> var0) {
        ArrayList var1 = new ArrayList(var0.entrySet());
        Collections.sort(var1, (Object var0x, Object var1x) -> ((Integer)var1x.getValue()).compareTo((Integer)var0x.getValue()));

        for (Map.Entry var3 : var1) {
            PrintStream var10000 = System.out;
            String var10001 = (String)var3.getKey();
            var10000.println(var10001 + "： " + String.valueOf(var3.getValue()) + "题");
        }

        if (!var1.isEmpty()) {
            Map.Entry var4 = (Map.Entry)var1.get(0);
            PrintStream var5 = System.out;
            String var6 = (String)var4.getKey();
            var5.println("错的最多的科目是： " + var6 + "， 错题数为： " + String.valueOf(var4.getValue()) + "题");
        }
    }
}

```