西南交通大學
Southwest Jiaotong University

移动商务应用开发

第4章 Activity与Intent

学习内容

4.1 认识Activity

4.2 Activity生命周期

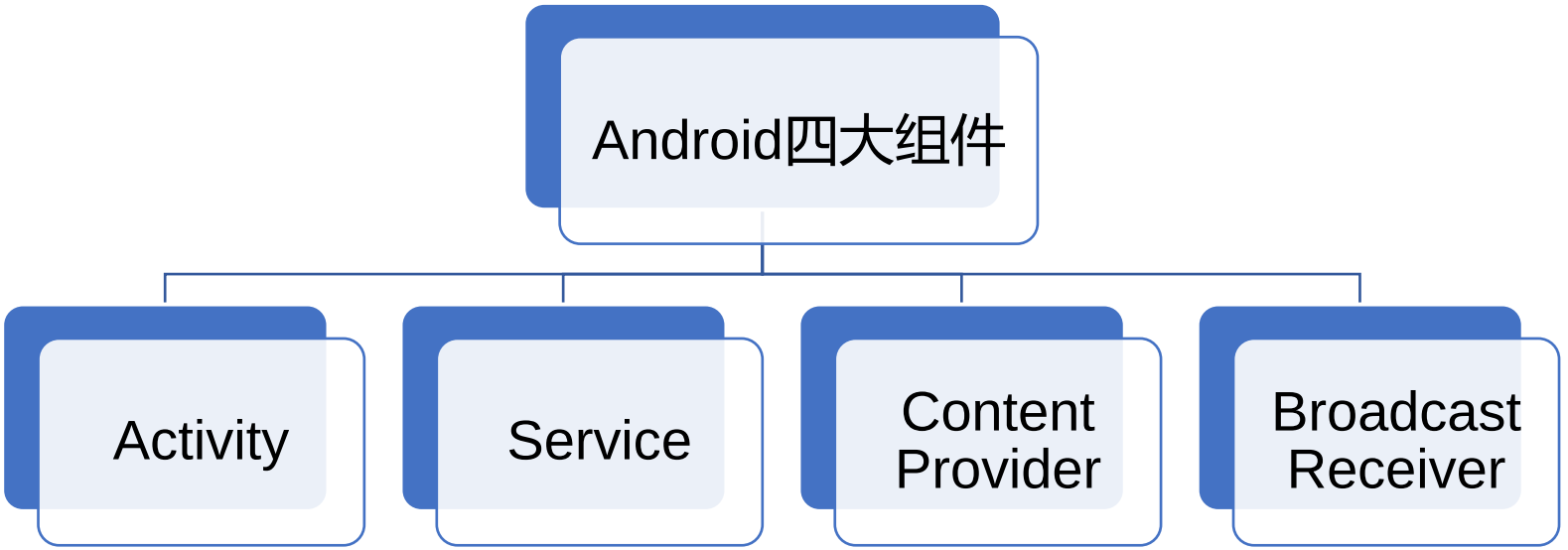
4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递

主要内容

Android 四大组件介绍



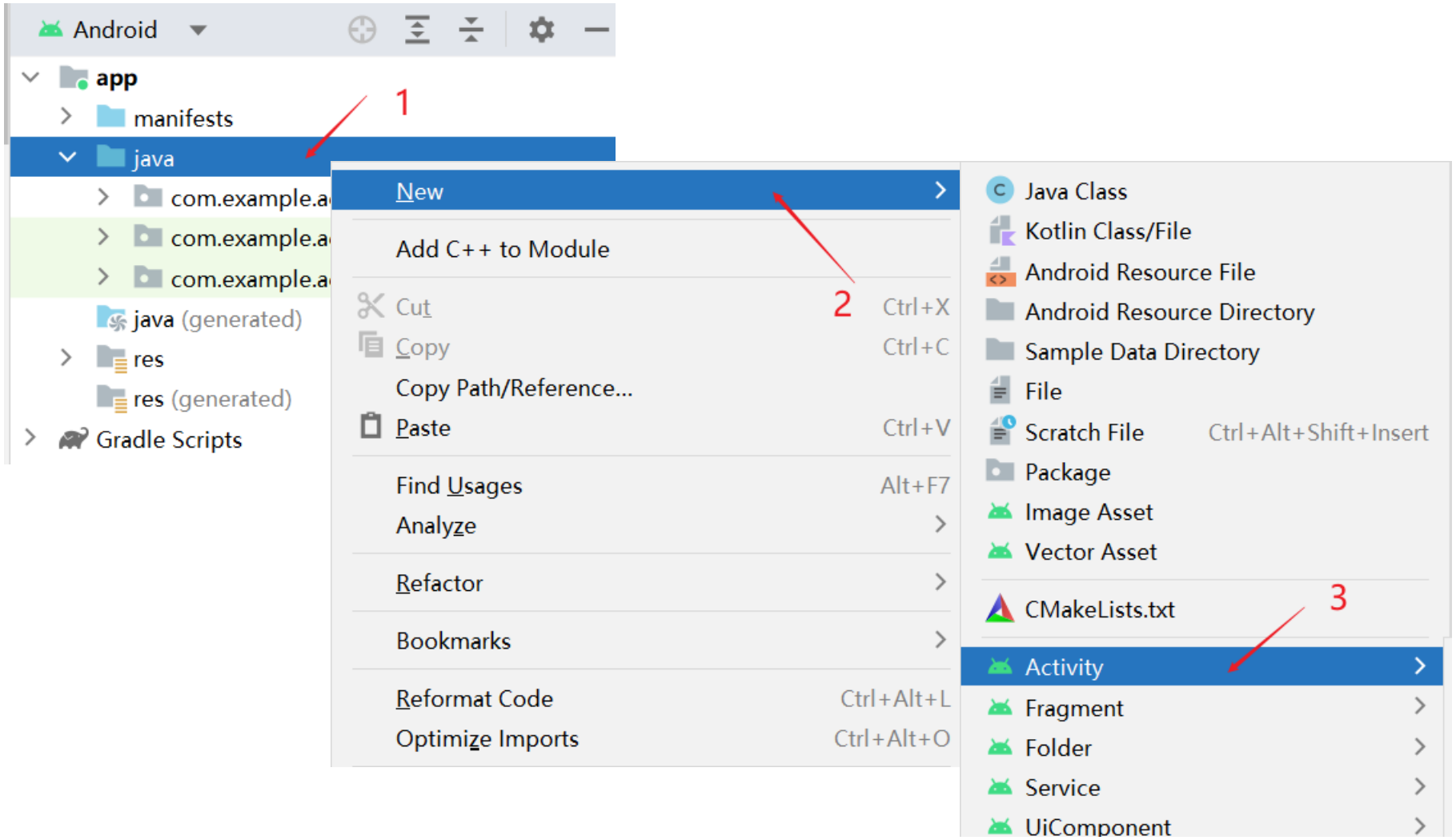
Android 四大组件介绍

- Android的四大组件：
  - Activity（活动）
    - Activity用于提供可视化用户界面并与用户交互。
  - Service（服务）
    - Service没有用户界面，也不需要和用户交互，能够长期在后台运行。
  - Content Provider（内容提供者）
    - 用于多个应用间共享数据。
  - Broadcast Receiver（广播接收器）
    - 用来接收来自系统和应用中的广播。
- 所有组件的使用都需要在清单文件中进行注册。

Activity基础

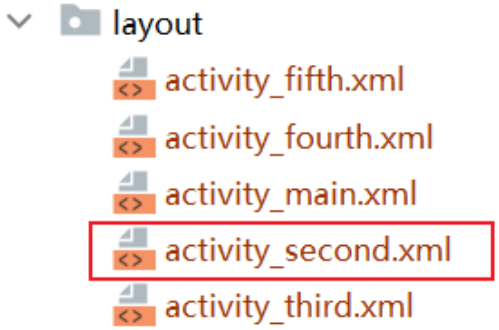
- Activity用于提供可视化用户界面并与用户交互。
- 一个Android应用中可以包括**多个**Activity。
  - 例如：电子邮件应用
  - 虽然多个Activity之间可能存在交互，但每个Activity都是**相对独立**的。
- Activity是应用与用户交互的入口点。
  - 当一个应用调用另一个应用时，实际上会调用另一个应用中的某个Activity，而不是整个应用。

创建Activity


Android
app
manifests
java
com.example.a
com.example.a
com.example.a
java (generated)
res
res (generated)
Gradle Scripts
1
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
2
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
CMakeLists.txt
3
Activity
Fragment
Folder
Service
UiComponent

创建Activity

- Android Studio创建Activity的步骤：
  1. 生成了一个继承Activity的**java文件**
  2. 生成了一个对应的**布局文件**
  3. 在**清单文件**中**注册**了该Activity。




```
<activity
    android:name=".SecondActivity"
    android:exported="true" />
```

学习内容

4.1 认识Activity

4.2 Activity生命周期

4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递

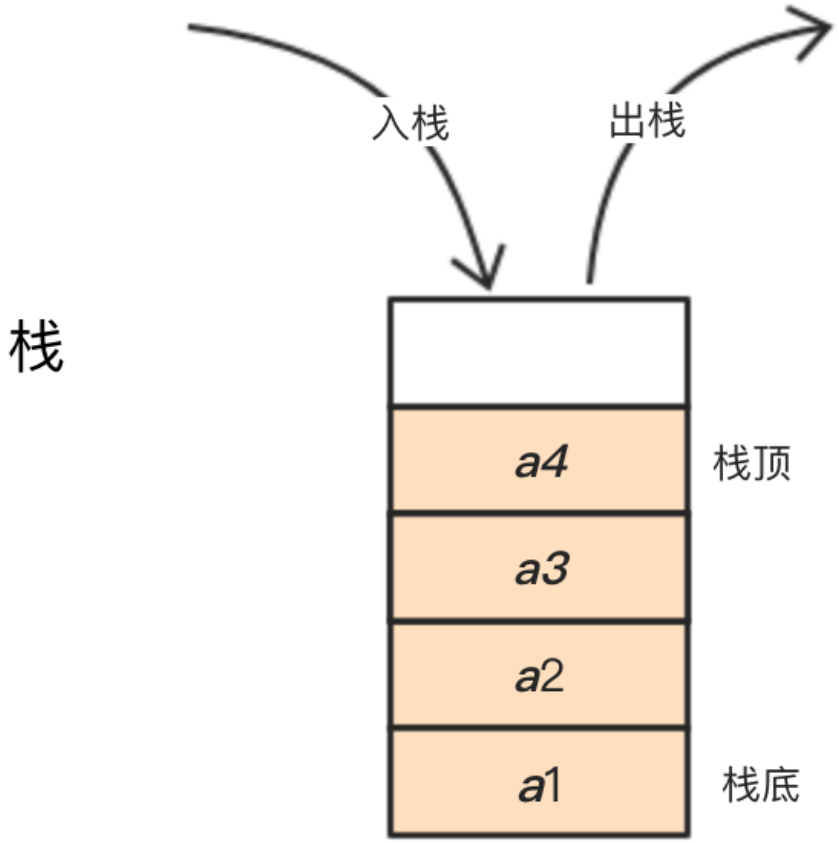
主要内容

Activity生命周期

- Activity具有生命周期。
- 当用户打开、退出和返回应用时，应用中的Activity实例会在其生命周期的不同状态间转换。
- Activity类提供了许多回调方法来让Activity知晓状态发生了改变。
  - 这些回调方法何时被调用，都是**由Android系统决定**的。

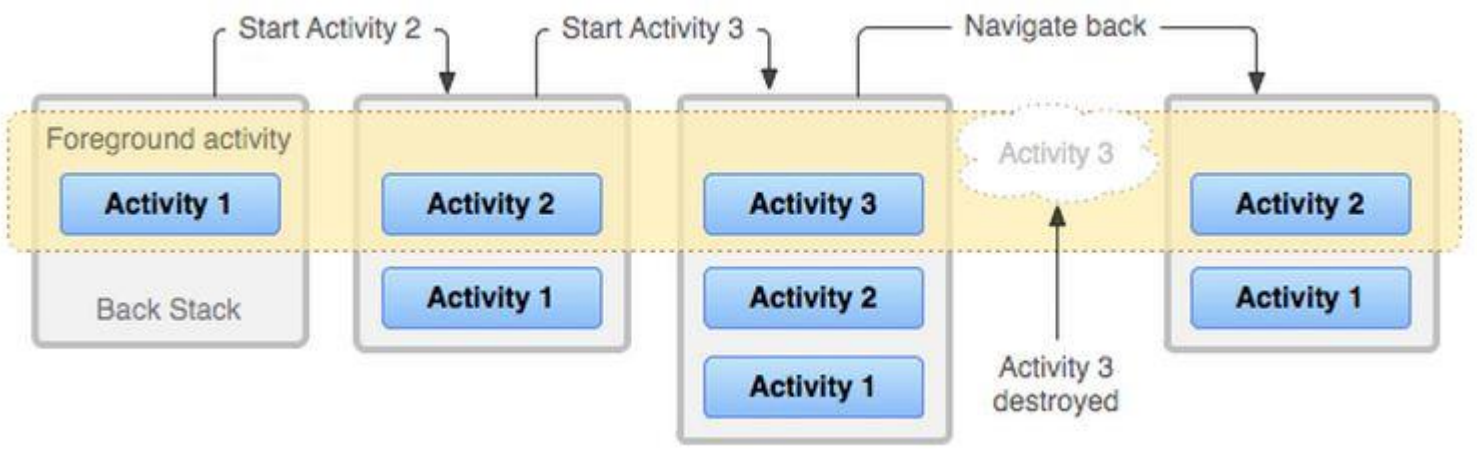
Activity任务栈

- 栈有什么特点?



Activity任务栈

- Android通过任务栈来管理多个Activity。任务栈中**只有一个栈顶。**
  - 当启动一个新的Activity时，新的Activity入栈。
  - 当关闭一个Activity时，该Activity出栈。



Activity任务栈

- 在任务栈中，一个Activity可能：
  - 处在**栈顶**。我们称此时的Activity处在前台、对用户可见、获得焦点、可以和用户交互。
  - 处在**非栈顶**。我们称此时的Activity处在后台、对用户不可见、失去焦点、无法和用户交互。

Activity生命周期的状态

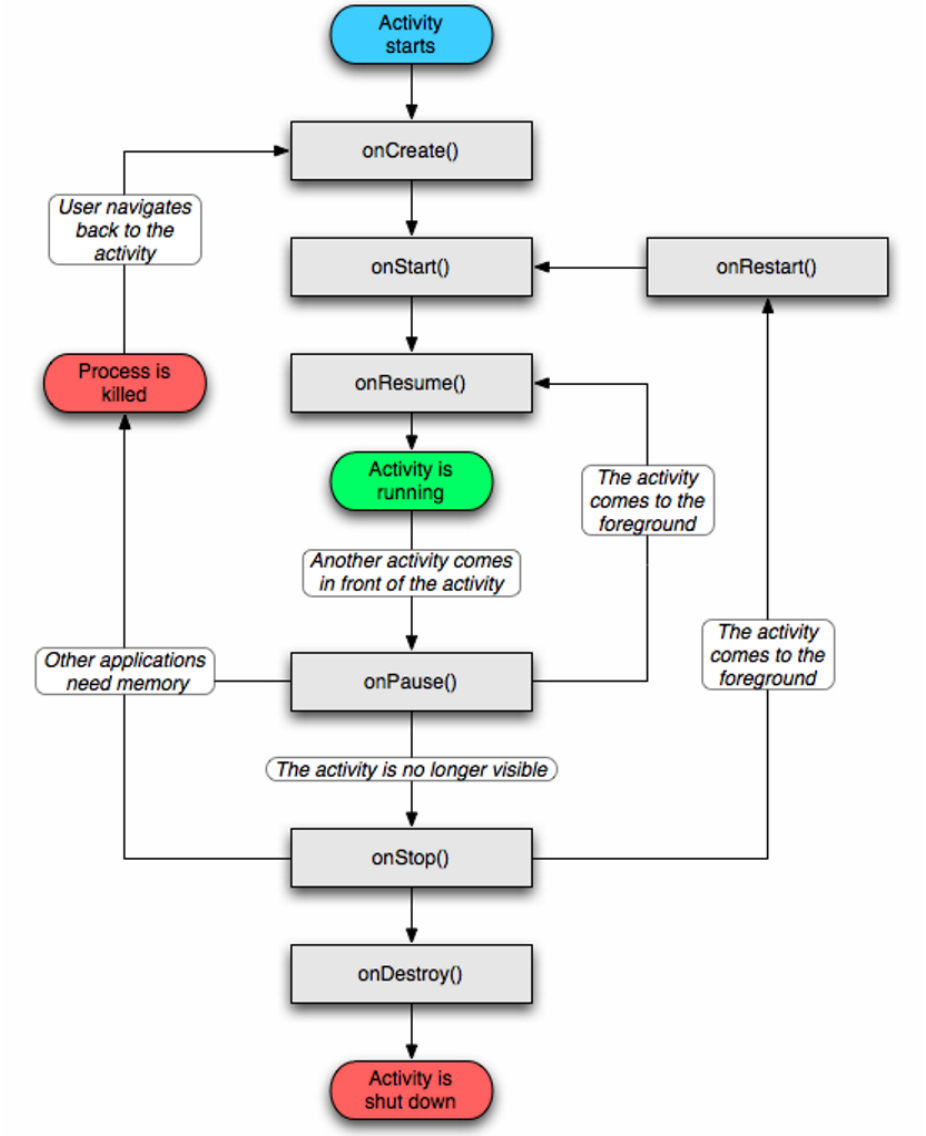
- **Starting（启动状态）**
  - 这个状态很短暂，当Activity启动后便会迅速进入运行状态。
- **Running（运行状态）**
  - 此时，Activity处在**栈顶**，位于前台，具有焦点，对用户可见，可以和用户交互。
- **Paused（暂停状态）**
  - 此时，Activity没有焦点，无法和用户交互，但对用户仍然（部分）可见。通常出现在新的Activity还没有完全覆盖屏幕的时候。

Activity生命周期的状态

- **Stopped（停止状态）**
  - 此时，Activity处在**非栈顶**，处在后台，没有焦点，对用户不可见，无法和用户交互。当系统内存不足时，这种状态下的Activity很容易被“杀掉”（Process killed）。
- **Destroyed（销毁状态）**
  - 此时，Activity已销毁，将被清理出内存。

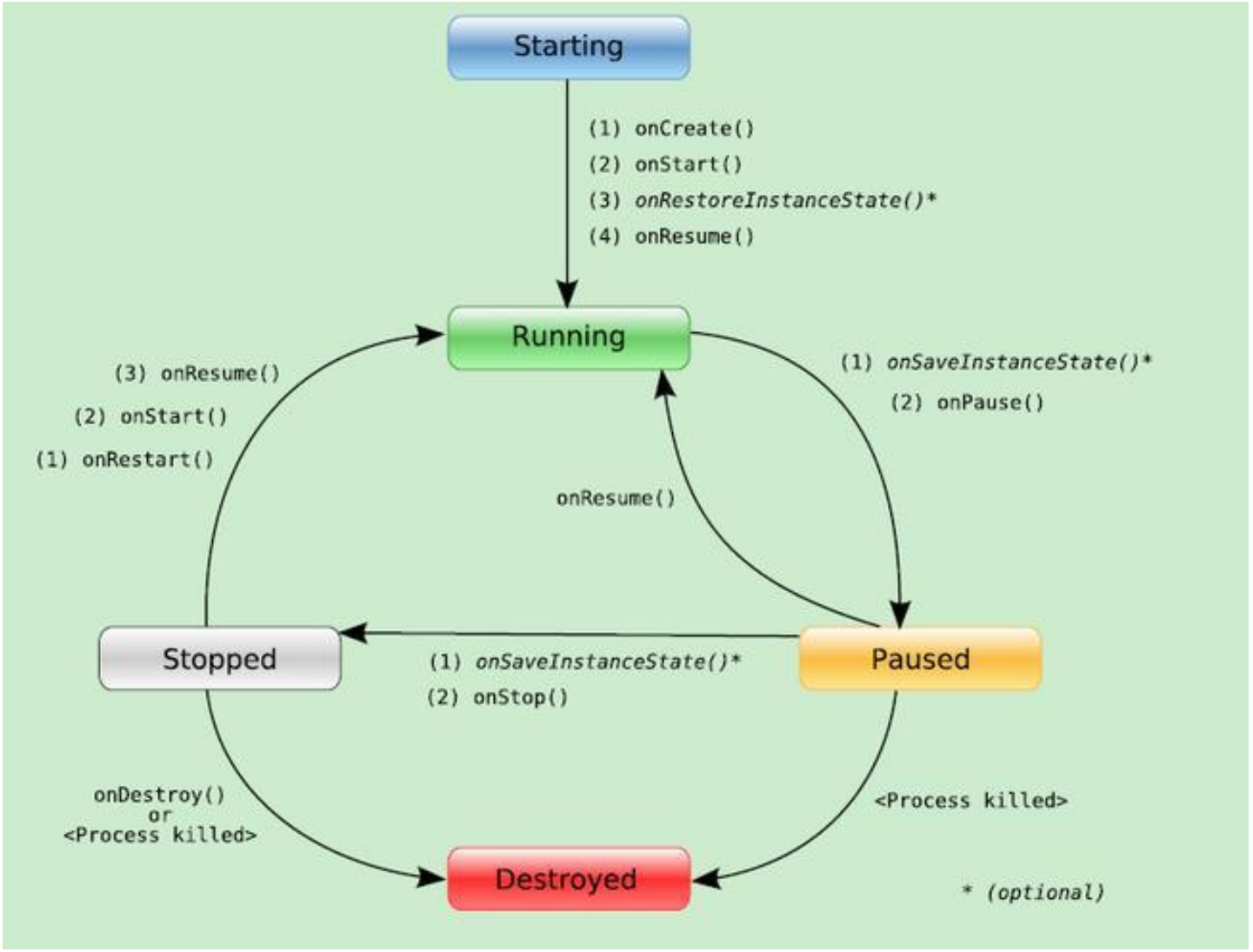
◆Activity主要在**Running、Paused、Stopped**这三个状态间转换。

Activity生命周期的回调方法



- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onRestart()
- onDestory()

Activity生命周期的状态和回调方法



Activity生命周期的回调方法

- 通过重写回调方法和输出日志观察回调方法的调用时机

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.i(TAG, msg: "onCreate: " + className);
}

@Override
protected void onStart() {
    super.onStart();
    Log.i(TAG, msg: "onSta
}

@Override
protected void onResume() {
    super.onResume();
    Log.i(TAG, msg: "onRes
}
```

Pixel 2 API 28 (emulator-5554) Android 9, API 28 | myTag

```
18:13:42.396  I  onCreate: MainActivity
18:13:42.590  I  onStart: MainActivity
18:13:42.593  I  onResume: MainActivity
```

启动Activity时调用的回调方法

常见操作下的状态转换和回调方法

- 启动Activity（入栈）：

  onCreate -> onStart -> onResume

  启动状态 -> 运行状态

- 切换到另一个Activity（例如：点击Home键，栈顶到非栈顶）：

  onPause -> onStop

  运行状态 -> 暂停状态 -> 停止状态

- 从另一个Activity回到该Activity（从非栈顶到栈顶）：

  onRestart -> onStart -> onResume

  停止状态 -> 运行状态

常见操作下的状态转换和回调方法

- 准备切换到另一个Activity，切到一半又回到该Activity：

onPause -> onResumed

运行状态 -> 暂停状态 -> 运行状态

- 退出该Activity（例如：点击Back键、手动杀掉进程，出栈）：

onPause -> onStop -> onDestroy

运行状态 -> 暂停状态 -> 停止状态 -> 销毁状态

常见操作下的状态转换和回调方法

- 退出后再回到程序：

onCreate -> onStart -> onResume

启动状态 -> 运行状态

> 注：重新创建了一个新的Activity实例

- 翻转屏幕/切换语言：(在允许屏幕翻转的情况下)

onPause -> onStop -> onDestroy -> onCreate -> onStart -> onResume

> 注：销毁并重新创建了一个新的Activity实例

练习1

- 熟悉生命周期的状态和回调方法
  - 重写所有回调方法，添加Log
  - 观察各操作下被调用的方法

学习内容

4.1 认识Activity

4.2 Activity生命周期

4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递

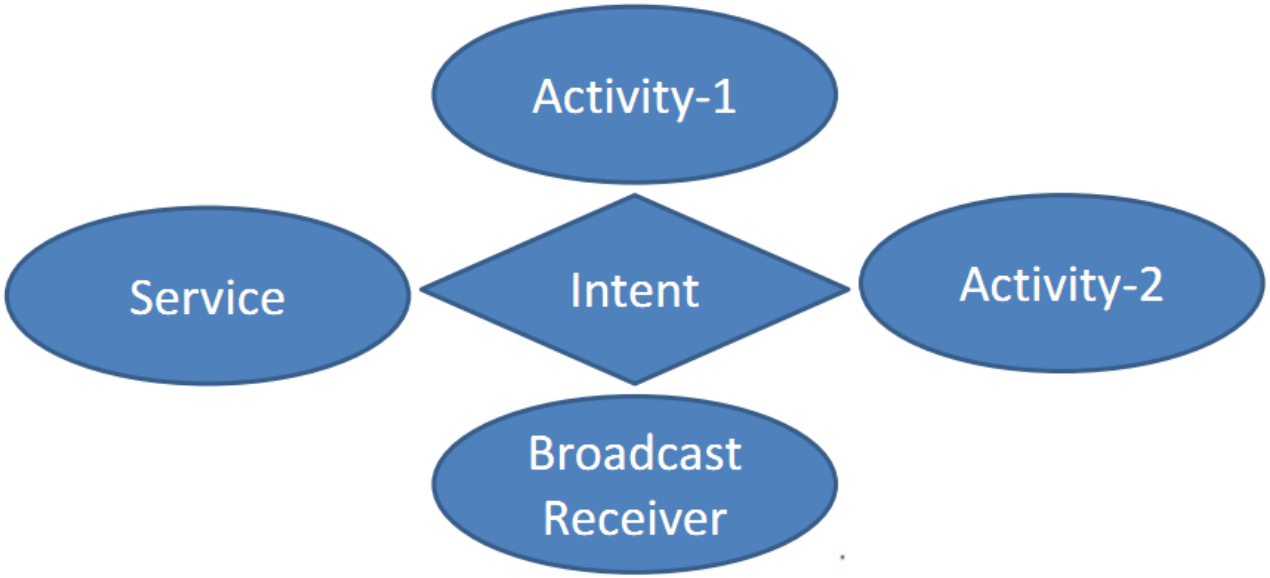
主要内容

Intent

- Android中提供了Intent机制来协助应用间的交互与通讯
- Intent负责对应用中的一次操作进行描述，Android则根据此Intent的描述，负责找到对应的组件，将 Intent传递给调用的组件，并完成组件的调用。
- Intent不仅可用于应用程序之间，也可用于应用程序内部的**activity、service和broadcast receiver**之间的交互。

Intent

- 我们通常称**Intent是连接四大组件的桥梁。**
  - 在四大组件中，Activity、Service和Broadcast Receiver之间是通过Intent进行通信的，而Content Provider本身就是一种通信机制，不需要通过Intent。



显式Intent

- 显式Intent：通过组件名（Component）**明确**指定需要启动的目标组件。
  - 通常会用显式Intent来启动同一个应用中的组件。
- 如何使用显式Intent：

```
1  Intent intent = new Intent(this, SecondActivity.class);
2  startActivity(intent);
```

隐式Intent

- 隐式Intent：**不明确**指定需要启动的目标组件，而是通过设置Action、Data、Category来让系统来筛选出合适的组件。
  - 通过会用隐式Intent来启动其他应用中的组件。
- 如何使用隐式Intent（以Activity为例）：
  - 首先，需要在清单文件中为被调用的Activity添加Action、Data和Category。

```
1  <activity
2      android:name=".ThirdActivity"
3      android:exported="true" >
4      <intent-filter>
5          <action android:name="com.example.activity.START_ACTIVITY"/>
6          <category android:name="android.intent.category.DEFAULT"/>
7      </intent-filter>
8  </activity>
```

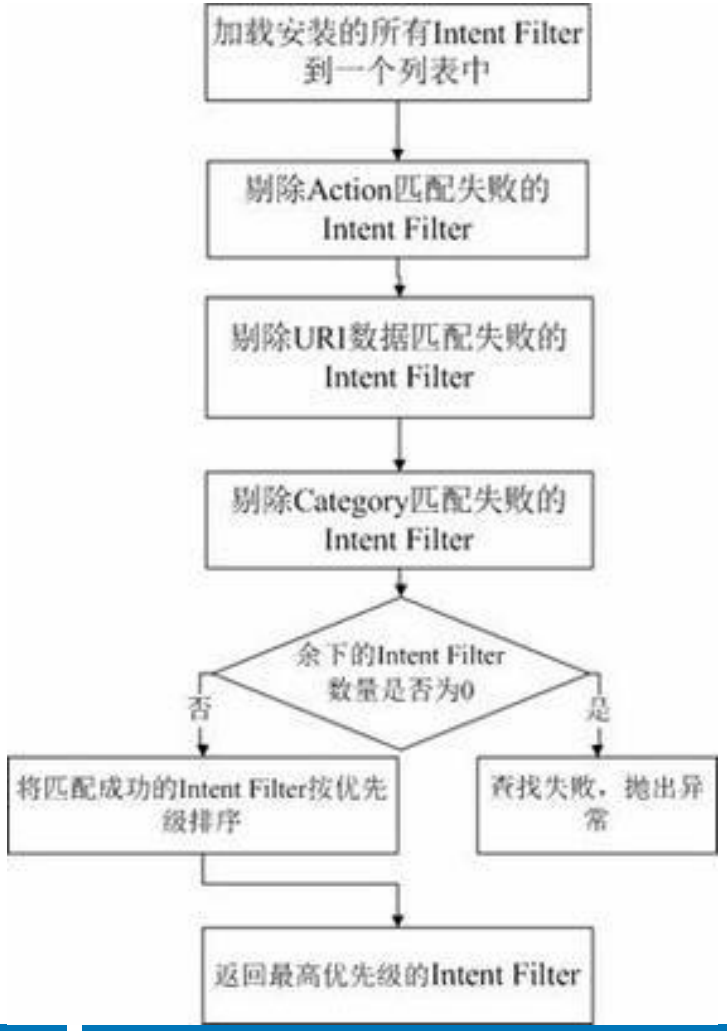
隐式Intent

- 如何使用隐式Intent（以Activity为例）：
  - 首先，需要在清单文件中为被调用的Activity添加Action、Data和Category。
  - 然后，就可以在其他Activity中通过action name找到这个Activity。

```
1  Intent intent = new Intent();
2  intent.setAction("com.example.activity.START_ACTIVITY");
3  startActivity(intent);
```

隐式Intent

- Android根据隐式Intent的描述匹配组件的过程：



Intent的属性

- Intent具有以下属性：
  - Component（组件）、Action（动作）、Data（数据）、Category（类别）、Type（数据类型）、Extra（扩展信息）、Flag（标志位）。
- 显示Intent：Component（组件）
- 隐式Intent：其他属性，主要为Action（动作）、Data（数据）、Category（类别）
  - Action：用来指定Intent要执行的动作。
  - Data：表示动作要操纵的数据。
  - Category：用来表示哪种类型的组件来处理这个Intent。

Intent属性：Action

- 列举一些常见的和Activity相关的action：

类型	描述
ACTION_MAIN	应用程序入口
ACTION_VIEW	显示数据
ACTION_EDIT	显示可编辑的数据
ACTION_PICK	选择一条数据并返回
ACTION_DELETE	删除数据
ACTION_DAIL	显示拨号面板
ACTION_SENDTO	发送短信到指定的人
ACTION_SEARCH	执行搜索
ACTION_WEB_SEARCH	执行网上搜索

Intent属性：Data

- Data属性的声明中要指定访问数据的Uri和MIME类型。
  - 示例：网址转换成Uri

```
Uri uri = Uri.parse("http://www.baidu.com");
```

字符串	描述
http://www.baidu.com	网址
content://contacts	联系人信息
tel:10010	给10010打电话
smsto:10010	给10010发短信
file://sdcard/mymusic.mp3	本地文件
geo:39.2456,116.3523	地理位置

Intent属性：Category

- 列举一些常见的Category：

类型	作用
CATEGORY_DEFAULT	把一个组件设为可被隐式启动的
CATEGORY_LAUNCHER	把一个action设置为在顶级执行
CATEGORY_BROWSABLE	当Intent指向网络相关时，必须要添加这个类别
CATEGORY_HOME	使Intent指向Home界面

学习内容

4.1 认识Activity

4.2 Activity生命周期

4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递

主要内容

启动Activity

- 通过显式Intent启动：

```
1 // 启动SecondActivity
2 Intent intent = new Intent(this, SecondActivity.class);
3 startActivity(intent);
```

- 通过隐式Intent启动：

```
1 Intent intent = new Intent();
2 intent.setAction("com.example.activity.START_ACTIVITY");
3 startActivity(intent);
```

启动Activity

- MainActivity启动SecondActivity时的状态变化和回调方法：

① MainActivity调用onPause
MainActivity进入**已暂停**状态。

② SecondActivity依次调用onCreate、onStart、onResume
SecondActivity进入**运行状态**。

③ MainActivity调用onStop
MainActivity进入**已停止**状态。

通过隐式Intent启动系统应用

- 打开拨号页面

```
1 // 显示给10010拨打电话的界面
2 Uri uri = Uri.parse("tel:10010");
3 Intent intent = new Intent(Intent.ACTION_DIAL, uri);
4 startActivity(intent);
```

通过隐式Intent启动系统应用

- 发送短信

```
1  // 显示给10010发送短信的界面
2  Uri uri = Uri.parse("smsto:10010");
3  Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
4  intent.putExtra("sms_body", "Hello"); // key-value
5  startActivity(intent);
```

- 打开网页

```
1  // 打开百度主页
2  Uri uri = Uri.parse("https://www.baidu.com");
3  Intent intent = new Intent(Intent.ACTION_VIEW, uri);
4  startActivity(intent);
```

关闭Activity

- 方法一：调用finish()
- 方法二：点击Back键
- SecondActivity时（回到MainActivity）的状态变化和回调方法：
  ① SecondActivity调用onPause
  SecondActivity进入**已暂停**状态。
  ② MainActivity依次调用onRestart、onStart、onResume
  MainActivity进入**运行状态**。
  ③ SeconActivity依次调用onStop、onDestroy
  SecondActivity进入**已销毁**状态。

学习内容

4.1 认识Activity

4.2 Activity生命周期

4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递

主要内容

Activity间的数据传递

- 实际开发中，一个Activity启动另一个Activity时通常需要传输数据，包括**传入数据**和**传回数据**。
- 可以通过Intent实现Activity间的数据传递。

方法	描述
putExtras(Bundle data)	放入Bundle数据包
getExtras()	取出Bundle数据包
putString(String str)	放入String
getString()	取出String

Activity传入数据

- MainActivity如何向FourthActivity传入数据?
  - 首先，在MainActivity中：
    ① 创建一个Intent，
    ② 将数据存入到Bundle数据包中，
    ③ 将Bundle数据包附加在Intent上，
    ④ 通过startActivity(Intent)的方式启动FourthActivity。
  - 然后，在FourthActivity中：
    ① 接收Intent，
    ② 从Intent中取出Bundle数据包，
    ③ 从Bundle数据包中提取数据。

Activity传入数据

- MainActivity中：

```java
// 创建Intent
Intent intent = new Intent(this, FourthActivity.class);
// 创建Bundle
Bundle bundle = new Bundle();
bundle.putString("username", username); // key-value
bundle.putString("password", password); // key-value
// 将Bundle附在Intent上
intent.putExtras(bundle);
// 调用startActivity(Intent)
startActivity(intent);
```

Activity传入数据

- FourthActivity中：

```
1  // 接收Intent
2  Intent intent = getIntent();
3  // 获取 Bundle
4  Bundle bundle = intent.getExtras();
5  // 解析Bundle中数据
6  String username = bundle.getString("username");
7  String password = bundle.getString("password");
8  // 显示获取的数据
9  textView.setText("您的用户名是: " + username);
10 textViewPassword.setText("您的密码是: " + password);
```

Activity传回数据

- MainActivity如何令FifthActivity传回数据?
  - 首先，在MainActivity中：
    - ① 通过registerForActivityResult()创建一个ActivityResultLauncher，
    - ② 创建一个Intent，
    - ③ 通过launcher.launch(Intent)的方式启动FifthActivity。
  - 然后，在FifthActivity中：
    - ① 创建一个Intent，
    - ② 将数据或Bundle数据包附在Intent上，
    - ③ 通过setResult(RESULT_OK, intent)传回数据，
    - ④ 结束FfithActivity。

Activity传回数据

- MainActivity中：

```
// 通过registerForActivityResult创建一个ActivityResultLauncher
ActivityResultLauncher<Intent> launcher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
    if (result.getResultCode() == RESULT_OK) {
        Intent intent = result.getData();
        if (intent != null) {
            int num = intent.getIntExtra("number", 0);
            textViewNumber.setText("输入的数字: " + num);
        }
    }
});

// 创建Intent
Intent intent = new Intent(MainActivity.this, FifthActivity.class);
// 通过ActivityResultLauncher.lanunch(Intent)启动Activity
launcher.launch(intent);
```

Activity传回数据

- FifthActivity中：

```
1  // 创建Intent
2  Intent intent = new Intent();
3  // 将数据附在Intent上
4  int num = Integer.parseInt(str);
5  intent.putExtra("number", num);
6  // 通过setResult返回数据
7  setResult(RESULT_OK, intent);
8  // 结束Activity
9  finish();
```

调用系统应用并传回数据

- 拍照并返回照片

```
ActivityResultLauncher<Intent> takePhotoPreviewLauncher =
registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == RESULT_OK) {
            Bundle bundle = result.getData().getExtras();
            if (bundle != null) {
                // 如果不指定图片保存地址，那么将会返回照片的缩略图
                Bitmap bitmap = (Bitmap) bundle.get("data");
                imageView.setImageBitmap(bitmap);
            }
        }
    }
});
buttonTakePhoto.setOnClickListener(v -> {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    takePhotoPreviewLauncher.launch(intent);
});
```

调用系统应用并传回数据

- 选择并返回一张图片

```
ActivityResultLauncher<Intent> selectImageLauncher =
registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == RESULT_OK) {
            Uri uri = result.getData().getData();
            if (uri != null) {
                imageView.setImageURI(uri);
            }
        }
    }
});
buttonSelectImage.setOnClickListener(v -> {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    selectImageLauncher.launch(intent);
});
```

调用系统应用并传回数据

- 获取联系人信息

```
ActivityResultLauncher<Intent> pickContactLauncher =
registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
    if (result.getResultCode() == RESULT_OK) {
        Uri uri = result.getData().getData();
        Cursor cursor = getContentResolver().query(uri, null, null, null, null);
        if (cursor != null) {
            cursor.moveToFirst();
            int nameColumnIndex = cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
            int phoneNumberColumnIndex =
cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
            String name = cursor.getString(nameColumnIndex);
            String phoneNumber = cursor.getString(phoneNumberColumnIndex);
            String str = "联系人姓名: " + name + "\r\n电话: " + phoneNumber;
            Toast.makeText(FifthActivity.this, str, Toast.LENGTH_SHORT).show();
            cursor.close();
        }
    }
});
```

调用系统应用并传回数据

- 获取联系人信息

```
20  buttonContact.setOnClickListener(v -> {
21      Intent intent = new Intent(Intent.ACTION_PICK);
22      //从有电话号码的联系人中选取
23      intent.setType(ContactsContract.CommonDataKinds.Phone.CONTENT_TYPE);
24      pickContactLauncher.launch(intent);
25  });
```

练习2

- 掌握创建Activity的方法
  - 创建多个Activity
  - 观察创建Activity后的变化
- 熟悉Activity的启动与关闭
  - 通过显示Intent启动SecondActivity
  - 通过finish()关闭SecondActivty
  - 通过隐式Intent启动ThirdActivity
  - 通过隐式Intent启动系统应用

练习2


2:04
LTE
启动SecondActivity：演示显式Intent
启动ThirdActivity：演示隐式Intent
请输入用户名
请输入密码
启动FourthActivity：演示传入数据
启动FifthActivity：演示数据回传
显示传回的数字

练习3

- 掌握Activity的传入数据
  - 创建FouthActivity
  - 利用Intent传入用户名和密码，并显示

练习4

- 掌握Activity的传回数据
  - 创建FifthActivity
  - 令FifthActivity传回一个数字
  - 打开系统应用并传回数据

课程总结

4.1 认识Activity

4.2 Activity生命周期

4.3 Intent

4.4 Activity的启动与关闭

4.5 Activity的数据传递