《移动应用软件开发》实验报告

年级、专业、班级	2021 级　XXX 班	姓名	XXX	学号	XXXX
实验题目	扑克牌计算 24 点 APP				
实验时间	2023.10.22	实验地点	DS1401		
学年学期	2023-2024(1)	实验性质	□验证性　■设计性　□综合性		

一、实验目的

1. 本次实验的目的是掌握开发工具的基本操作，了解所开发 APP 项目的基本结构。掌握常用组件和布局的使用。开发平台 HarmonyOS，Android，iOS，**原生开发**，三选一。
2. 设计完成一个 APP 开发：扑克牌计算 24 点。
3. 抄袭计 0 分。

二、实验项目内容

已知一副扑克牌有 54 张，去除大王和小王，剩余 52 张。在其中抽取 4 张牌，利用加减乘除进行计算得到 24，除法必须能够除尽。编写程序从一副扑克牌，选择 4 张，进行计算是否能得到 24。如果可以，排序列出可能的计算表达式，可能有多种计算形式。

评分点：（1）在界面上显示出 52 张扑克牌。（2）通过点击的方式选出 4 张扑克牌并放置在界面某一个地方，位置自己确定。（3）如果可以计算出 24，排序列出可能的计算方式，并显示在界面上，如果不能算出，请提示。（4）APP 界面自行设计，至少包含两个界面。（5）界面美观度，功能完整度，扩展性功能，程序稳定性。

提交：（1）本实验报告，（2）源代码压缩文件 zip，（3）软件演示的 MP4 视频，视频大小不超过 20M，视频请在**搜狗浏览器**测试能否正常播放。注意源代码加注释。注意文件名称的规范性。文件名：组号-学号姓名 1.doc，组号-学号姓名 1.zip, 组号-学号姓名 1.mp4。**三个文件分别提交**。

三、实验过程或算法（写明设计思想、程序的结构、功能关系图、类的说明和类之间的关系图、程序主要执行流程图，最后是核心源代码，截图等）

1　设计思想：

1.1 算法设计部分：

本实验涉及到的核心逻辑是设计算法实现给定四张对应特定

数值的卡牌，计算出所有结果为 24 的表达式。

一种很容易想到的方式是用多重循环嵌套，每层循环确定第一个操作数，第二个操作数……，再确定第一个操作符，第二个操作符……，最后很容易能得到一个特定表达式，再判断运算结果是否为 24。但这种方式写出的代码显然不太“优雅”，并且有一个致命的问题是这种方法做出来的结果会少掉所有形如“(a op1 b) op3 (c op2 d)”这样的结果。

于是我们换一种思路，仔细分析一下 24 点游戏，它有几个关键的特征，一是只有四个操作数，二是只有三个操作符，所以他的组合方式其实是很有限的，写出来也只有五种

1.((a op1 b) op2 c) op3 d

2.(a op2 (b op1 c)) op3 d

3.a op3 ((b op1 c) op2 d)

4.a op3 (b op2 (c op1 d))

5.(a op1 b) op3 (c op2 d)

再仔细观察一下可以发现其实只需要考虑 1,5 即可，即 1-4 其实都是等价的，比如我们可以将他们的逆波兰（后缀）表达式写出来：

1.a b op1 c op2 d op3

2.b c op1 a op2 d op3

3.b c op1 d op2 a op3

4.c d op1 b op2 a op2

5.a b op1 c d op2 op3

不难发现 1-4 的结构都是一样的，换言之，2-4 都是 1 的某种排列能得出的结果，所以我们不妨就借助逆波兰表达式来解决这个问题，因为相比中缀表达式，后缀表达式更容易计算结果（借助栈）。

于是我们得出解决问题的大致思路：

1. 得到用户选择的四张扑克牌对应数字的全排列（可以去重以提高运算效率）

2. 将数字的某个特定排列与三个操作符得到的所有排列一一组合得到一个最终排列方案

3. 根据这个排列生成形如“a b op1 c op2 d op3”，“a b op1 c d op2 op3”的两种逆波兰表达式

4. 计算逆波兰表达式的值，筛选出结果为 24 的表达式添加进

结果。

5. 将结果中的逆波兰表达式转化为中缀表达式（方便显示）作为最终结果。

1.2 界面设计部分：

整个 app 主要有两个界面，主页面和结果展示页面：

1.2.1　主页面：

主页面主要包含以下板块：

1.卡牌展示区：展示所有 52 张卡牌，并且当点击卡牌时将卡牌加入已选择卡牌区。由于每张卡牌属于重复视图，所以主要采用了 GridView 加自定义 Adapter 来实现卡牌展示。并且由于 52 张卡牌实在太多，会导致页面过于冗长，于是还采用了类似蜘蛛卡牌游戏那样的重叠显示，大大节省了页面空间。

2. 已选择卡牌展示区： 展示用户已选择的所有卡牌（最多四张），同样采用 GridView 实现，并且支持用户再次点击卡牌以实现取消选择。

3. 功能按钮区： 主要有两个按钮，“清空选择”按钮用于将卡牌展示区已选择的卡牌全部清空，“计算”按钮用于调用核心算法程序计算用户选择的四张卡牌所能得到的所有结果并跳转至结果展示页。

1.2.2　结果展示页面

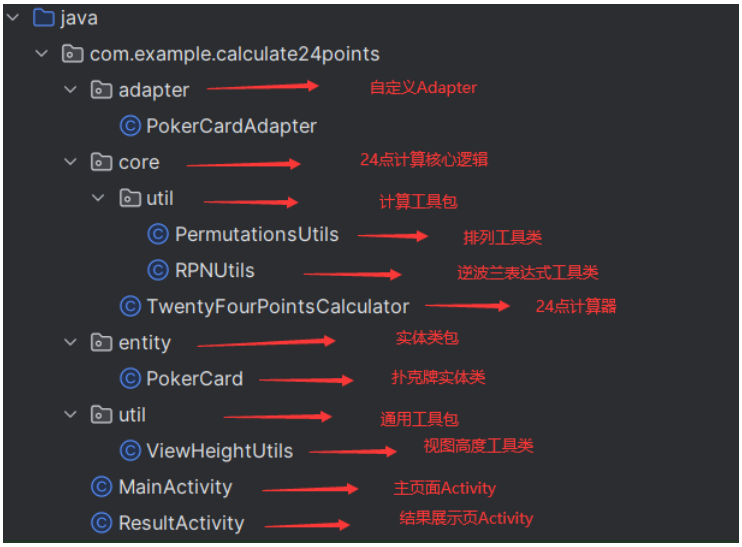
结果展示页面主要包含以下板块：

1. 结果展示区：展示选定卡牌等于 24 点的所有表达式，采用 ListView 实现，同时显示结果总数。

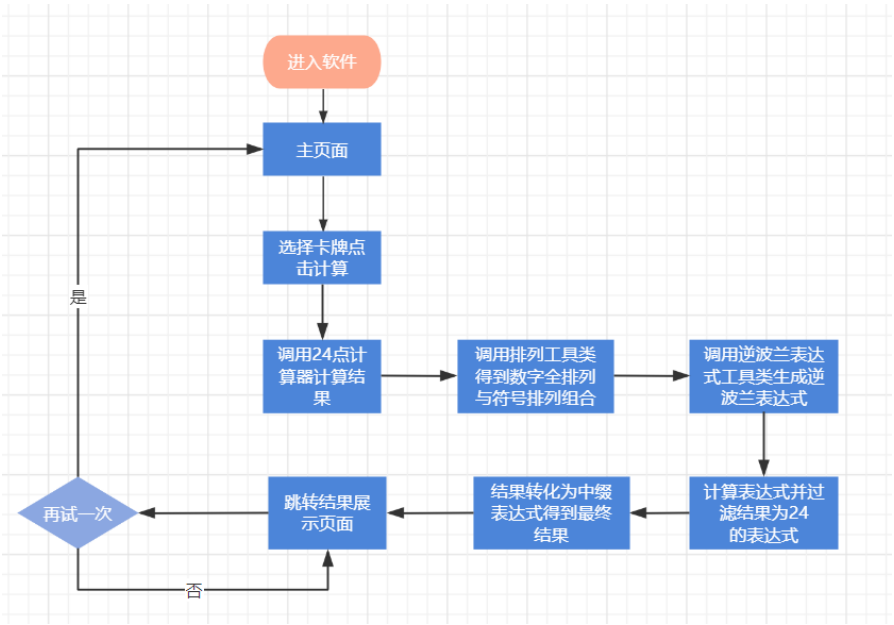
2. 再来一次： 用户通过点击再来一次按钮可以再次跳转到主页面再次计算。

2　程序结构：

程序整体结构如下：



3　程序主要执行流程图：



4　核心源代码：

4.1 24 点计算核心逻辑：

```java
/**
 * 计算24点游戏的结果
 *
 * @param numbers
 * @return
 */
1 个用法  Oreki
public static List<String> calculate(String[] numbers) {
    // 生成数字的所有全排列
    List<String[]> numberPermutations = PermutationsUtils.permuteUnique(numbers);
    // 使用并行流处理，加快运算速度
    return numberPermutations.parallelStream() Stream<String[]>
            // 生成所有可能的逆波兰表达式并合并为一个流
            .flatMap(numberPermutation ->
                    OPERATOR_PERMUTATIONS
                            .stream()
                            // 组合数字的某种排列和操作符的某种排列，生成逆波兰表达式
                            .flatMap(operatorPermutation -> Arrays.stream(RPNUtils.generateRPNs(numberPermutation, operatorPermutation)))
            )
            // 过滤出等于24的逆波兰表达式
            .filter(RPNUtils::checkRPN)
            // 将逆波兰表达式转换为中缀表达式，便于展示结果
            .map(RPNUtils::rpnToInfix) Stream<String>
            .collect(Collectors.toList());
}
```

4.2 计算逆波兰表达式算法：

```java
public static boolean checkRPN(String[] rpn) {
    Deque<Integer> stack = new LinkedList<>();
    for (String s : rpn) {
        if (Character.isDigit(s.charAt(0))) {
            // 如果是数字，将其压入栈
            stack.push(Integer.valueOf(s));
            continue;
        }
        // 如果是操作符，进行一次计算
        // 弹出栈顶元素作为第二个操作数
        int num2 = stack.pop();
        // 弹出栈顶元素作为第一个操作数
        int num1 = stack.pop();
        if (s.equals("+")) {
            // 执行加法并将结果压入栈
            stack.push( e: num1 + num2);
        } else if (s.equals("-")) {
            // 执行减法并将结果压入栈
            stack.push( e: num1 - num2);
        } else if (s.equals("*")) {
            // 执行乘法并将结果压入栈
            stack.push( e: num1 * num2);
        } else {
            // 无法整除或者除数为0直接返回false
            if (num2 == 0 || num1 % num2 != 0) return false;
            // 执行除法并将结果压入栈
            stack.push( e: num1 / num2);
        }
    }
    // 判断结果是否为24
    return stack.pop() == 24;
}
```

4.3 逆波兰表达式转中缀：

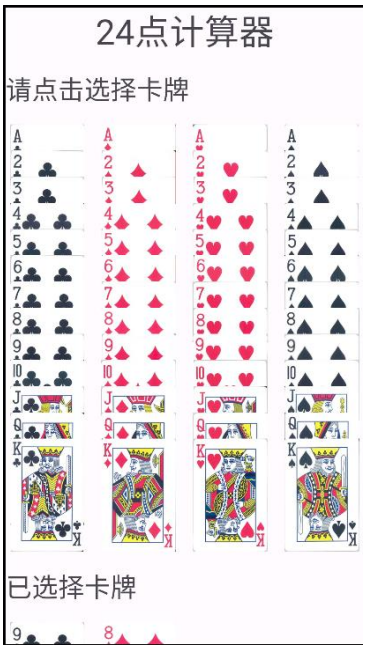
```java
/**
 * 将逆波兰表达式转换为中缀表达式
 *
 * @param rpn 逆波兰表达式
 * @return 中缀表达式
 */
1 个用法  Oreki
public static String rpnToInfix(String[] rpn) {
    Deque<String> stack = new ArrayDeque<>();
    for (String s : rpn) {
        if (Character.isDigit(s.charAt(0))) {
            // 如果是数字，将其压入栈
            stack.push(s);
        } else {
            // 弹出栈顶元素作为第二个操作数
            String operand2 = stack.pop();
            // 弹出栈顶元素作为第一个操作数
            String operand1 = stack.pop();
            // 构建中缀表达式
            String result = "(" + operand1 + s + operand2 + ")";
            // 将中缀表达式压入栈
            stack.push(result);
        }
    }
    // 最后的栈顶元素即为中缀表达式
    String infixExp = stack.pop();
    // 去除多余的括号并返回
    return infixExp.substring(1, infixExp.length() - 1);
}
```

四、实验结果及分析和（或）源程序调试过程（界面截图和文字）、实验总结与体会

1　**实验结果及分析：**

最终效果大致如下：

主页面：



结果展示页面：

用户在选择完四张卡牌后即可点击计算按钮开始计算结果并跳转到结果展示页面。下面我们做一些简单测试：

测试一（有结果时）：

如我们选择 9, 9, 6, 7：

会得到结果：

测试二（无结果时）：

如我们选择 13, 13, 13, 13（显然是无结果的）：

会得到结果：

2　实验总结与体会：

在这个安卓开发实验中，我有机会创建了一个 24 点游戏计算器应用程序，这个经历对我来说是非常有意义的。通过这个实验，我学到了很多关于安卓应用程序开发的知识和技能，同时也提高了我的问题解决和编程能力。

- 项目概述：

在这个实验中，我创建了一个 24 点游戏计算器应用程序，用户可以输入四个数字，然后尝试使用这些数字通过加、减、乘、除等运算得到结果为 24 的表达式。我在应用程序中实现了一个算法来解决这个问题，同时还添加了用户友好的界面和错误检测功能。

- 技术学习：

在实验过程中，我学到了许多与安卓开发相关的技术和概念，包括：

1. Android Studio： 我熟悉了如何使用 Android Studio 来创建、编辑和调试 Android 应用程序。
2. 用户界面设计： 我学习了如何创建简单的用户界面，熟悉了各种控件的使用与自定义。
3. 事件处理： 我掌握了如何处理用户输入事件，例如按钮点击事件，以便触发应用程序中的相应操作。
4. 数学算法： 为了解决 24 点游戏，我研究了数学算法，包括利用逆波兰表达式来辅助问题的解决。
5. 错误处理： 我学习了如何检测和处理应用程序中的错误，以提供更

好的用户体验。

实验报告填写说明：

1、第一、二部分由老师提供；

2、第三部分填写源程序或者算法，清单文件，资源文件等。源程序要符合程序编写风格（缩进、注释等）；

3、第四部分主要填写程序调试运行过程、结果（截图）、解决问题的方法、总结和体会等；

4、报告规范：包含报告页眉、报告的排版、内容是否填写，命名是否规范等。