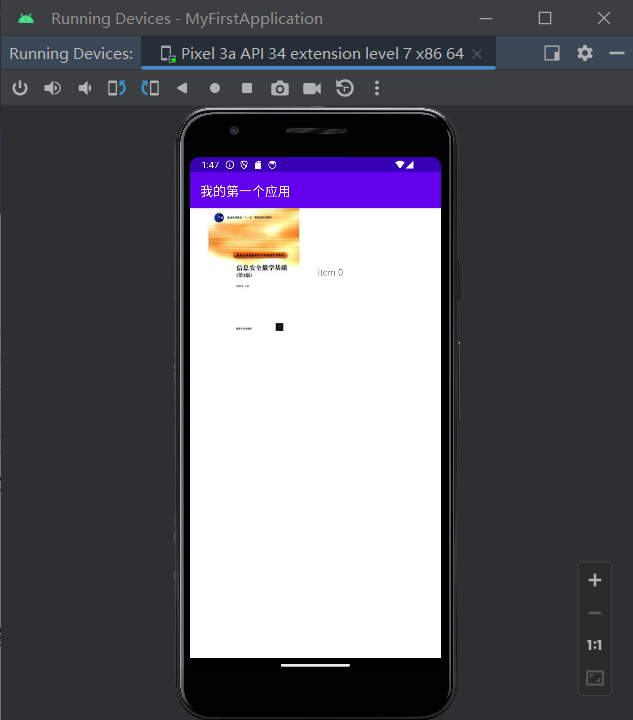
|  |
| --- |
| **一、实验目的** |
| * 了解Activity之间的跳转 * 了解Activity之间的数据传递。 * 掌握数据持久化 |

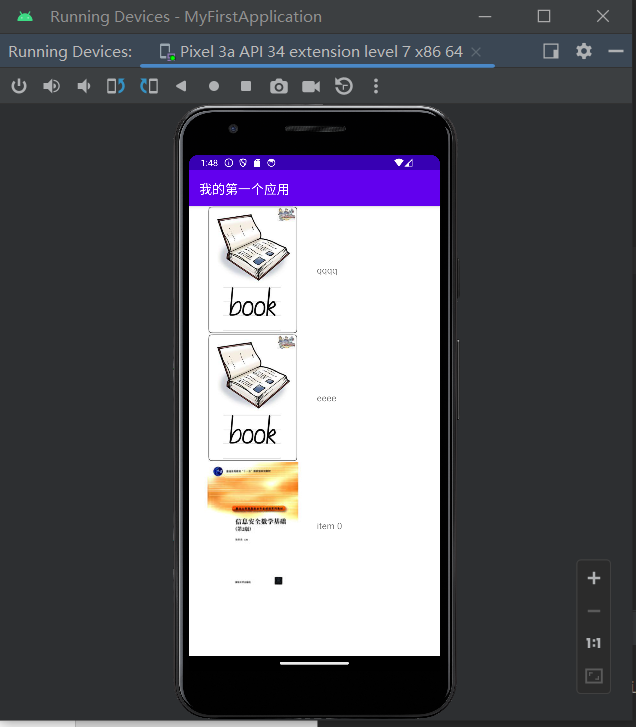
|  |
| --- |
| **二、实验环境** |
| Android Studio |
| **三、实验内容** |
| 在上次实验的基础上将添加新数据的Activity命名为的BookDetailsActivity。这个BookDetailsActivity只包含一个TextView显示“图书名称”.一个TextEdit和一个“确定”Button，一个“取消”Button。给现有程序RecyclerView中添加“修改”上下文菜单，点击菜单件时，把对应Item的文本/位置打包传递给新创建的一个BookDetailsActivity窗口,在BookDetailsActivity的EditText显示传过来要编辑的文字。编辑文字后，点击“确定”Button。实现点击事件关闭BookDetailsActivity把数据传回主界面并在RecyclerView中添加,添加的图书封面为book\_no\_name.png。同时也请实现“新建”菜单的事件响应功能，新建打开新窗口，传回数据并在RecyclerView上添加新建的图书记录。  Android数据的持久化有很多种方法，由于我们的数据不会太多，我们采取将数据以对象的方式保存到APP的私有空间。这样的操作不需要额外的权限。参考 https://www.jianshu.com/p/4c020b45f9e3 可以实现文件的读取/保存。参考https://www.jianshu.com/p/0997e287f341实现数据对象的序列化。在我们上次程序的基础上，在合适的地方添加数据保存和读取的功能实现数据的持久化。数据/显示/控制的分离是软件设计中常用的方式。合理地设计你的数据类，把你上次的程序数据和控制分离开来。  在实现上述功能的基础上，回答下述问题。  1.Activity采用bundle和intent传递数据时，对数据类型有没有要求，能不能传递对象？  Intent需要提供数据的键值对，键是一个字符串，值可以是各种类型，如字符串、字节、字符数组、整数、浮点数、布尔值、意图、广播、包、IBinder等。但是，不能直接传递对象。如果你需要传递对象，那么这个对象必须实现Serializable接口。  Bundle的工作方式类似于一个Map，可以存储键值对。但是，不能直接存储对象。它主要用于在Activity之间传递原始数据和基本类型。然而，和Intent一样，如果你要传递的对象实现了Serializable接口，那么你可以通过Bundle传递它。  2.直接使用静态全局对象传递数据的话有什么优缺点？  优点：  1.易于访问：静态全局对象在程序的任何地方都可以访问，这使得数据传递非常方便。  2.单例模式：静态全局对象只会在程序启动时创建一次，这有助于管理资源，例如数据库连接或线程池。  3.无需反复创建对象：由于静态全局对象只存在一个实例，所以不会反复创建和销毁对象，这可以减少内存和CPU的开销。  缺点：  1.全局状态：静态全局对象可能会导致全局状态的问题。由于在程序的任何地方都可以修改它，可能会导致程序的行为变得难以预测和理解。  2.线程安全问题：如果多个线程同时访问和修改静态全局对象，可能会引发线程安全问题。需要使用同步机制（如锁）来确保线程安全。  3.生命周期问题：静态全局对象的生命周期与程序的生命周期相同，因此可能难以管理和清理资源。例如，如果静态全局对象持有数据库连接，那么在程序结束时必须确保正确关闭这个连接。  4.可测试性和依赖性：静态全局对象往往使得单元测试和依赖注入变得更加困难。依赖于静态全局对象，使得代码更难于独立测试和替换。  3.Activity退出时有很多种退出，一种是按钮实现调用finish的这种,还有一种是直接按手机上的返回键。如果按手机上的返回键需要实现哪些函数以便返回数据。  当用户按下手机上的返回键时，系统会调用 Activity 的onBackPressed()方法，在启动这个 Activity 的前一个 Activity 中，你可以在onActivityResult()方法中获取这些返回的数据   1. Activity之间传递数据为什么要特殊处理，请调查Activity的生命周期和这个问题之间的关系。   在Activity之间传递数据时，需要特殊处理，因为不同的Activity处于不同的生命周期状态，可能导致数据的丢失或者无法传递。例如，当从Activity A启动Activity B时，如果直接使用Intent对象传递数据，那么当Activity B变为暂停状态或者停止状态时，这部分数据可能会丢失。  Activity是Android应用中的基本组件之一，用于向用户展现数据并与之交互。在Activity的生命周期中，它经历了多个状态，包括活动状态、暂停状态和停止状态等。当用户启动一个Activity时，它处于活动状态，可以与用户进行交互。但当用户按下返回键或者有新的Activity启动时，当前Activity会变为暂停状态或者停止状态。   1. Android有几种保存数据文件的方式方法？各有什么优缺点。   Android平台上有多种保存数据文件的方式，以下是五种常用的存储方式及其优缺点：  1.文件存储：可以用于存储大量的数据，包括图片、视频、文本等。文件存储又分为内部存储和外部存储（例如SD卡）。内部存储在应用程序被卸载后会被清除，而外部存储则不会。这种方式的优点是简单易用，适用于大量数据的存储，缺点是如果采用内部存储的方式，存储过量的数据可能会导致内存的不足，而外部存储的方式如果要删除或卸载应用，相关的数据需要手动进行删除，比较麻烦。  2.SharePreferences存储：轻量级、键值对存储，使用方便，易于理解，是作为一些小数据的存储比较常规的存储方式。缺点是适用于存储少量的数据，并且数据的格式只能是基本的数据类型（int、float、long、boolean）、字符串类型（string），不能单独使用，必须与其他存储方式结合使用。  3.SQLite数据库存储：将数据存储在数据库中，有利于数据的长期保存和高效检索，同时也支持对数据的增删改查操作。但是，如果数据量较大，会增加应用程序的负担，同时也会占用较多的系统资源。  4.ContentProvider：通过ContentProvider来实现不同应用程序之间的数据共享，可以保证数据的一致性和安全性。但是，使用ContentProvider需要开发人员具有一定的开发经验和对Android系统深入的理解。  5.网络存储：将数据存储在云端或服务器上，可以实现数据的共享和实时更新。但是，需要考虑到网络连接的稳定性和数据的安全性。   1. 如果要保证编辑的数据不丢失，你的程序中数据保存、读取的代码应当放在哪些事件响应函数中？   1.onCreate() 和 onResume(): 这两个生命周期方法是在Activity创建和恢复到前台时被调用的。你可以在这两个方法中执行数据读取和保存的逻辑，以确保在用户与应用程序交互之前，数据已经正确加载 并保存用户可能已经之前的修改。  2.onPause() 和 onStop(): 当Activity进入暂停或停止状态时，你可以在这两个方法中执行数据保存的逻辑。这样，即使用户离开Activity或设备发生屏幕锁定等操作，你也可以保存他们未保存的更改。  3.onOptionsItemSelected() 或 onMenuItemSelected(): 如果你有一个菜单选项，用户可以通过它来保存或加载数据，那么你可能会在这些方法中编写保存和读取的逻辑。  4.onSaveInstanceState() 和 onRestoreInstanceState(): 当Activity保存和恢复实例状态时，你可以使用这两个方法来保存和读取数据。这对于那些在Activity生命周期中可能丢失的数据非常重要。  5.onBackPressed(): 如果用户在编辑数据后选择返回，你可以在此方法中保存更改。  6.onClick() 或其他用户交互方法: 如果你的用户可能会通过点击按钮或其他交互方式来保存数据，那么这些方法也是保存数据的好地方。 |

**四、实验及分析**

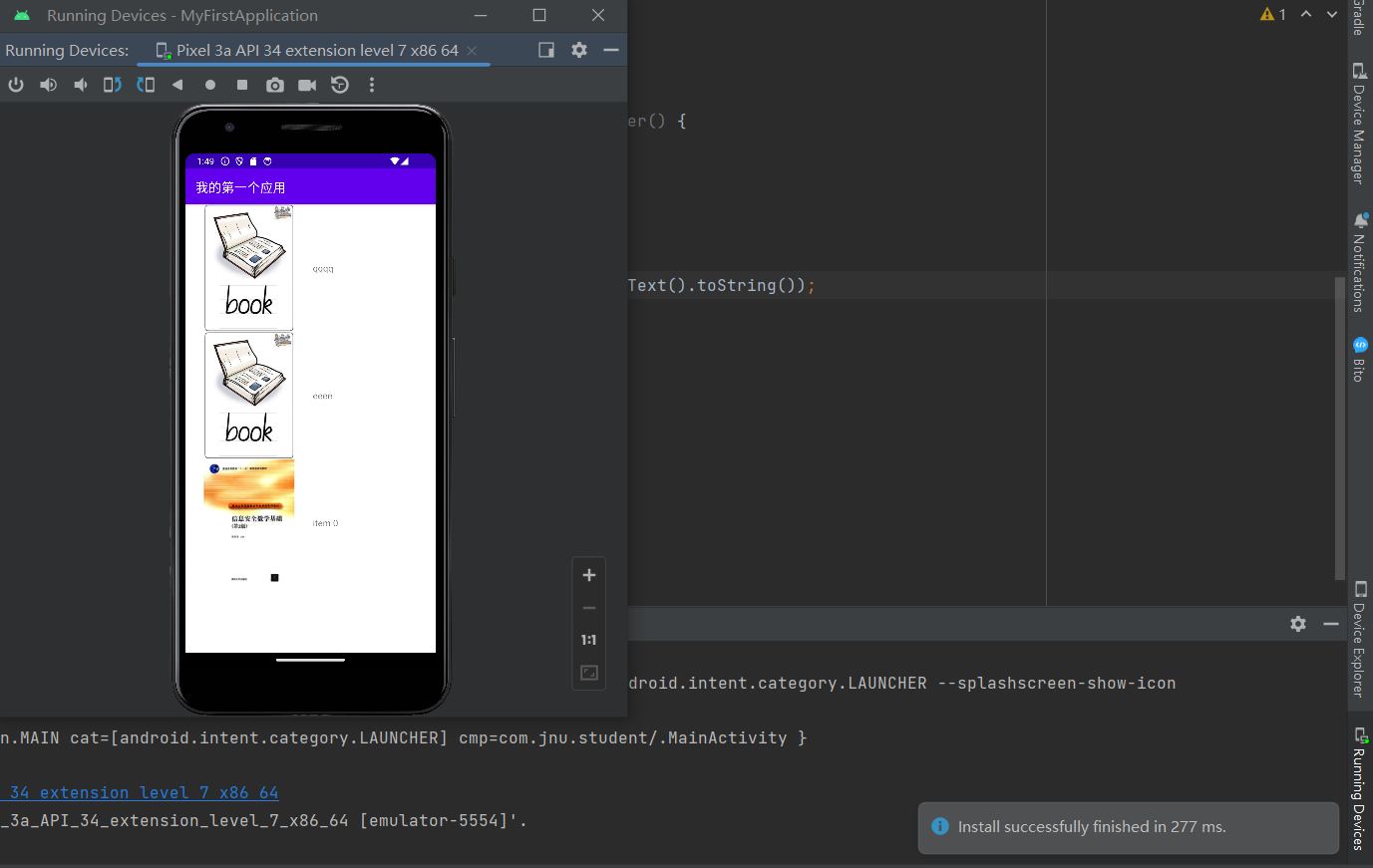
**App界面初始化：**

****

**app：界面修改：**

****

**app重启并保存数据：**

****

**五、实验总结**

通过本次实验，我对Activity之间的跳转、数据传递和数据持久化有了更深入的了解和实际操作经验。虽然实验过程中遇到了一些问题，但我们成功地解决了它们并获得了满意的结果。

在未来的学习和开发过程中，我还将进一步了解和掌握更多与Android开发相关的技术和知识点，通过实际项目来应用和巩固所学知识，提高开发能力和项目质量。