



面向对象的软件构造实践

实验四
(4学时)

2024春



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

用户界面

事件处理

图形系统

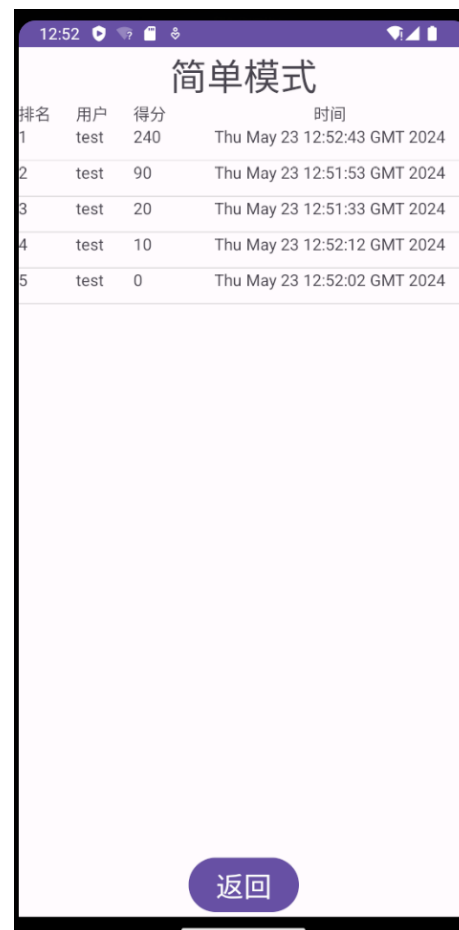
**数据存储
与展示**

音乐音效

网络编程

模块功能：完成数据存储和展示

① 完成用户数据的存储，及得分页面的展示



The screenshot shows a mobile application interface with a purple header bar. The title '简单模式' (Simple Mode) is centered at the top. Below the title is a table with four columns: '排名' (Rank), '用户' (User), '得分' (Score), and '时间' (Time). The table contains five rows of data. At the bottom of the screen, there is a purple button labeled '返回' (Return).

排名	用户	得分	时间
1	test	240	Thu May 23 12:52:43 GMT 2024
2	test	90	Thu May 23 12:51:53 GMT 2024
3	test	20	Thu May 23 12:51:33 GMT 2024
4	test	10	Thu May 23 12:52:12 GMT 2024
5	test	0	Thu May 23 12:52:02 GMT 2024

- 了解Android数据存储的方法，掌握APP内部文件存储的实现方式；
- 掌握使用适配器和适配器控件展示和处理数据的方法；
- 掌握AlertDialog的使用；
- 了解Handler的工作机制，掌握子线程通过主线程Handler更新主线程UI的方法。

4.1 数据存储

4.2 适配器和适配器控件

4.3 AlertDialog

4.4 Handler

Android应用数据存储的方式:

- 1、File文件存储：将文件保存在设备的内部存储空间；
- 2、SharedPreferences存储数据：一种轻量级的存储方式，用于存储键值对数据，适用于存储一些简单的配置信息；
- 3、SQLite数据库：用于存储结构化的数据，轻量级的数据库，支撑标准的SQL语句进行数据的增删改查操作；

注：SharedPreferences和SQLite存储方式，如若同学想使用可以自学！

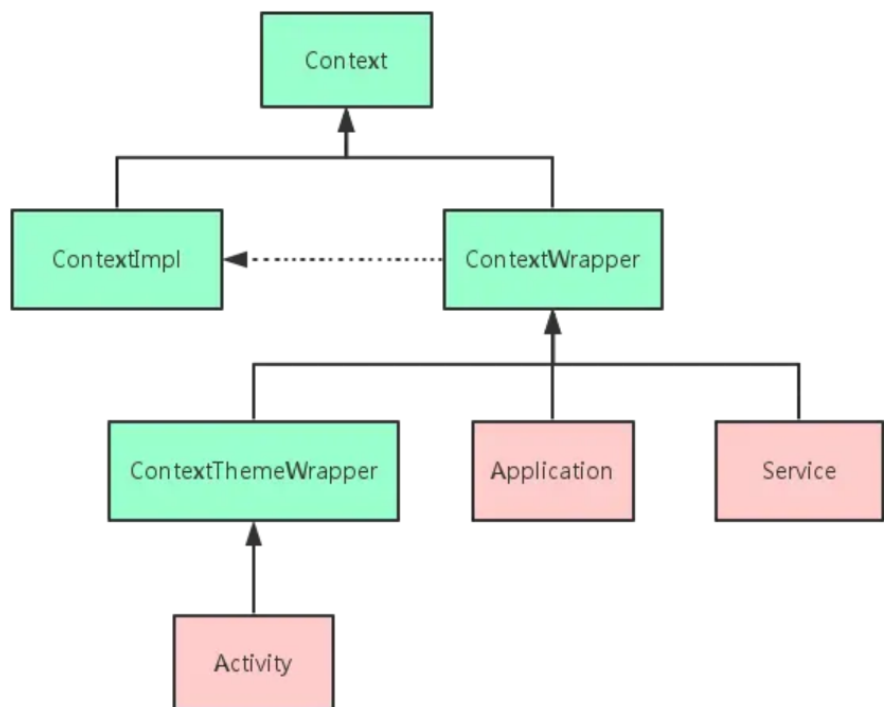
文件存储

<code>openFileOutput(filename, mode)</code>	打开文件输出流，往文件中写入数据，第二个参数是模式
<code>openFileInput(filename)</code>	打开文件输入流，读取文件中的信息到程序中

```
String file = "userinfo.txt";
try {
    FileOutputStream fos = openFileOutput(file, MODE_PRIVATE);
    FileInputStream fis = openFileInput(file);
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

- 使用openFileOutput和openFileInput方法读写文件，指定文件名，不需要指定路径
- 如文件不存在，会在应用项目所在空间下创建文件
- Activity类中可以直接使用上述读写方法，非Activity类要先获得Context对象，使用 *context.openFileInput/openFileOutput()*。

4.1 数据存储



- Context描述一个应用程序环境的信息，是维持Android程序中各组件能够正常工作的一个核心功能类。
- Context是抽象类，Android提供了该抽象类的具体实现类；通过它我们可以获取应用程序的资源。
- ***Activity.this***：返回当前的Activity实例，从而获得Context对象。

4.1 数据存储

查看本地文件

- 必须先启动Android Studio的模拟器，然后单击Android Studio编辑窗右下角的Device File Explorer标签，才能查看本地存储的内容。所有的本地存储都在data/data/<包名>/files下面。

Device File Explorer		
Copy_of_Pixel 3a API 34 Android API 34		
Name	Permissions	Date
> com.android.wallpapercropper	drwxrwx--x	2024-05-10 06:58
▼ com.example.aircraftwar2024	drwxrwx--x	2024-05-10 06:58
> .agent-logs	drwx-----	2024-05-11 01:27
> cache	drwxrws--x	2024-05-10 06:58
> code_cache	drwxrws--x	2024-05-23 08:26
▼ files	drwxrwx--x	2024-05-11 01:08
records.txt	-rw-rw----	2024-05-11 01:04
simple.txt	-rw-rw----	2024-05-11 01:08

4.2 适配器和适配器控件

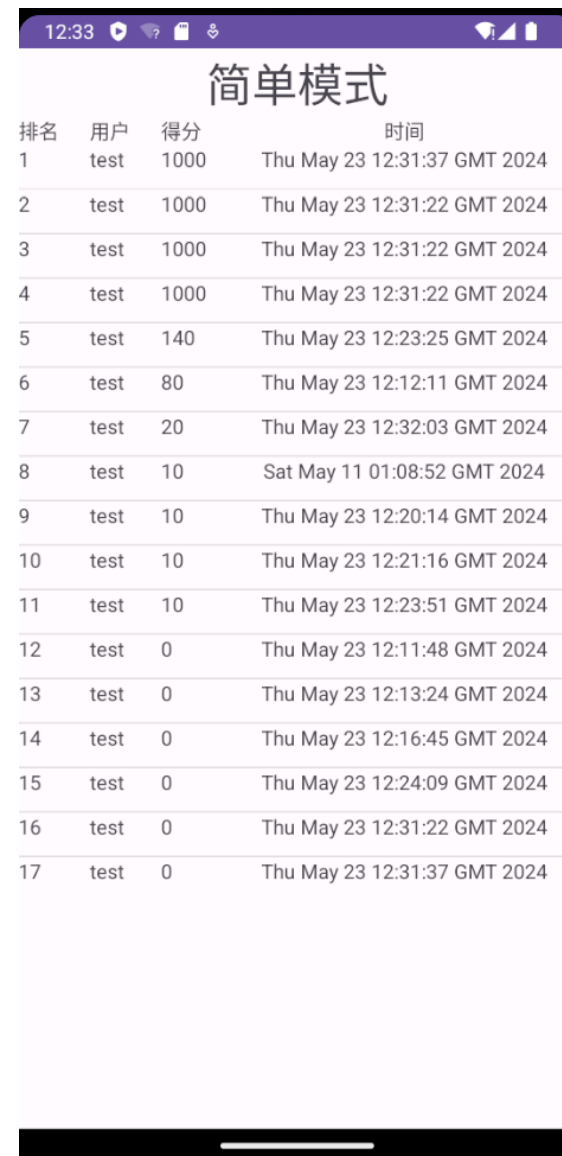
适配器控件ListView

- 以表格的形式显示数据
- 处理用户操作

适配器SimpleAdapter

- 绑定数据，用数据填充布局。

注：
适配器也可以使用BaseAdapter



排名	用户	得分	时间
1	test	1000	Thu May 23 12:31:37 GMT 2024
2	test	1000	Thu May 23 12:31:22 GMT 2024
3	test	1000	Thu May 23 12:31:22 GMT 2024
4	test	1000	Thu May 23 12:31:22 GMT 2024
5	test	140	Thu May 23 12:23:25 GMT 2024
6	test	80	Thu May 23 12:12:11 GMT 2024
7	test	20	Thu May 23 12:32:03 GMT 2024
8	test	10	Sat May 11 01:08:52 GMT 2024
9	test	10	Thu May 23 12:20:14 GMT 2024
10	test	10	Thu May 23 12:21:16 GMT 2024
11	test	10	Thu May 23 12:23:51 GMT 2024
12	test	0	Thu May 23 12:11:48 GMT 2024
13	test	0	Thu May 23 12:13:24 GMT 2024
14	test	0	Thu May 23 12:16:45 GMT 2024
15	test	0	Thu May 23 12:24:09 GMT 2024
16	test	0	Thu May 23 12:31:22 GMT 2024
17	test	0	Thu May 23 12:31:37 GMT 2024

实现步骤

- 准备要显示的数据;
- 在xml布局文件中添加ListView;
- 创建列表项的布局文件, 定义ListView每行的布局;
- 构建适配器;
- 使用setAdapter(), 把适配器绑定到控件上;
- 为适配器控件添加事件监听器, 响应用户操作。

具体实现方法参考项目ListViewSimpleAdapter

4.3 AlertDialog

1. 标题

此为可选项，仅当内容区域被详细消息、列表或自定义布局占据时才应使用标题。如果需要陈述的是一条简单消息或问题（如图 1 中的对话框），则不需要标题。

2. 内容区域

内容区域可显示消息、列表或其他自定义布局。

3. 操作按钮

对话框中的操作按钮不应超过三个。

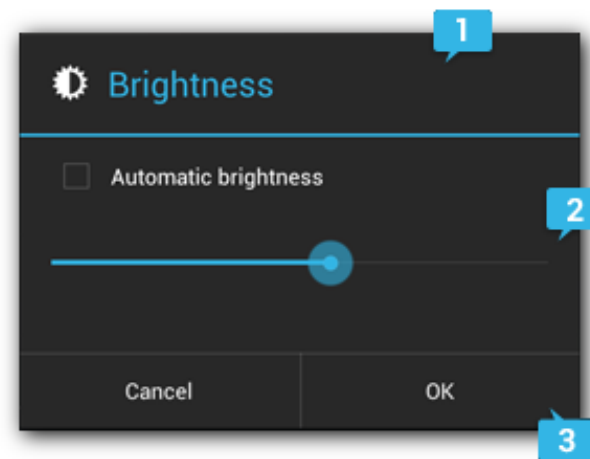


图 2. 对话框的布局。

使用AlertDialog.Builder类提供的API来创建AlertDialog

4.3 AlertDialog

构建AlertDialog

```
// 1. Instantiate an AlertDialog.Builder object  
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
  
// 2. Chain together various setter methods to set the dialog characteristics  
builder.setMessage(R.string.dialog_message)  
    .setTitle(R.string.dialog_title);  
  
// 3. Get the AlertDialog object from the builder  
AlertDialog dialog = builder.create();
```

创建成功后调用**dialog.show()**
方法显示AlertDialog控件

添加按钮

`setPositiveButton()` 方法需要一个按钮标题（由字符串资源提供）和一个 `DialogInterface.OnClickListener`，后者用于定义用户按下该按钮时执行的操作。

```
builder.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int id) {  
        // User clicked OK button  
    }  
});  
builder.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int id) {  
        // User cancelled the dialog  
    }  
});  
// Set other dialog properties  
...
```

Android**单线程** (Single-threaded) 模型

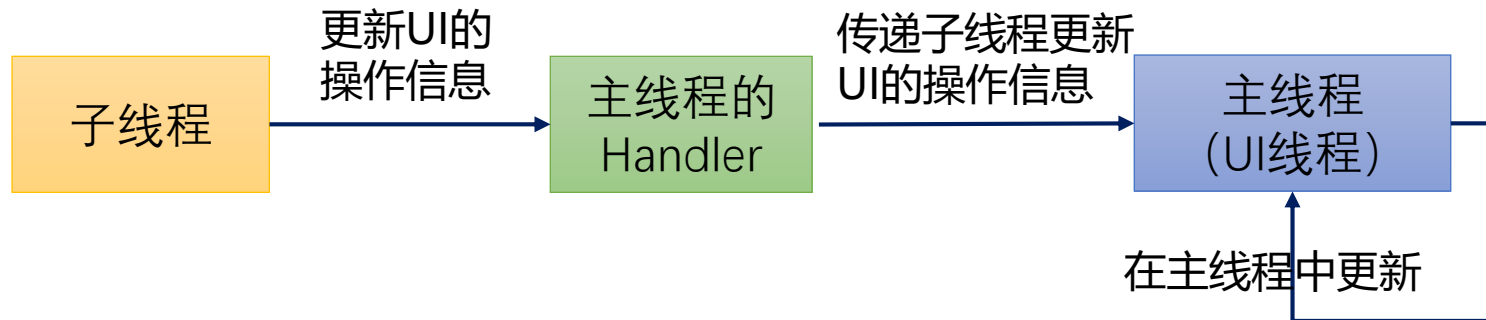
- 当一个程序第一次启动时，Android会启动一个**主线程** (Main Thread) 。
- 主线程主要负责处理与UI相关的事件，常被称为**UI线程**。
- UI线程能根据用户的要求做出快速响应，不宜占用太长的时间。如果占用时间超过10秒，Android系统就会给用户显示ANR提示信息。
- Android系统将所有运行慢的、耗时操作交给**子线程**，以解放UI线程，避免阻塞。
- 子线程操作UI对象是不安全的，会导致CalledFromWrongThreadException。为了解决子线程与主线程（UI线程）间的信息交互问题，Android设计了一种**Handler消息传递机制**。

4.4 Handler

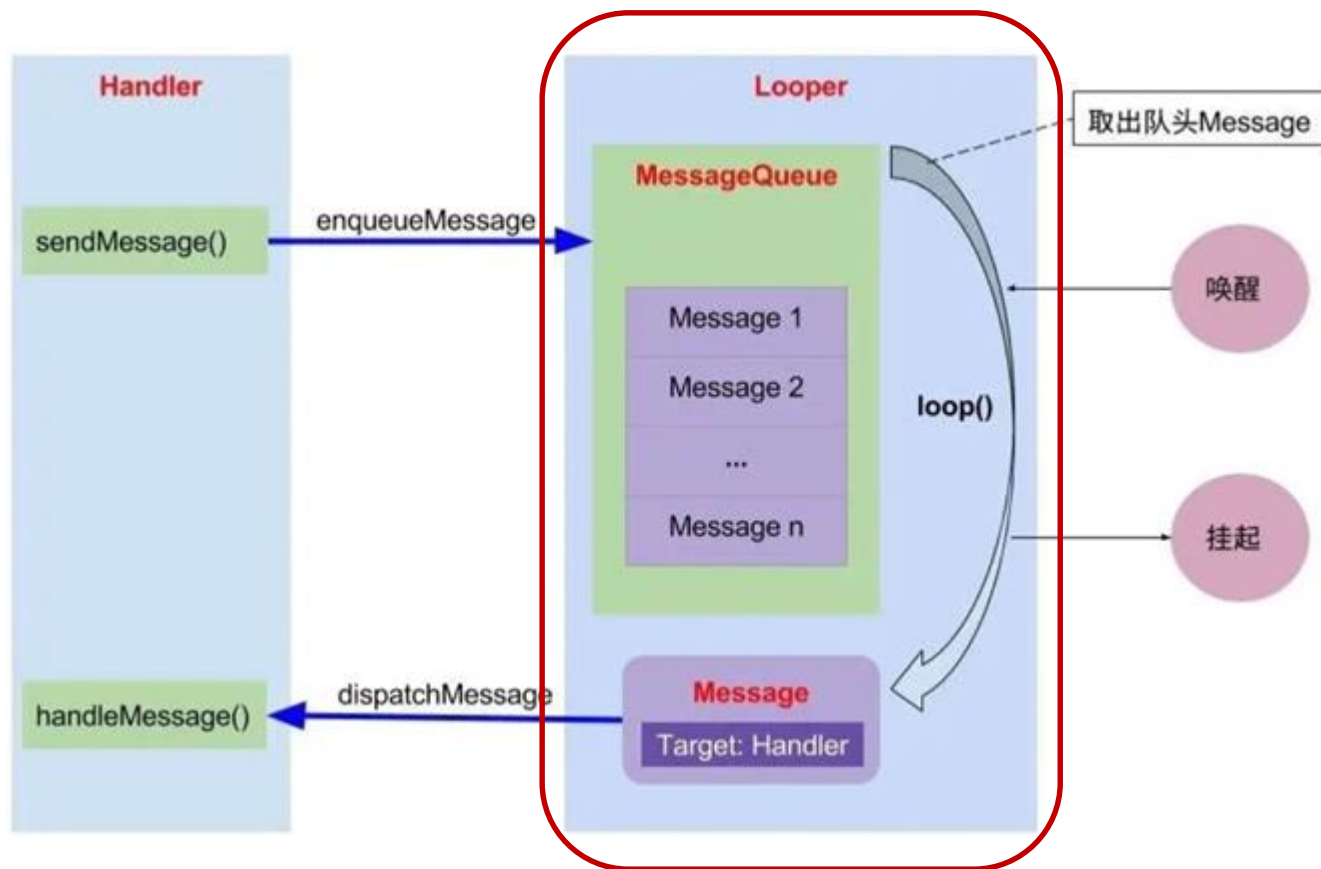
思考：游戏结束时如何跳转到排行榜页面？

BaseGame是SurfaceView，运行在子线程，无法直接更新UI。

GameActivity运行在主线程，BaseGame发送信息给GameActivity，由GameActivity加载排行榜页面。

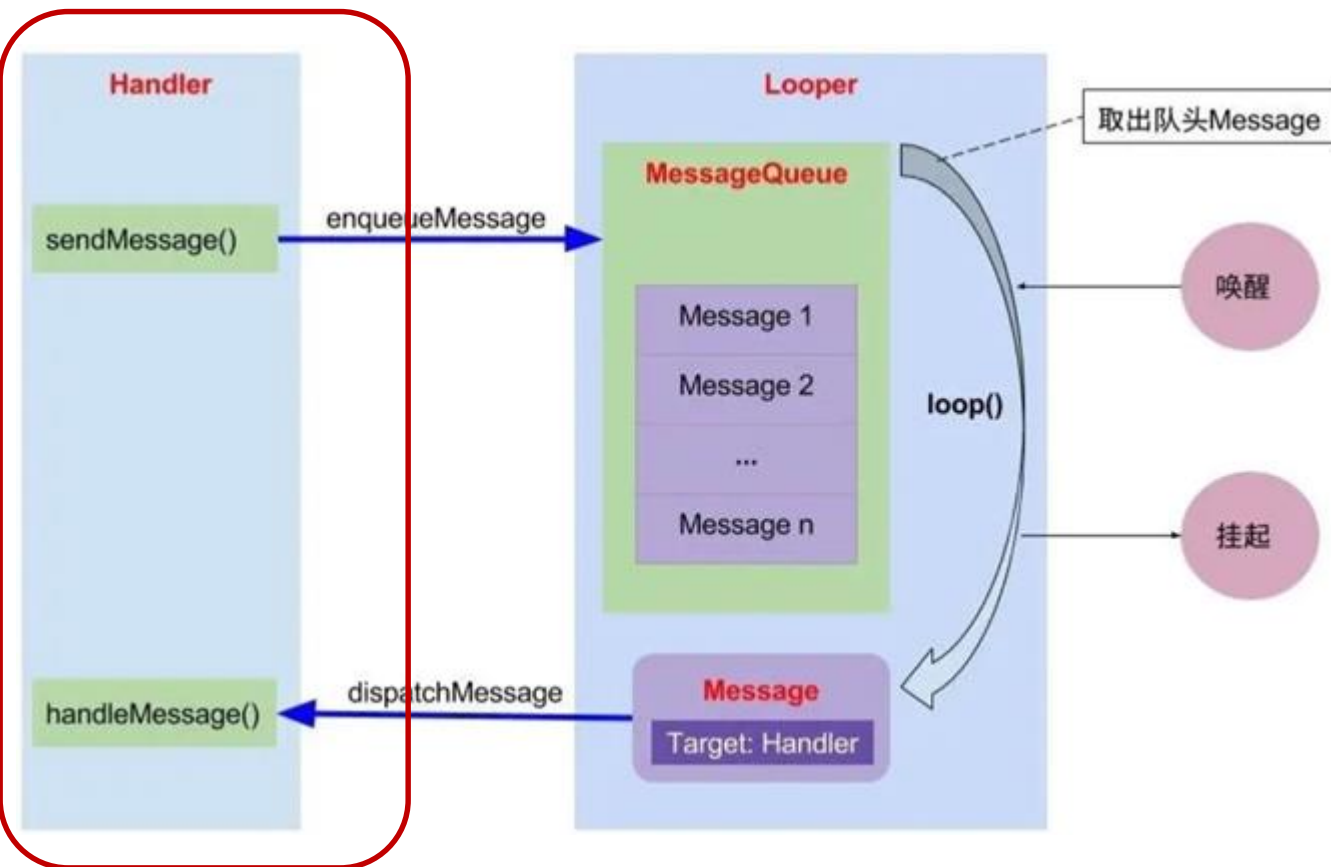


4.4 Handler



- **Message** (消息) 类, 保存要传递的信息;
- **MessageQueue** (消息队列) 类, 采用先进先出的方式来管理Message;
- **Looper类**, 每个Looper对应一个Message Queue, 所有线程有且只有一个Looper, 应用在启动的时候, 系统就自己给我们创建了一个Looper并与主线程绑定了。

4.4 Handler



Handler的作用有两个：**发送消息**和**处理消息**。

- Handler通过`sendMessage()`发送Message到MessageQueue队列，指明target为当前Handler
- Looper通过`loop()`，不断提取出达到触发条件的Message，并将Message交给target来处理
- 经过`dispatchMessage()`后，交回给Handler的`handleMessage()`来进行相应地处理

子线程通知主线程更新UI

- 在主线程Activity中创建Handler类的对象，重写handleMessage()方法。
 - 创建主线程Handler时，系统自动为UI线程初始化了一个Looper对象

```
public Handler mHandler = new Handler(Looper.getMainLooper);
```
- 在子线程中调用Handler对象的sendMessage()或sendMessage()方法发送消息。
- Handler类的对象用handleMessage()方法接收消息，然后根据消息的执行相应的操作。

具体实现方法参考实验包中的项目HandlerDemo

4-1 排行榜页面的跳转及事件处理

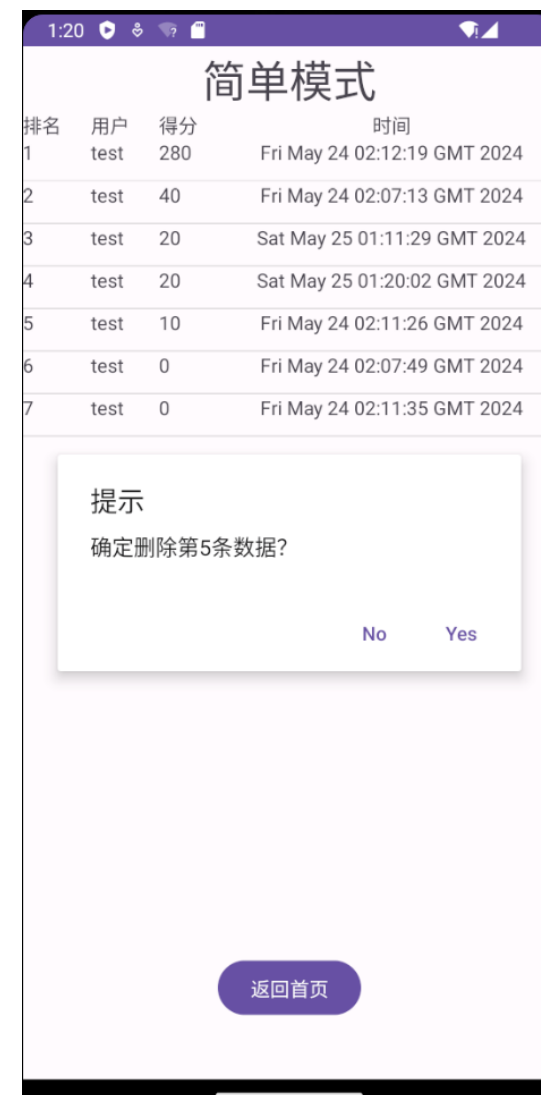
- 游戏结束通过Handler跳转到排行榜页面；
- 删除选定行的数据；
- 实现返回首页的功能。

4-2 数据存取和显示

- 移植导论的文件存储部分代码到安卓平台，文件存储在应用程序空间内；
- 设计开发排行榜页面；
- 在排行榜界面显示当前游戏难度和游戏数据。

4-1 排行榜页面的跳转及事件处理

1. 在GameActivity类中创建主线程Handler;
2. BaseGame类中检测到英雄机死亡后向主线程Handler发送消息;
3. 主线程Handler接收到游戏结束消息后, 加载 activity_record.xml布局文件, 展示排行榜数据;
4. 使用AlertDialog实现单击一行删除记录的功能 (删除后需要更新视图) ;
5. 使用ActivityManager类实现返回游戏首页的功能;



4-1 排行榜页面的跳转及事件处理

6. 由于英雄机使用的是单例模式，需要修改HeroAircraft.java文件。

```
private HeroAircraft() {  
    super( locationX: GameActivity.screenWidth / 2, locationY: GameActivity.  
        speedX: 0, speedY: 0, hp: 1000);  
    this.shootNum = 1;  
    this.power = 30;  
    this.direction = -1;  
    this.rate = 3;  
    this.isValid = true;  
    shootStrategy = new DirectShoot();  
}  
/** 通过单例模式获得初始化英雄机 ...*/  
6 usages  
public static HeroAircraft getHeroAircraft(){...}  
7 usages  
@Override  
public void forward() {}  
11 usages  
@Override  
public void vanish(){  
    super.vanish();  
    heroAircraft = null;  
}
```

4-2 数据存储及显示

1. 迁移并修改导论中**自己实现的文件读取代码**（文件应该创建在应用程序私有空间）；
2. 新建布局文件activity_record.xml，此布局文件显示游戏难度，同时使用ListView来展示排行榜数据；
3. 新建布局文件activity_item.xml，此布局规定了排行榜中一行使用4个文本框分别显示用户数据的排名，用户名，得分和时间，其中用户名硬编码为 *test*；
4. 创建SimpleAdapter对象绑定数据，调用ListView.setAdapter方法将适配器绑定到视图控件上以显示数据。