Android移动开发基础案例教程

第3章 Activity

- Activity生命周期
- Activity启动模式
- Activity之间的跳转
- Activity中的数据传递

作业点评

- 请简要说明LogCat区域中的5种字母所代表的含义。
- 请简要说明Android中有几种布局，以及每种布局的特点。

预习检查

- Activity的生命周期
- Android中跳转Activity有几种方法

学习目标

1 Activity生命周期

Acitvity之间的跳转

2 Activity中的数据传递

3 Activity启动模式

主讲内容

Activity 是一个应用组件，用户可与其提供的屏幕进行交互，以执行拨打电话、拍摄照片、发送电子邮件或查看地图等操作。 每个 Activity 都会获得一个用于绘制其用户界面的窗口。窗口通常会充满屏幕，但也可小于屏幕并浮动在其他窗口之上

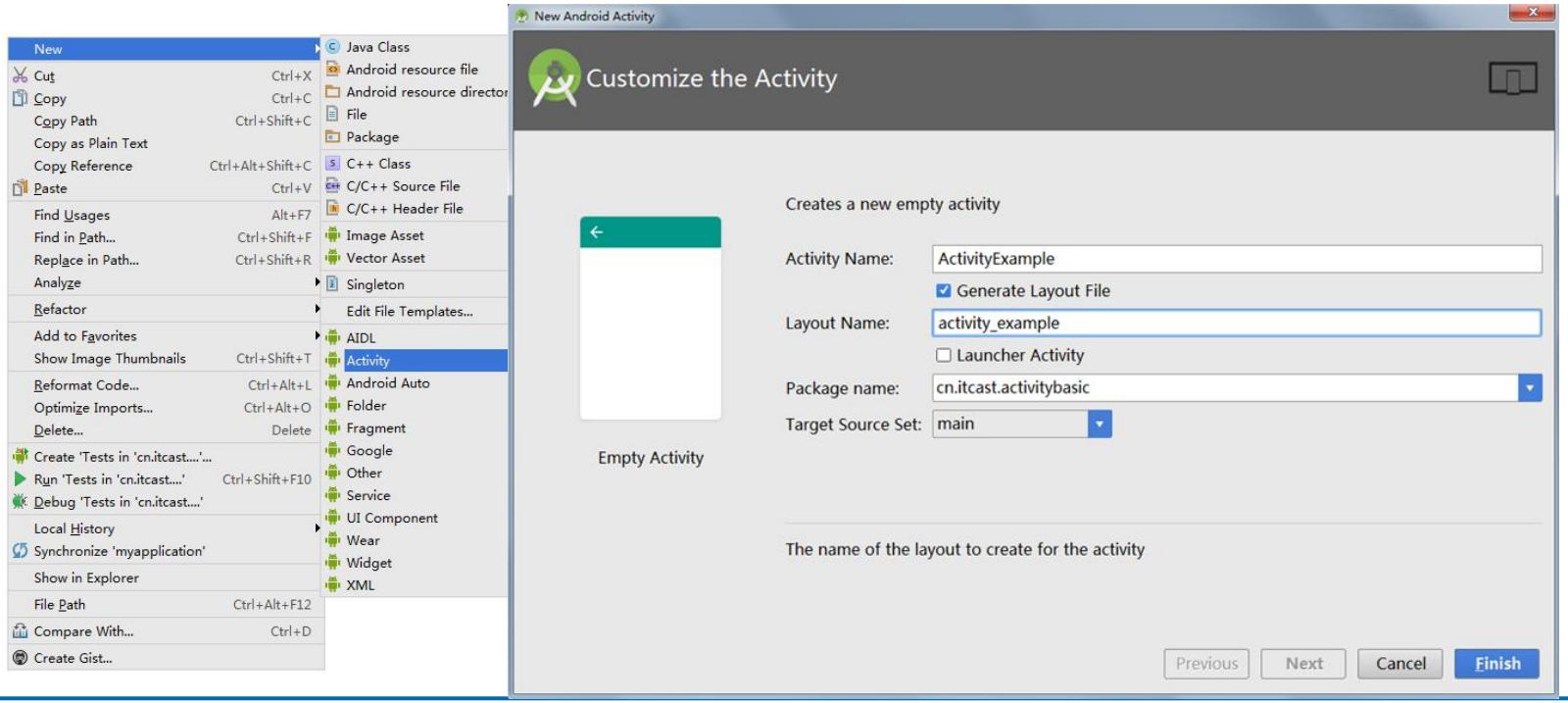
一个应用通常由多个彼此松散联系的 Activity 组成。 一般会指定应用中的某个 Activity 为“主”Activity，即首次启动应用时呈现给用户的那个 Activity。 而且每个 Activity 均可启动另一个 Activity，以便执行不同的操作。 每次新 Activity 启动时，前一 Activity 便会停止，但系统会在堆栈（“返回栈”）中保留该 Activity。 当新 Activity 启动时，系统会将其推送到返回栈上，并取得用户焦点。 返回栈遵循基本的“后进先出”堆栈机制，因此，当用户完成当前 Activity 并按“返回”按钮时，系统会从堆栈中将其弹出（并销毁），然后恢复前一 Activity。

当一个 Activity 因某个新 Activity 启动而停止时，系统会通过该 Activity 的生命周期回调方法通知其这一状态变化。Activity 因状态变化—系统是创建 Activity、停止 Activity、恢复 Activity 还是销毁 Activity— 而收到的回调方法可能有若干种，每一种回调都会提供执行与该状态变化相应的特定操作的机会。 例如，停止时， Activity 应释放任何大型对象，例如网络或数据库连接。 当 Activity 恢复时，您可以重新获取所需资源，并恢复执行中断的操作。 这些状态转变都是 Activity 生命周期的一部分。

Activity的创建

两种创建方式

1）包名处点击右键选择【New】→【Activity】→【Empty Activity】选项，填写Activity信息，完成创建。



主讲内容

3.1 Activity的创建

3.2 Activity的生命周期

3.3 Activity的启动模式

3.4 Activity之间的跳转

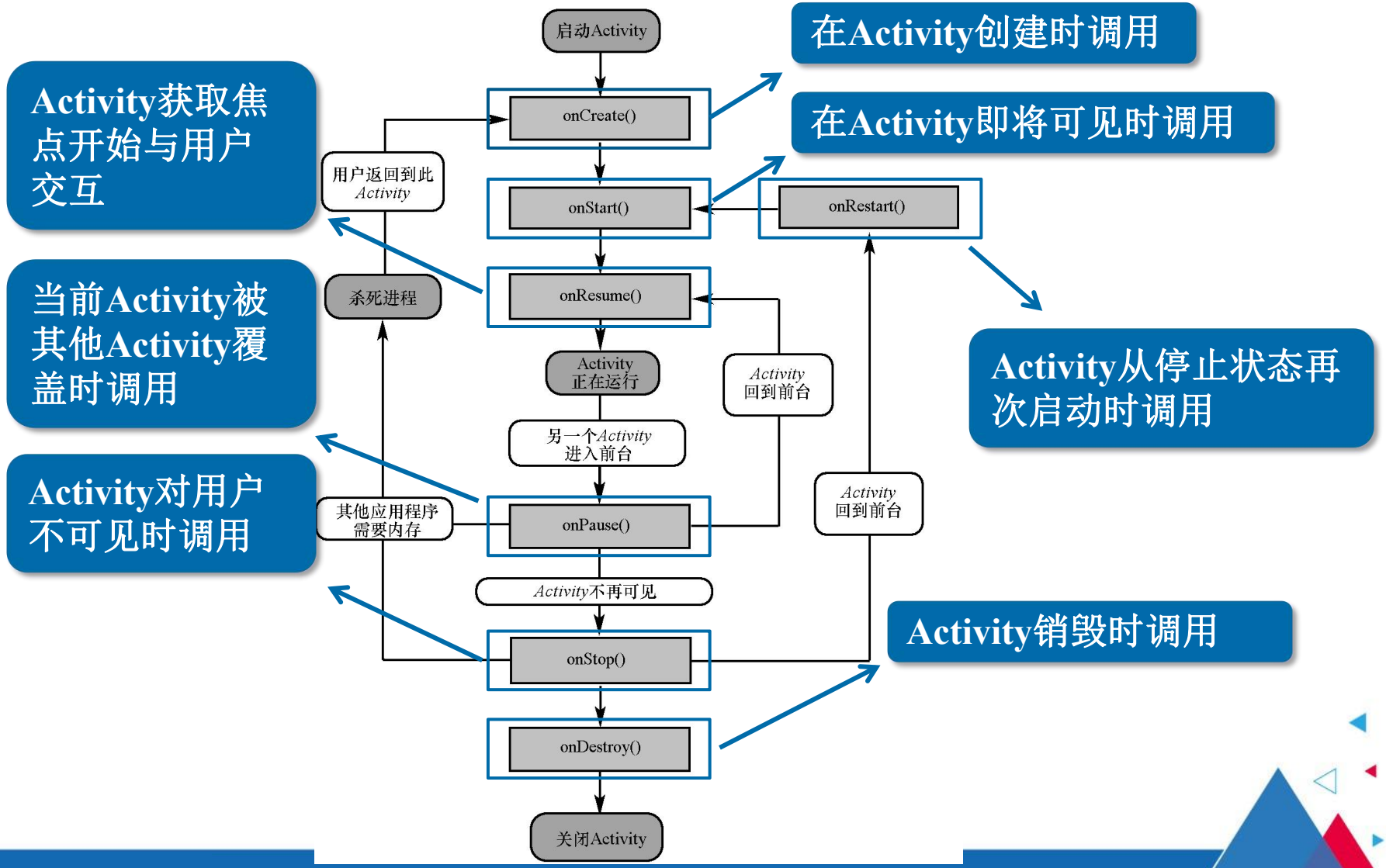
主讲内容

Speech content

3.2.1 生命周期状态

- **启动状态**：当Activity启动之后便会进入下一状态。
- **运行状态**：Activity处于屏幕最前端，可与用户进行交互。
- **暂停状态**：Activity仍然可见，但无法获取焦点，用户对它操作没有响应。
- **停止状态**：Activity完全不可见，系统内存不足时会销毁该Activity。
- **销毁状态**：Activity将被清理出内存。

3.2.2 生命周期方法


启动Activity
onCreate()
onStart()
onRestart()
onResume()
Activity
正在运行
另一个Activity
进入前台
onPause()
Activity不再可见
onStop()
onDestroy()
关闭Activity
用户返回到此
Activity
杀死进程
其他应用程序
需要内存
Activity
回到前台
Activity
回到前台
在Activity创建时调用
在Activity即将可见时调用
Activity从停止状态再次启动时调用
Activity销毁时调用
Activity获取焦点开始与用户交互
当前Activity被其他Activity覆盖时调用
Activity对用户不可见时调用

3.2.1 生命周期状态

onCreate方法

Android操作系统创建该Activity类的实例对象，对象创建完成之后，会执行到该类的onCreate方法，此onCreate方法是重写父类Activity的onCreate方法而实现的

```
public void onCreate (Bundle savedInstanceState){
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
}
```

super.onCreate(savedInstanceState)的作用是调用其父类Activity的onCreate方法来实现对界面的图画绘制工作。在实现自己定义的Activity子类的onCreate方法时一定要记得调用该方法，以确保能够绘制界面

setContentView(R.layout.main)的作用是加载一个界面。该方法中传入的参数是"R.layout.main"，其含义为R.java类中静态内部类layout的静态常量main的值，而改值是一个指向res目录下的layout子目录下的main.xml文件的标识符。因此代表着显示main.xml所定义的画面。

3.2.1 生命周期状态

onStart方法

onStart()和onResume()的区别
onStart()是activity界面被显示出来的时候执行的，用户可见，包括有一个activity在他上面，但没有将它完全覆盖，用户可以看到部分activity但不能与它交互
onResume()是当该activity与用户能进行交互时被执行，用户可以获得activity的焦点，能够与用户交互。

主讲内容

3.1 Activity的创建

3.2 Activity的生命周期

3.3 Activity的启动模式

3.4 Activity之间的跳转

Activity 的任务栈

任务栈是用来提升用户体验而设计的:
1. 程序打开时就创建了一个任务栈, 用于存储当前程序的 activity,所有的 activity 属于一个任务栈。
2. **一个任务栈包含了一个 activity 的集合**, 去有序的选择哪一个 activity 和用户进行交互:
只有在任务栈栈顶的 activity 才可以跟用户进行交互。
3. 任务栈可以移动到后台, 并且保留了每一个 activity 的状态. 并且有序的给用户列出它们的任务, 而且还不丢失它们状态信息。
4. 退出应用程序时： 当把所有的任务栈中所有的 activity 清除出栈时,任务栈会被销毁,程序退出。

Activity 的任务栈

任务栈的**缺点**:

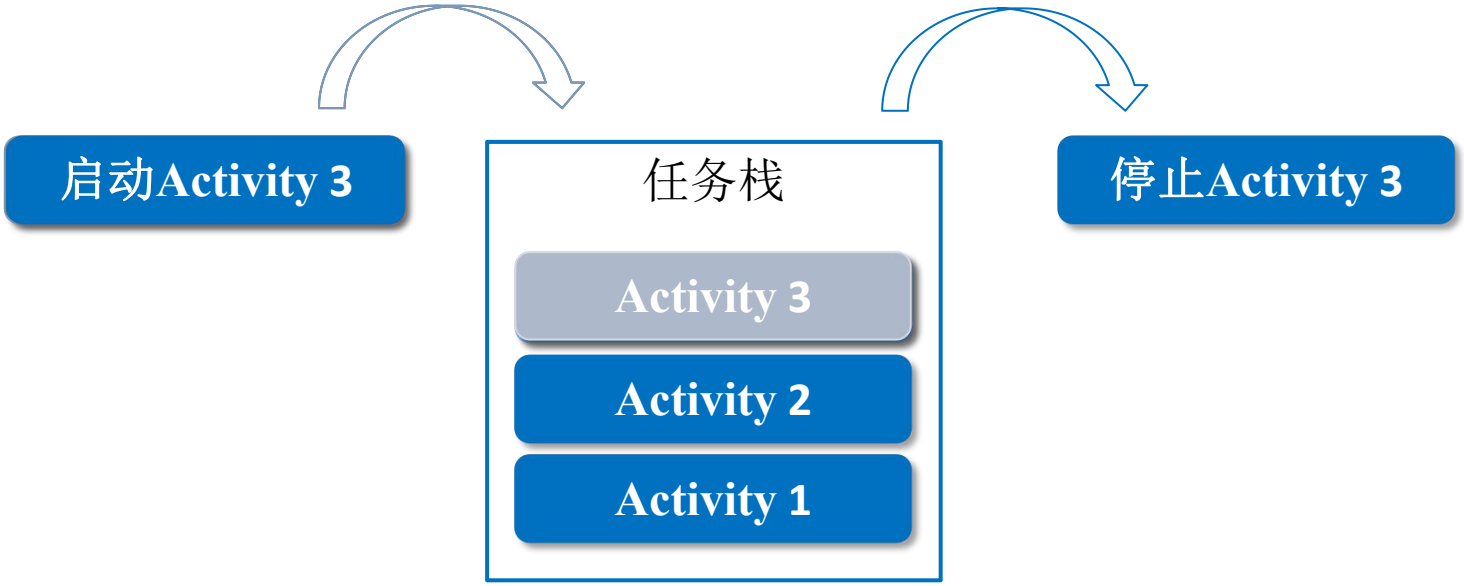
1. 每开启一次页面都会在任务栈中添加一个 Activity,而只有任务栈中的 Activity 全部清除出栈时，任务栈被销毁，程序才会退出,这样就造成了用户体验差, 需要点击多次返回才可以把程序退出了。

2. 每开启一次页面都会在任务栈中添加一个Activity还会造成数据冗余, 重复数据太多, 会导致内存溢出的问题(OOM)。

3.3.1 Android中的任务栈

栈的特点

- 栈是一种“先进后出”的数据结构。Android中，采用任务栈的形式来管理Activity。



Activity 启动模式

启动模式（launchMode）在多个 Activity 跳转的过程中扮演着重要的角色，它可以决定是否生成新的 Activity 实例，是否重用已存在的 Activity 实例，是否和其他 Activity 实例公用一个 task 里。

task 是一个具有栈结构的对象，一个 task 可以管理多个 Activity，启动一个应用，也就创建一个与之对应的 task。

Activity 一共有以下四种 launchMode：

1.standard

2.singleTop

3.singleTask

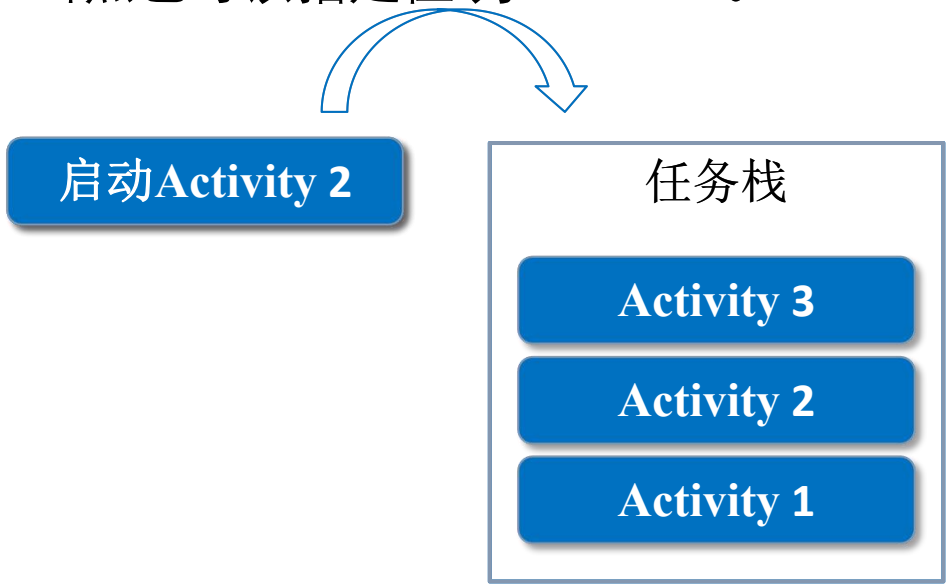
4.singleInstance

可以在 AndroidManifest.xml 配置的 android:launchMode 属性为以上四种之一即可

3.3.2 Activity的四种启动模式

standard模式

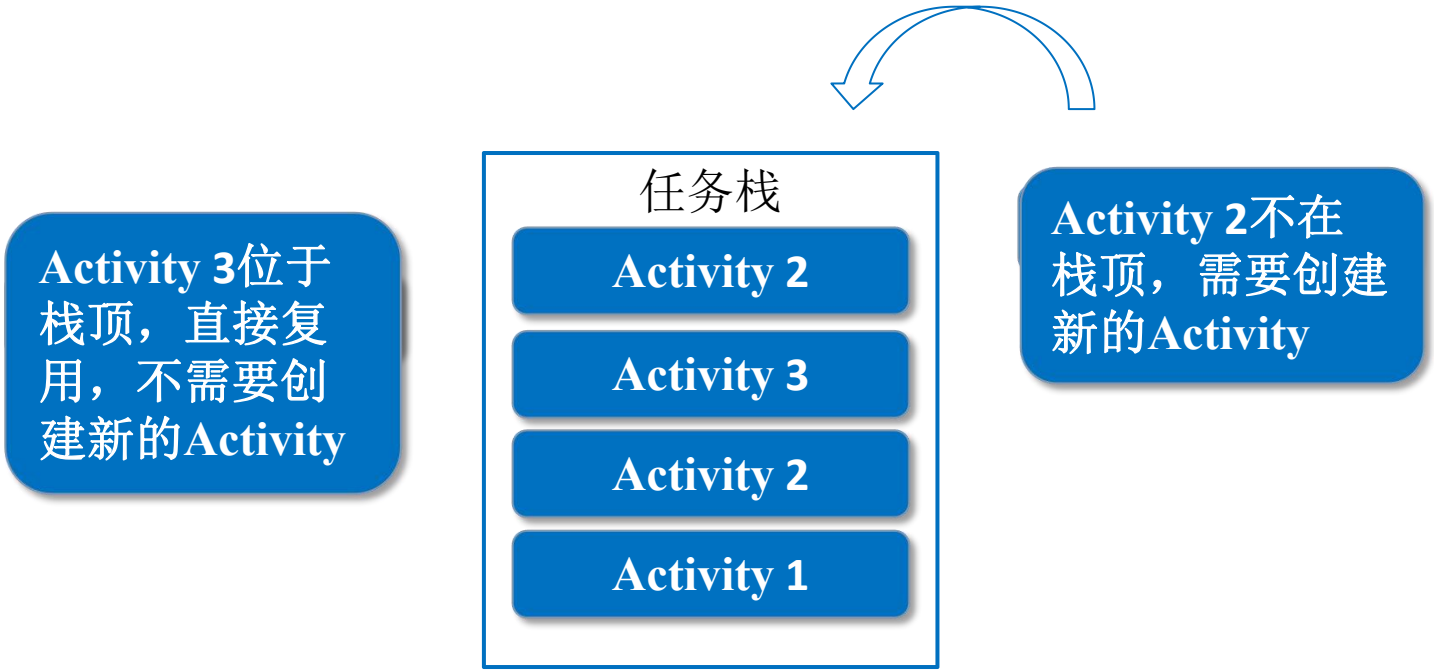
– standard模式是Activity的默认启动方式，每启动一个Activity就会在栈顶创建一个新的实例。不用为配置 android:launchMode 属性即可，当然也可以指定值 为 standard。



3.3.2 Activity的四种启动模式

singleTop模式

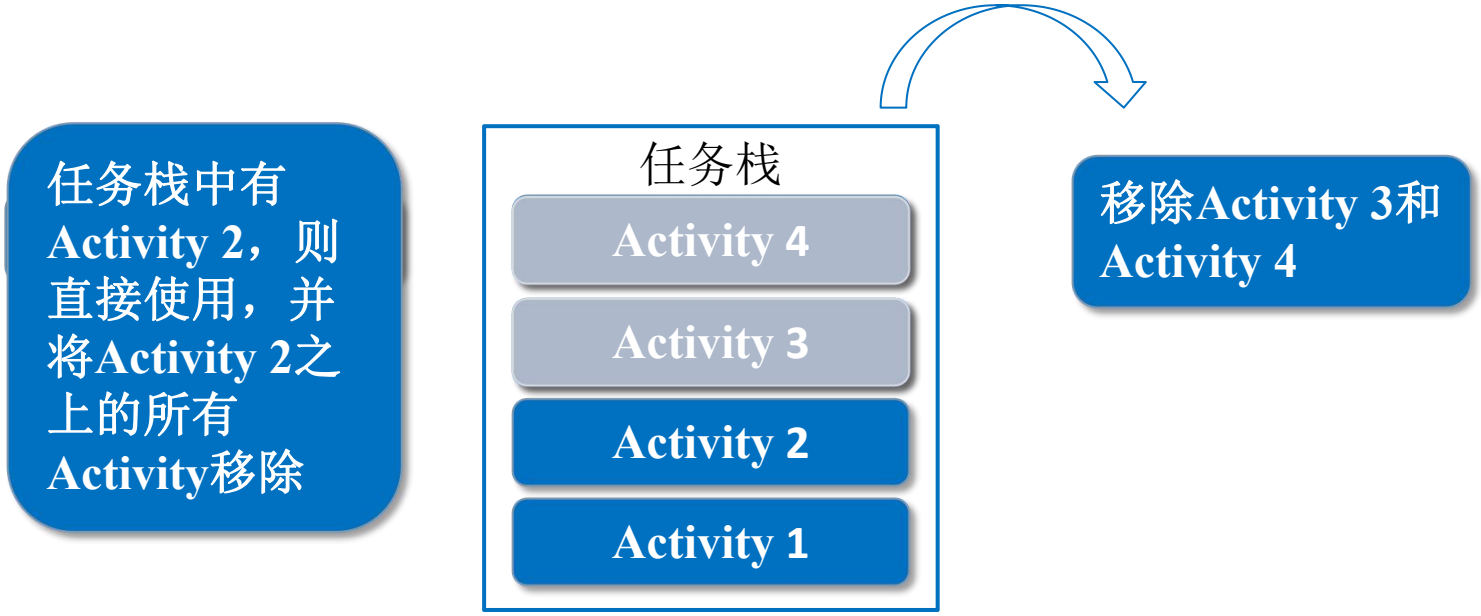
– singleTop模式会判断要启动的Activity实例是否位于栈顶，如果位于栈顶则直接复用，否则创建新的实例。



3.3.2 Activity的四种启动模式

singleTask模式

– singleTask模式下每次启动该Activity时，系统首先会检查栈中是否存在当前Activity实例，如果存在则直接使用，并把当前Activity之上的所有实例全部出栈。



3.3.2 Activity的四种启动模式

singleInstance模式

– singleInstance模式会启动一个新的任务栈来管理Activity实例，无论从哪个任务栈中启动该Activity，该实例在整个系统中只有一个。

主讲内容

3.1 Activity的创建

3.2 Activity的生命周期

3.3 Activity的启动模式

3.4 Activity之间的跳转

主讲内容

Speech content

3.4.1 Intent介绍

Intent(意图)

- Intent被称为意图，是程序中各组件进行交互的一种重要方式，它不仅可以指定当前组件要执行的动作，还可以在不同组件之间进行数据传递。
- 一般用于启动Activity、Service以及发送广播等。根据开启目标组件的方式不同，Intent被分为两种类型显示意图和隐式意图。

3.4.1 Intent介绍

显式意图可以直接通过名称开启指定的目标组件

显式意图

隐式意图

隐式意图通过指定action和category等属性，系统根据这些信息进行分析后寻找目标Activity

3.4.1 Intent介绍

显式意图

创建一个Intent对象，其中第1个参数为Context表示当前的Activity对象，第2个参数表示要启动的目标Activity。

```
Intent intent = new Intent(this, Activity02.class);
startActivity(intent);
```

调用Activity的startActivity方法启动目标组件

3.4.1 Intent介绍

隐式意图

```
Intent intent = new Intent();
intent.setAction("cn.itcast.START_ACTIVITY");
startActivity(intent);
```

设置action动作，当与清单文件中的action相匹配时启动目标组件。

```
<activity android:name="cn.itcast.A
  <intent-filter>
    <action android:name="cn.itca
    <category android:name="andr
  </intent-filter>
</activity>
```

设置action动作，当与代码中的action与该action相匹配时启动该组件。action android:name 配置用于启动此Activity的请求字串, category android:name 只能使用系统定义好的类型，这里类型为默认

3.4.2 实战演练——打开浏览器

1 功能描述： 展示HelloWorld界面。

2 技术要点： 使用AndroidStudio创建程序，
使用模拟器运行程序。

3 实现步骤：

① AndroidStudio中选择【File】→【New】→【New Project】选项创建项目

② 点击工具栏中【AVD Manager】按钮启动模拟器

③ 点击工具栏中的运行按钮运行程序

主讲内容

3.2 Activity的生命周期

3.3 Activity的启动模式

3.4 Activity之间的跳转

3.5 Activity中的数据传递

主讲内容

Speech content

3.5.1 数据传递

数据传递

– Activity之间传递数据需要用到Intent提供的putExtra()方法。

```
String data = "Hello Activity02"
Intent intent = new Intent(this,Activity02.class);
intent.putExtra("extra_data",data);
startActivity(intent);
```

在上述代码中，通过显式意图开启Activity02，并通过putExtra()方法传递了一个字符串data。putExtra()方法中第一个参数接收的是key，第二个参数接收的是value。

3.5.1 数据传递

数据传递

如果想要在activity02中取出传递过来的数据，可以使用如下代码：

```
Intent intent = getIntent();
String data == intent.getStringExtra("extra_data");
log.i("Activity02",data);
```

上述这种数据传递方式是最简单的一种数据传递方式，还有一种传递数据的方式是调用putExtras()方法传递数据，该方法传递的是Bundle对象。调用putExtra()方法传递数据可以使用如下代码：

3.5.1 数据传递

数据传递

```
Bundle bundle = new Bundle();
bundle.putString("name","Linda");
bundle.putInt("age",20);
Intent intent = new Intent(this,Activity02.class);
intent.putExtras(bundle);
startActivity(intent);
```

如果想要在activity02中取出上述方式传递的数据，可以使用如下代码

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
String stuName = bundle.getString("name");
int stuAge = bundle.getString("age");
```

3.5.2 实战演练——注册用户信息

1 功能描述： 将注册界面信息传递到展示界面进行展示。

2 技术要点： 使用Intent传递数据，获取Intent中的数据。

3 实现步骤：

① 注册界面的设计与实现

② 数据展示界面的设计与实现

③ 注册界面逻辑代码的设计与实现

④ 数据展示界面逻辑代码的设计与实现

3.5.3 数据回传


下午4:25
0.00K/s
发送
传智播客-黑马程序员
所在位置
谁可以看
公开
提醒谁看
下午4:26
0.03K/s
所在位置
温都水城
北京市昌平区北七家镇平西府街55号宏福创业园8号
修正大厦
北京市昌平区郑上路西100米
TBD云集中心
北京市昌平区北七家七北路与黄平路路口
中央戏剧学院(宏福校区)
北京市昌平区宏福中路4号
湖湾酒店
北京市昌平区北七家镇平西王府街51号温都水城内
点击“所在位置”跳转新的Activity进行选择
下午4:26
0.00K/s
发送
传智播客-黑马程序员
TBD云集中心
谁可以看
公开
提醒谁看
将所选位置信息回传给上个Activity

3.5.3 数据回传

```
Intent intent = new Intent(this,Activity02.class);
startActivityForResult(intent,1);
```

```
Intent intent = new Intent();
intent.putExtra("extra_data","Hello Activi
setResult(1,intent);
```

```
protected void onActivityResult(int reque
  super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1){
        if (resultCode == 1) {
            String string= data.getStringExtra("extra_data");
        }
    }
}
```

使用startActivityForResult方法开启新的Activity，第1个参数是Intent对象，第2个参数是请求码，用于判断数据的来源输入一个唯一值。

在Activity02 中添加返回数据。

Activity02被销毁之后在Activity01中回调onActivityResult()方法。

3.5.4 实战演练——选择宝宝装备

1 功能描述： 将注册界面信息传递到展示界面进行展示。

2 技术要点： 使用Intent回传数据，Activity之间的跳转。

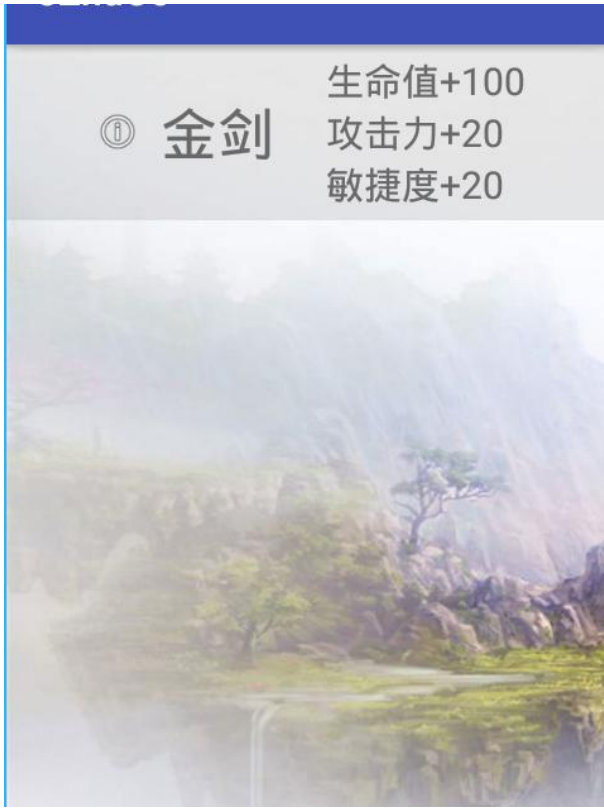
3 实现步骤：

① 宠物显示界面的设计与实现
② 购买装备界面的设计与实现
③ 创建封装装备信息的实体类
④ 购买装备界面逻辑代码的设计与实现
⑤ 宠物显示界面逻辑代码的设计与实现

案例代码（详见教材P21—P26）


用户名 请输入用户名
密　码 请输入密码
注册


用户名：
密　码：
主人，快给小宝宝购买装备吧
生命值：
0
攻击力：
0
敏　捷:
0
立即购买 GO！


金剑
生命值+100
攻击力+20
敏捷度+20

3.6 本章小结

本章主要讲解了Activity的相关知识，包括Activity入门、Activity生命周期、Activity启动模式、Intent的使用以及Activity中的数据传递，并在讲解各个知识点时编写了相应的使用案例。在应用程序中凡是有界面都会使用到Activity，因此，要求初学者必须熟练掌握该组件的使用。

本章作业

- 请简要说明Activity有几种启动模式，以及每种启动模式的特点。
- 请简要写出Activity生命周期中的方法及其作用。

预习作业

- Android平台提供了几种数据存储方式
- XML数据与JSON数据的区别

Thank You!