《软件设计文档》

老司机的眼 项目组

（**github**：https://github.com/chouhaohui/test）

项目组前言：

由于电子设备越来越发达，几乎人人都有手机。有时候在我们需要计算一个算式，但是懒得动脑手算或者手头没有计算器或者希望可以不用手动输入算式内容的时候，我们就希望可以有一个只需要用手机拍下算式就可以得到算式的解的手机应用。这时候“老司机的眼”就应运而生了。这款安卓手机 app 使用手机摄像功能拍下我们需要计算的算式，然后在后台计算得到结果之后显示给用户，解决了上述用户遇到的问题。

Part1 技术选择

1. 安卓开发技术

选择原因：

(1)安卓是一个开发平台，正如 windows 系统一样征服了所有人的使用习惯.

(2)它是基于 Linux 平台的开源操作系统，从而避开了阻碍市场发展的专利壁垒，是一款完全免费的智能手机平台.采用 Android 系统的终端可以有效的降低产品成本.Android 系统对第三方软件开发商也是完全开放和免费的.

(3)该系统由 Google 主导研发，搜索、天气预报、GoogleTalk、地图、Gmail 等均一应俱全，应用方便拥有其他系统无可比拟的优势.

(4)中国移动在安卓操作系统基础上自主研发了 OMS 和基于 OMS 的 Ophone 手机，任何用安卓 API 开发的应用都可以在 OMS 终端上正确运行.

2. OpenCV 技术

选择原因：

计算机视觉市场巨大而且持续增长，且这方面没有标准 API，如今的计算机视觉软件大概有以下三种：

[1]研究代码（慢，不稳定，独立并与其他库不兼容）

[2]耗费很高的商业化工具（比如 Halcon, MATLAB+Simulink）

[3]依赖硬件的一些特别的解决方案（比如视频监控，制造控制系统，医疗设备）这是如今的现状.而标准的 API 将简化计算机视觉程序和解决方案的开发.OpenCV 致力于成为这样的标准 API.

OpenCV 致力于真实世界的实时应用，通过优化的 C 代码的编写对其执行速度带来了可观的提升，因此本小组在图像处理上引用了次技术，以提高识别效率以及准确率.

3. 单一职责原则

单一职责原则又称单一功能原则，面向对象五个基本原则（SOLID）之一.它规定一个类应该只有一个发生变化的原因. 所谓职责是指类变化的原因.如果一个类有多于一个的动机被改变，那么这个类就具有多于一个的职责.而单一职责原则就是指一个类或者模块应该有且只有一个改变的原因.

如果一个类承担的职责过多，就等于把这些职责耦合在一起了.一个职责的变化可能会削弱或者抑制这个类完成其他职责的能力.这种耦合会导致脆弱的设计，当发生变化时，设计会遭受到意想不到的破坏.而如果想要避免这种现象的发生，就要尽可能的遵守单一职责原则。此原则的核心就是解耦和增强内聚性.

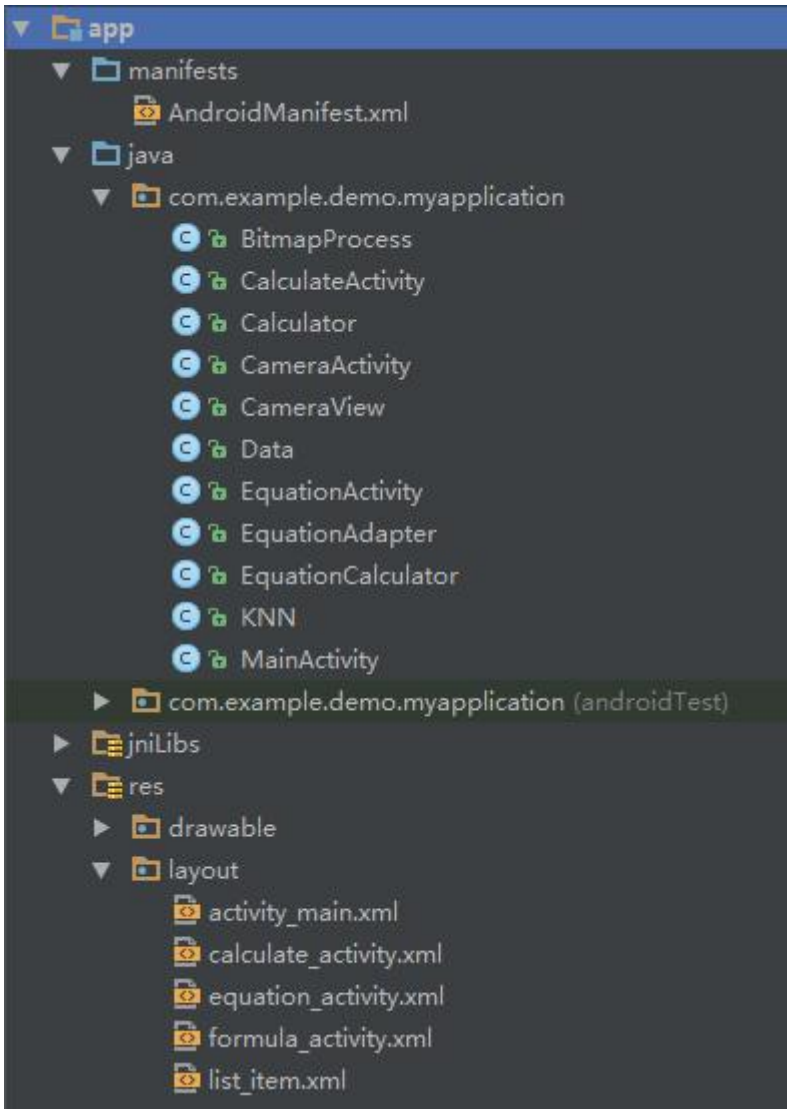
举例说明：

这个类只提供图像处理的函数

```
public class BitmapProcess {

    public BitmapProcess() {...}

    // 轮廓检测
    private ArrayList<MatOfPoint> mFindContours(Bitmap bitmap) {...}

    // 获取轮廓的外接矩形
    public ArrayList<Bitmap> mBoundRect(Bitmap bitmap) {...}

    // 调整切割字符顺序的函数
    private ArrayList<Rect> adjustment(ArrayList<Rect> rectList) {...}

    private int getMinIndex(ArrayList<Rect> rectList) {...}

    // 二值化
    public Bitmap binarization(Bitmap img) {...}

    public ArrayList<Vector> allBitmap2Matrix(ArrayList<Bitmap> boundrects) {...}

    // 将bitmap转换成32*32的Bitmap
    public Bitmap bitmap2SquareBitmap(Bitmap bitmap) {...}

    // 将Bitmap转成32*32矩阵
    public Vector bitmap2Matrix(Bitmap bitmap) {...}

    // Bitmap缩放到32*32Bitmap
    private Bitmap zoom(Bitmap bitmap) {...}

    // Bitmap放大
    private Bitmap zoomIn(Bitmap bitmap) {...}
}
```

Part2 架构设计

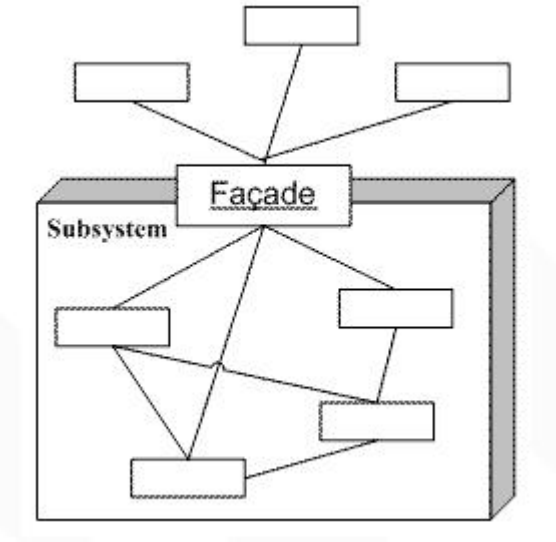
项目目录架构



设计模式：参考了门面模式

门面模式要求一个子系统的外部与其内部的通信必须通过一个统一的门面(Facade)对象进行.门面模式提供一个高层次的接口，使得子系统更易于使用.

就如同医院的接待员一样，门面模式的门面类将客户端与子系统的内部复杂性分隔开，使得客户端只需要与门面对象打交道，而不需要与子系统内部的很多对象打交道.



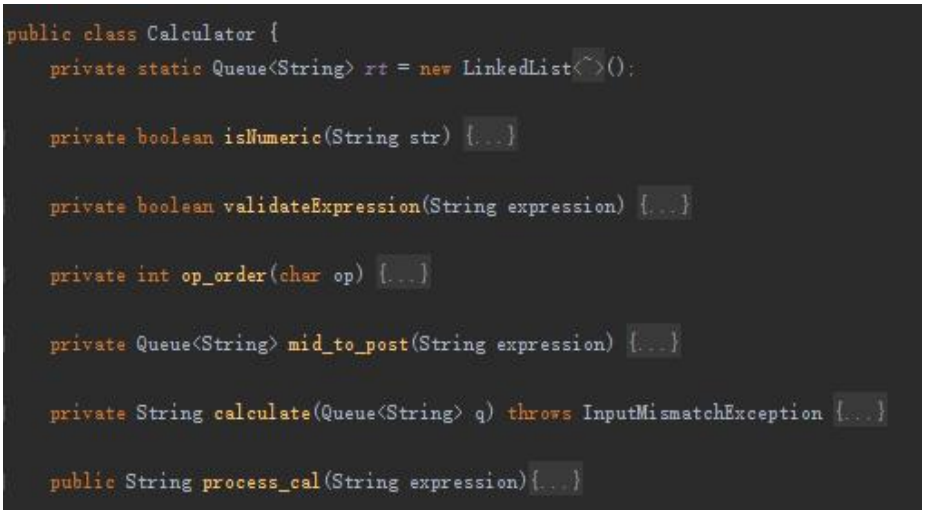
在这个对象图中，出现了两个角色：

门面(Facade)角色：客户端可以调用这个角色的方法. 此角色知晓相关的(一个或者多个)子系统的功能和责任. 在正常情况下，本角色会将所有从客户端发来的请求委派到相应的子系统去.

子系统(subsystem)角色：可以同时有一个或者多个子系统. 每一个子系统都不是一个单独的类，而是一个类的集合。每一个子系统都可以被客户端直接调用，或者被门面角色调用. 子系统并不知道门面的存在，对于子系统而言，门面仅仅是另外一个客户端而已.

本项目参考门面模式，例如以下两个类只提供一个对外的方法：

```
public class EquationCalculator {

    private boolean isNumeric(String str) {...}

    private double[] get_coe(String str) {...}

    // 只需要把包含方程的字符串数组传入即可得到一个解的数组，x = ans[0], y = ans[1]
    // 例如:
    //  String[] s——> s[0] = "4.5x+6y=2"     s[1] = "x+y=3".
    //  调用solution(s) 即可得到一个关于解的数组
    public double[] solution(String[] functions) throws InputMismatchException {...}
}
```



这两个类分别只提供了 solution 和 process_cal 两个方法作为门面角色

门面模式优势：

1) 由于抽象类的实现部分与客户程序之间存在着很大的依赖性。引入 facade 将这个子系统与客户以及其他的子系统分离，可以提高子系统的独立性和可移植性

2) 当你需要构建一个层次结构的子系统时，使用 facade 模式定义子系统中每层的入口点。如果子系统之间是相互依赖的，你可以让它们仅通过 facade 进行通讯，从而简化了它们之间的依赖关系。

3) 它可以对客户屏蔽子系统组件，因而减少了客户处理的对象的数目并使得子系统使用起来更加方便。

4) 它实现了子系统与客户之间的松耦合关系，而子系统内部的功能组件往往是紧耦合的。松耦合关系使得子系统的组件变化不会影响到它的客户。 Facade 模式有助于建立层次结构系统，也有助于对对象之间的依赖关系分层。 Facade 模式可以消除复杂的循环依赖关系。这一点在客户程序与子系统是分别实现的时候尤为重要。在子系统类改变时，希望尽量减少重编译工作以节省时间。用 Facade 可以降低编译依赖性，限制重要系统中较小的 变化所需的重编译工作。

Part3 模块划分

0. 应用启动，预处理模块

功能：载入用于识别模块的数据集

```
// 提前加载KNN数据集
private void loadKNN() {
    String[] trainFiles = getTrainingFileName();
    Vector<Vector> dataset = new Vector<>();
    Vector class_name = new Vector();
    for(int i = 0; i < trainFiles.length; i++) {
        Vector vector = string2Vector(readFile(trainFiles[i]));
        Integer label = KNN.get_class(trainFiles[i]);
        dataset.add(vector);
        class_name.add(label);
    }
    data.setDataset(dataset);
    data.setClassNames(class_name);
}
```

设计思路：

由于数据集数量不少，如果在识别模块才开始载入数据集，会带来两个问题：（1）每次拍照后进入识别的模块都会加载一次数据集（2）加载数据会影响识别速度。所以我们采取了如下改进：在应用启动后，马上加载数据集，并存储在 Android 的 Application 中，Android 的 Application 的生命周期是整个应用的生命周期，所以保存的数据可以随时在 App 中使用。

1. 应用功能选择模块

```
public void onClick(View v) {
    Button button = (Button)findViewById(v.getId());

    final Intent intent = new Intent();
    intent.setClass(MainActivity.this, CameraActivity.class);
    Bundle bundle = new Bundle();
    switch(v.getId()) {
        case R.id.formulaButton:
            bundle.putString("MODE", "formula");
            break;
        case R.id.equationButton:
            bundle.putString("MODE", "equation");
            break;
        case R.id.sudokuButton:
            bundle.putString("MODE", "sudoku");
            break;
    }
    intent.putExtras(bundle);

    mPopField.popView(button);
    new Handler().postDelayed(() -> {
            startActivity(intent);
            MainActivity.this.finish();
    }, 500);
}
```

2. **摄像头拍摄模块**

   功能：拍摄照片并保存

（第一个部分是动态设置界面的大小）

```
private void setLayout(Bundle bundle) {
    GradientDrawable drawable = new GradientDrawable();
    // 获取当前窗口长和宽
    WindowManager windowManager = this.getWindowManager();
    int width = windowManager.getDefaultDisplay().getWidth();

    // 重新设置Camera窗口的大小（不然效果有一点变形）
    int cameraWidth = width;
    int cameraHeight = (int)(((double)width / 480) * 640);
    ViewGroup.LayoutParams layoutParams = cameraSurfaceView.getLayoutParams();
    layoutParams.height = cameraHeight;
    cameraSurfaceView.setLayoutParams(layoutParams);

    if(bundle.getString("MODE").equals("formule") || bundle.getString("MODE").equals("equation")
        //if(getResources().getConfiguration().orientation == Configuration.ORIENTATION_PORTRAIT
        int rectangleWidth = (int)(0.8 * cameraWidth);
        int rectangleHeight = (int)(0.2 * cameraHeight);
        drawable.setSize(rectangleWidth, rectangleHeight);
        //}
    } else if (bundle.getString("MODE").equals("sudoku")) {
        int rectangleWidth = (int)(0.8 * cameraWidth);
        int rectangleHeight = rectangleWidth;
        drawable.setSize(rectangleWidth, rectangleHeight);
    }
    drawable.setStroke(10, Color.parseColor("#FF0000"));
    rectangleLayout.setBackground(drawable);

    // N.LayoutParams中的N是要调整的控件的父类
    FrameLayout.LayoutParams lp = (FrameLayout.LayoutParams)rectangleLayout.getLayoutParams();
    lp.setMargins(0, (int) (0.1 * cameraHeight), 0, 0);

}
```

（第二部分是拍摄部分）

```
public void takePicture() {
    camera.takePicture(null, null, this);
}

public void onPictureTaken(byte[] data, Camera camera) { // 拍摄完成后保存照片
    try {
        String path = Environment.getExternalStorageDirectory() + "/result.png";
        data2file(data, path);
    } catch(Exception e) {
        e.printStackTrace();
    }
    camera.startPreview();
}
```

设计思路：

针对当前市场上手机的屏幕大小不一样，在摄像头拍摄模块我们采用了动态的布局，这样能适应不同的手机的屏幕大小，而拍摄部分是使用了 Android 的接口 android.hardware.Camera，没有采用调用每台手机自带的摄像机的原因是国产 Android 基本都不是原生的 Android 系统，不同系统的摄像 APP 参差不齐，难以根据我们自己的需求去自定义功能和界面。

3. 图像处理模块

功能：将图像中出现的数字切分并转换成一个 01 矩阵

```
public class BitmapProcess {

    public BitmapProcess() {...}

    // 轮廓检测
    private ArrayList<MatOfPoint> mFindContours(Bitmap bitmap) {...}

    // 获取轮廓的外接矩形
    public ArrayList<Bitmap> mBoundRect(Bitmap bitmap) {...}

    // 调整切割字符顺序的函数
    private ArrayList<Rect> adjustment(ArrayList<Rect> rectList) {...}

    private int getMinIndex(ArrayList<Rect> rectList) {...}

    // 二值化
    public Bitmap binarization(Bitmap img) {...}

    public ArrayList<Vector> allBitmap2Matrix(ArrayList<Bitmap> boundrects) {...}

    // 将bitmap转换成32*32的Bitmap
    public Bitmap bitmap2SquareBitmap(Bitmap bitmap) {...}

    // 将Bitmap转成32*32矩阵
    public Vector bitmap2Matrix(Bitmap bitmap) {...}

    // Bitmap缩放到32*32Bitmap
    private Bitmap zoom(Bitmap bitmap) {...}

    // Bitmap放大
    private Bitmap zoomIn(Bitmap bitmap) {...}
}
```

设计思路：

为了后续的识别，在图像识别这一步要将一个数字转换成为 01 矩阵，首先就是要先识别出数字的位置，在图像处理这个部分上 Opencv 就体现了它的优势，Opencv 提供了大量的图像处理函数，这里我们不用关心具体的实现，只需要调用其函数，就可以达到我们的目的。

4. **对象识别模块**

功能：将拍摄对象转化为字符

核心算法：KNN

```
static void KNN(Vector test) {
    Vector<sorting_item> distances = new Vector();
    int dataset_size = dataset.size();
    int dis = 0;

    for (int i = 0; i < dataset_size; i++) {

        for (int j = 0; j < 1024; j++)
            if (test.get(j) != dataset.get(i).get(j))
                dis++;                                          // calculate the distance

        sorting_item temp = new sorting_item(dis, class_names.get(i));

        distances.add(temp);
        dis = 0;
    }

    Comparator my_cmp = new cmp();
    Collections.sort(distances, my_cmp);

    int probable_class_names[] = new int[18];
    for (int i = 0; i < 18; i++)
        probable_class_names[i] = 0;

    int a_class_name;
    int max = 0;
    int result = 0;
    for (int i = 0; i < k; i++) {
        a_class_name = distances.get(i).label;
        probable_class_names[a_class_name]++;
    }

    for (int i = 0; i < 18; i++) {
        if (probable_class_names[i] > max) {
            max = probable_class_names[i];
            result = i;
        }
    }

    predictions.add(result);
}
```

设计思路：

事先预备好了数据集,即不同字体，大小的所有可能出现的字符的多个 01 矩阵样式，保存在特定路径，图像处理模块完成了对象的分割，使本模块获得了即将进行识别的所有 01 矩阵，即特征向量，然后使用 KNN 算法进行分类，最后将字符连成字符串，传递给后续运算模块.

5. **对象运算模块**

功能：将四则运算算式，方程组进行求解

设计思路：

为了实现最初的算式计算的功能，我们决定使用现今非常流行的逆波兰式来计算。

首先我们设计了一个函数（mid_to_post）把中缀表达式转换成逆波兰表达式，然后调用一个函数（calculate）利用逆波兰式进行计算得到结果。其中我们设计了一个用于算式合法性验证的函数（validateExpression）.

为了实现求解 2 元 1 次方程组的功能，我们决定使用矩阵求解方法。因为这种算法可以有效判断方程组有无解，以及得到方程组的正确解。首先我们设计了一个函数（get_coe）用于提取方程组的各个系数。然后利用矩阵求解算法求解方程组的解。

（截止 7.15，成品尚未添加数独识别功能，但数独求解算法已经完成）为了实现求解数独的功能，我们决定使用类似求解 8 皇后问题的标准回溯算法。首先设计一个用于解析数独的函数（parseProblem）,然后用递归算法先填列再填行，每次判断一个数字能否合法填入（legal），当需要回溯的时候把之前填入的数字回复初始的‘0’.求解数独的算法使用了一个递归函数（solve）.

Part4 可扩展性

项目组最初的设想是可以识别并求解四则运算算式，真正实现了这个功能之后，开始考虑能不能加多一些别的功能，比如可以求解方程、方程组、数独等，增加一些附加功能.

于是我们开始设计求解方程的算法，求解 2 元 1 次方程组的算法，以及求解数独的算法。并增加相关的识别功能。我们的数独求解算法使用类似 8 皇后的标准回溯算法，可以在极短的时间内完成对一个数独的求解。我们的 2 元 1 次方程组的算法使用矩阵求解算法，可以正确解出一个 2 元 1 次方程组的解。

之后我们陆续增加了小数计算功能、友好的错误提示功能等，我们一直在设计功能更加完善、对用户更加友好的软件。

Part5 非原创声明

以下给出项目中非原创部分的声明：

1.图像二值化部分（参考网址：
http://blog.csdn.net/nannan_smile/article/details/26738721）

2.项目使用了 opencv-for-android

3.项目中的 UI 动态效果使用 github 上的开源项目（github 地址：
https://github.com/krishnarb3/Popview-Android）