作业 1（场景 1）分析报告

一、主要代码与思路

主要代码可分为三部分。以下是代码结构与主要代码思路。

1. app 配置代码

app/src/main/AndroidManifest.xml，其中：添加了一些需要用到的，以及未来可能用到的权限，于 uses-permission 标签中；添加了一段 query；添加了 app 中使用到的若干个 activity，于 activity 标签中。

2. app UI 代码

位于 app/src/main/res/layout/目录下，其中：

- activity_main.xml：app 主界面的 UI 设计。
- activity_phone.xml：手机信息界面的 UI 设计。
- activity_app.xml：应用信息界面的 UI 设计。
- activity_loc.xml：定位信息界面的 UI 设计。
- activity_sensor.xml：传感器信息界面的 UI 设计。

注：其中仅 main 和 sensor 进行了设计，其余界面的 UI 设计在 kotlin 代码中以另一种方式实现。

3. app 功能代码

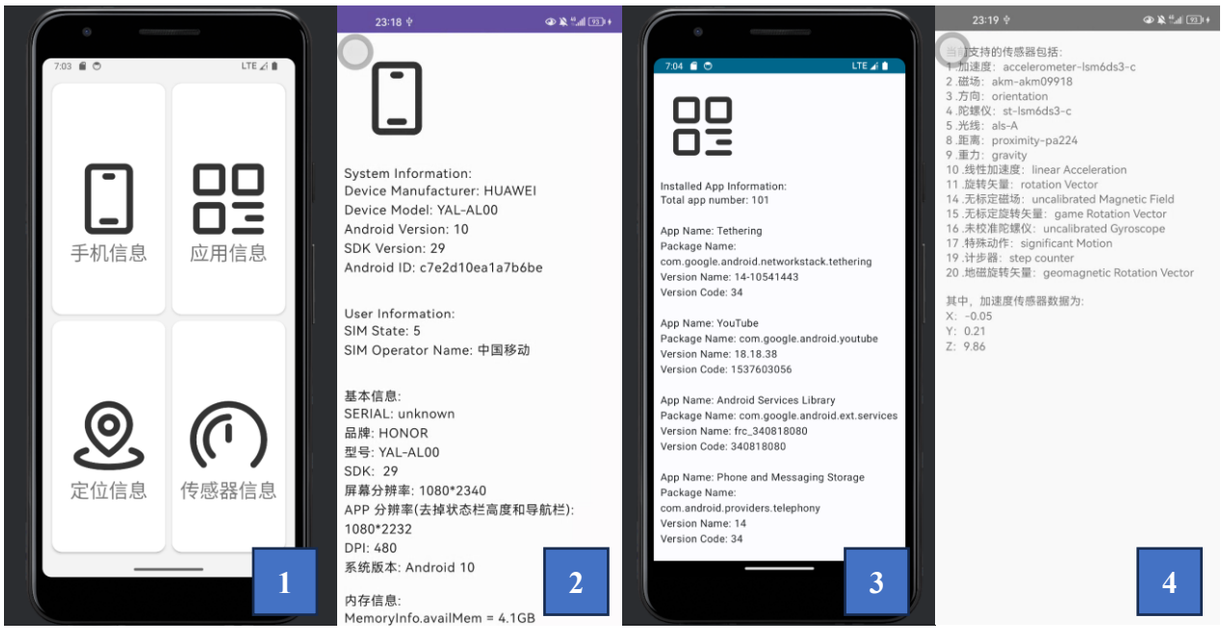
位于 app/src/main/java/com/example/collectinformation/目录下

- MainActivity.kt：app 主界面点击跳转代码。实现了二级菜单界面，美化 app。
- PhoneActivity.kt：**手机信息**界面 UI 绘制与信息收集和输出。调用 DeviceInfoCollector.kt 和 SystemUtils.java 中的方法，获取系统信息和部分用户信息。DeviceInfoCollector 中的函数见下述 DeviceInfoCollector.kt 文件分析。
- AppActivity.kt：**应用信息**界面 UI 绘制与信息收集和输出。调用 DeviceInfoCollector.kt 中的 getInstalledApps 函数。
- LocActivity.kt：**定位信息**界面 UI 绘制信息收集和输出。在实验过程中我们发现，用户位置信息难以获取，因此此功能暂未实现，仅有 UI 界面。在实验报告第三部分中，将具体分析问题的原因。
- SensorActivity.kt：**传感器信息**收集和输出。首先获取了 SensorManager 服务，并使用 getDefaultSensor() 方法获取了加速度传感器（类型为 Sensor.TYPE_ACCELEROMETER）。然后使用 registerListener() 方法注册了一个传感器事件监听器（SensorEventListener）。这个监听器会在传感器数据发生变化时被调用。在 onSensorChanged() 方法中，当传感器数据发生变化时，应用程序会获取传感器的三个轴向（X、Y、Z）的数值并显示在相应的 TextView 控件上。同时，代码还调用了 SystemUtils.showSensorInfo，该方法会遍历设备上所有可用的传感器，并将传感器类型和名称进行输出。
- DeviceInfoCollector.kt：公共 object 定义，其中实现了一些获取信息的代码。
  getSystemInfo 函数用于获取系统信息（包含在手机信息中），其通过访问 Build 类的静态属性来获取设备制造商、型号、Android 版本和 SDK 版本；使用 Settings.Secure.getString() 方法获取 Android ID。
  getUserInfo 函数用于获取用户信息，它通过获取 TelephonyManager 服务，并调用 getSimState() 和 getNetworkOperatorName() 方法来获取 SIM 卡状态和网络运营商名称。
  getInstalledApps 函数用于获取应用信息，它通过调用 PackageManager 的 getInstalledApplications() 方法获取所有已安装的应用程序列表，然后遍历这个列表并使用 loadLabel(), getPackageName(), getAppVersionName() 和 getAppVersionCode() 等方法获取每个应用的名称、包名、版本名称和版本号。
- util/SystemUtils.java：其中实现了一些获取信息的代码。封装了各类实用方法。

二、收集信息展示

图 1：运行 app，会显示主界面，有四个卡片，可以点击查看相应的信息。
图 2：点击手机信息，会输出系统信息、和部分用户信息。包括设备厂商、安卓版本、SDK 版本、安卓 ID、品牌名、屏幕分辨率、DPI、内存信息、MAC 地址、存储信息、CPU 信息等。

图 3：点击应用信息，会输出各 app 信息。包括 app 的总数目，以及每个 app 的名称、包名称、版本号和版本代码。
图 4：点击传感器信息,会输出目前支持的传感器列表，并以其中一个加速度传感器为例，输出其数据。



三、问题与分析

1. 为什么用户位置信息难以获取

APP 应用获取用户位置的定位方式为网络定位，网络定位相当于 GPS 定位+WIFI 定位+基站定位。发起定位时的定位优先顺序为 GPS 优先，如果手机 GPS 关闭或获取不到卫星信号则采用 WIFI 定位，如果手机 WIFI 功能关闭则使用基站定位。而 GPS 在室内、隧道等卫星信号覆盖弱的地方是定不了位的，基站定位没有这些问题却又误差太大，WIFI 定位则很好的解决了这二种定位方式所存在的问题。

使用 WIFI 定位时，手机只需要开启 WIFI 功能，无需连接网络。WIFI 只要通电，不管用什么加密方式，都一定会向周围发射信号。信号中包含此 WIFI 的唯一 MAC 地址。MAC 地址是网卡决定的，它是固定的，即使距离此热点比较远，但还是可以侦听到它的存在。地图商采集了所有热点的 MAC 地址，与经纬度对应，并持续更新，当 APP 应用请求获取位置信息时，地图商位置数据库根据手机所侦听到的 WIFI 信息，结合经纬度信息运算出用户当前的地理位置。

然而，通常情况下，没有用户授权，APP 是无法直接访问设备的 GPS、WIFI 等信息的，这是为了保护用户的隐私。用户必须明确地授予应用程序权限才能访问他们的位置信息。在大多数操作系统中，用户会在安装应用程序或在应用程序首次尝试获取位置信息时被提示授予权限。

因此，没有与用户进行交互的情况下，我们的 APP 是无法收集到用户的位置信息的。

2. 用户信息及获取的难易程度

我们探究了安卓中用户信息有哪些，获取的难易程度由什么决定。在安卓中用户信息包括位置定位、联系人、通讯录联系人、短信/通话记录、相册图片、账号信息、日历等等。获取用户信息的难易程度主要取决于：1.信息涉及的用户隐私程度，涉及隐私的信息获取难度大；2. 系统对信息访问的限制，隐私信息访问一般受到系统限制，需要用户授权；3. 信息的敏感程度，账号等信息相对容易获取。

具体而言，安卓通过权限管理来决定用户信息的可获取性：安卓中权限可分为两类：危险权限和普通权限。危险权限需要在运行时动态请求用户授权，如果用户未授权，则应用程序无法使用该权限。危险权限包括 9 组：短信权限、存储权限、联系人权限、手机权限、日历权限、相机权限、位置权限、传感器权限、麦克风权限。而对于普通权限，则不需要动态请求用户授权，只需要在 AndroidManifest.xml 文件中声明即可。

四、其他说明

- 开发环境：使用的 IDE 为 Android Studio。项目代码可直接在 Android Studio 手机模拟器上运行。
- 项目 GitHub 仓库：https://github.com/woozyc/MISS-project