西南交通大學
Southwest Jiaotong University

移动商务应用开发

第3章 Android界面布局

学习内容

3.1 常见界面布局

- LinearLayout
- TableLayout
- FrameLayout
- ConstraintLayout

3.2 RecyclerView列表视图

3.3 菜单

- OptionsMenu
- PopupMenu

学习目标

了解

- LinearLayout
- TableLayout
- FrameLayout

掌握

- ConstraintLayout
- RecyclerView列表视图
- OptionsMenu
- PopupMenu

学习内容

3.1 常见界面布局

- LinearLayout
- TableLayout
- FrameLayout
- ConstraintLayout

3.2 RecyclerView列表视图

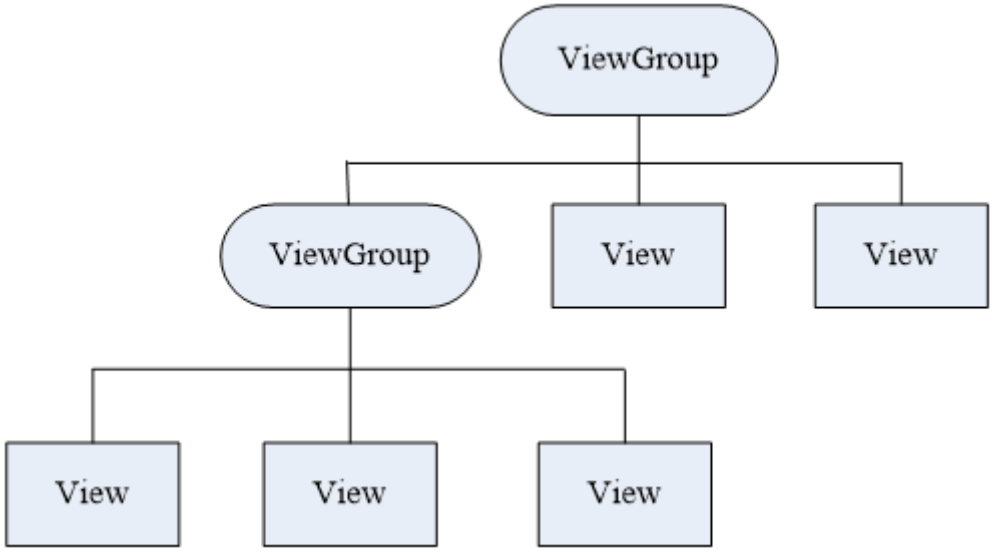
3.3 菜单

- OptionsMenu
- PopupMenu

主要内容

知识回顾

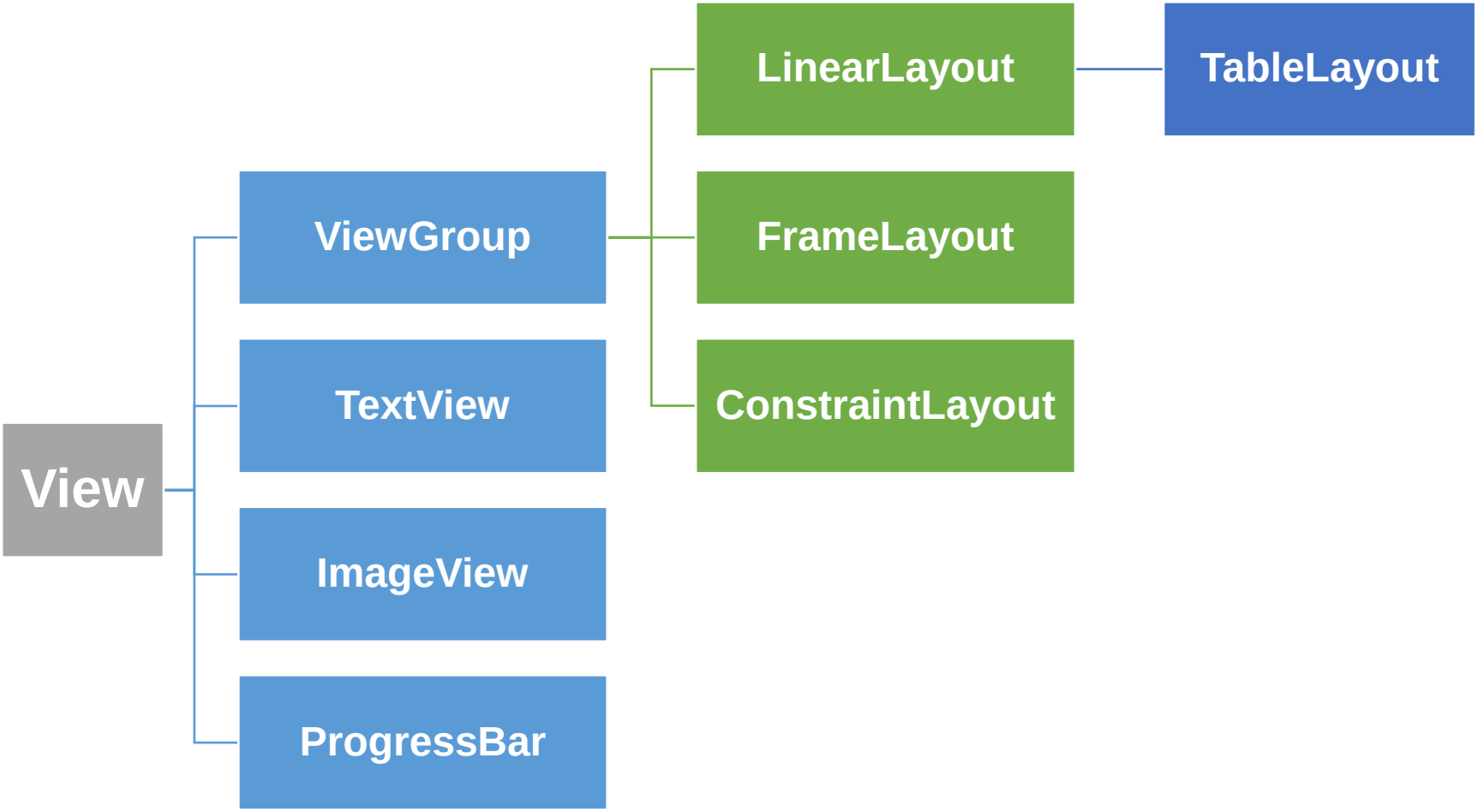
- Android页面中的所有元素均使用View和ViewGroup对象的层次结构进行构建。
  - view对象通常称为" 控件"，用户可查看并进行交互。
  - viewGroup对象通常称为“容器”或“布局”，用户不可见



常见界面布局

常见界面布局	特点
LinearLayout	以水平或竖直方向排列
TableLayout	以表格方式排列
FrameLayout	以帧的方式叠加
ConstraintLayout	控件位置灵活，主要以可视化的方式编写布局

常见界面布局


View
ViewGroup
TextView
ImageView
ProgressBar
LinearLayout
FrameLayout
ConstraintLayout
TableLayout

线性布局 LinearLayout

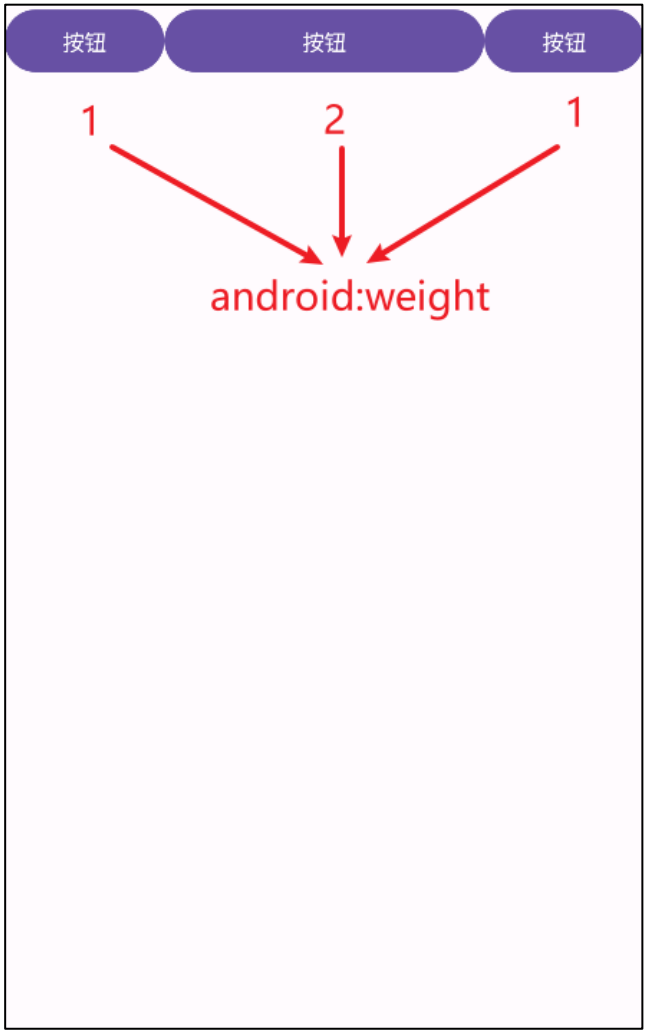
- **线性布局**以单个方向（水平或竖直）来显示界面中的控件。
- 可以通过布局的android:orientation属性来控制布局内控件的排列方向。
  - 当排列方向为水平时，控件从左到右依次排列；
  - 当排列方向为竖直时，控件从上到下依次排列。

线性布局 LinearLayout

线性布局 LinearLayout

- 可以通过控件的android:weight属性来为控件分配权重，进而调节控件所占空间的大小。
  - 如果控件使用了android:weight属性，当布局为水平方向时，控件的宽度应设为0dp，当布局为竖直方向时，控件的高度应设为0dp。
  - 如果同一个布局中存在多个控件，那么android:weight的值越大的控件占据的空间越大，按比例分配控件。
  - 如果同一个布局中存在多个控件，有的控件没使用android:weight属性，有的控件使用了android:weight属性，那么先按照前者指定的大小显示前者，然后根据后者中各个控件的weight大小将剩余空间按比例分配给后者中各个控件。

线性布局 LinearLayout



线性布局 LinearLayout

➢LinearLayout的常见属性：

属性	功能描述
android:orientation	布局中空间的排列方向

- orientation选项：`horizontal` `vertical`

➢LinearLayout中控件的常见属性：

属性	功能描述
android:layout_weight	控件权重
android:gravity	控件内文字的显示位置
android:layout_gravity	控件在父布局中的显示位置

- gravity常见选项：`bottom` `center` `start` `end` 等，可多选。
- layout_gravity常见选项：`bottom` `center` `start` `end` 等，可多选。

表格布局 TableLayout

- **表格布局**采用行、列的形式来管理控件
  - 它不需要明确声明包含多少行、多少列
  - 通过在TableLayout布局中添加TableRow布局来控制表格的行数，
  - 通过在TableRow布局中添加控件来控制这行的列数。
- TableLayout继承于LinearLayout，TableRow也继承于LinearLayout。

表格布局 TableLayout

- TableLayout的常见属性：

属性	功能描述
android:stretchColumns	设置该列被拉伸（使得填充界面宽度）
android:shrinkColumns	设置该列被收缩（使得不超出界面宽度）
android:collapseColumns	设置该列被隐藏

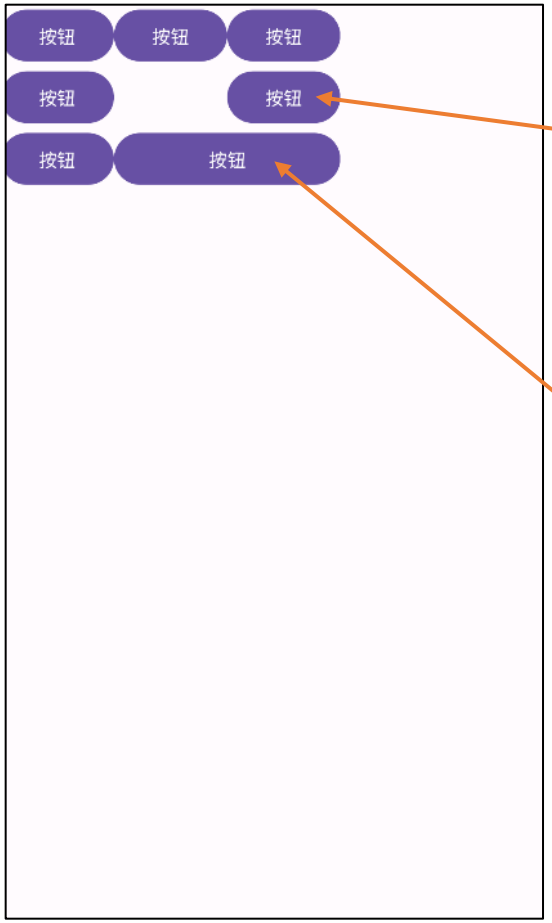
- TableLayout中列的index从0开始。
- android:xxxColumns可设置多列被拉伸，例如：`android:stretchColumns="0,1"`

表格布局 TableLayout

- TableLayout中控件的常见属性：

属性	功能描述
android:layout_column	设置控件显示在哪一列
android:layout_span	设置控件占据几列
android:layout_weight	控件权重

表格布局 TableLayout




```
<Button
    android:id="@+id/button20"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:text="按钮" />
```

```
<Button
    android:id="@+id/button22"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_span="2"
    android:text="按钮" />
```

帧布局 FrameLayout

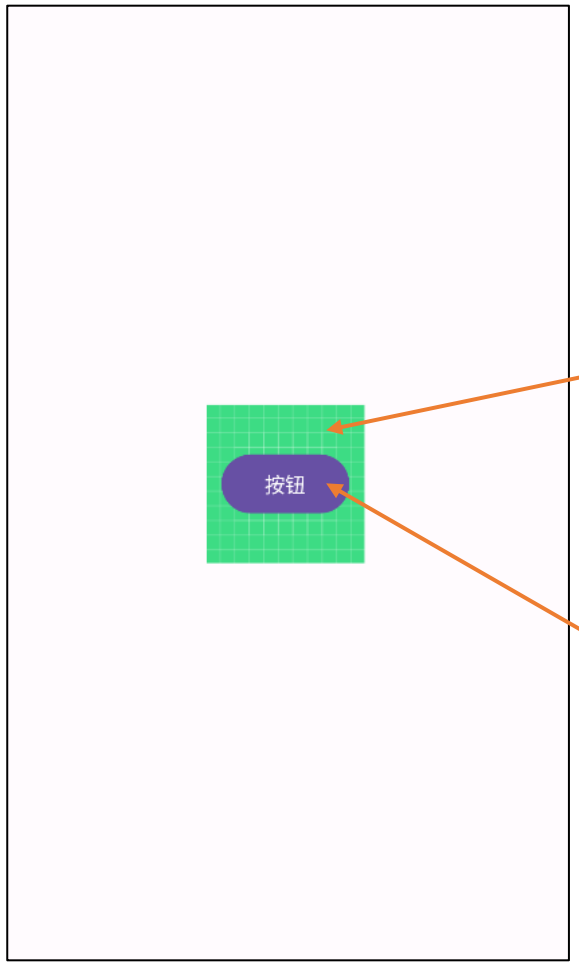
- **帧布局**为每个添加的控件创建一帧，这些帧会一个个叠加在一起，后加入的控件会叠加在前一个控件上层。
- FrameLayout中的常见属性：

属性	功能描述
android:foreground	设置前景图片
android:foregroundGravity	设置前景图片的显示位置

- FrameLayout中控件的常见属性：

属性	功能描述
android:layout_gravity	设置控件在父布局中的显示位置

帧布局 FrameLayout




```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:contentDescription="背景图片"
    app:srcCompat="@drawable/ic_launcher_background" />
```

```
<Button
    android:id="@+id/button8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="按钮" />
```

约束布局 ConstraintLayout

- **约束布局**根据控件之间或控件与父布局之间的关系进行布局。
- 它适合使用可视化的方式编写界面布局——当然，可视化操作的背后仍然是使用XML代码实现的，只不过这些代码是Android Studio根据我们的操作自动生成的。

约束布局 ConstraintLayout

- 相对定位

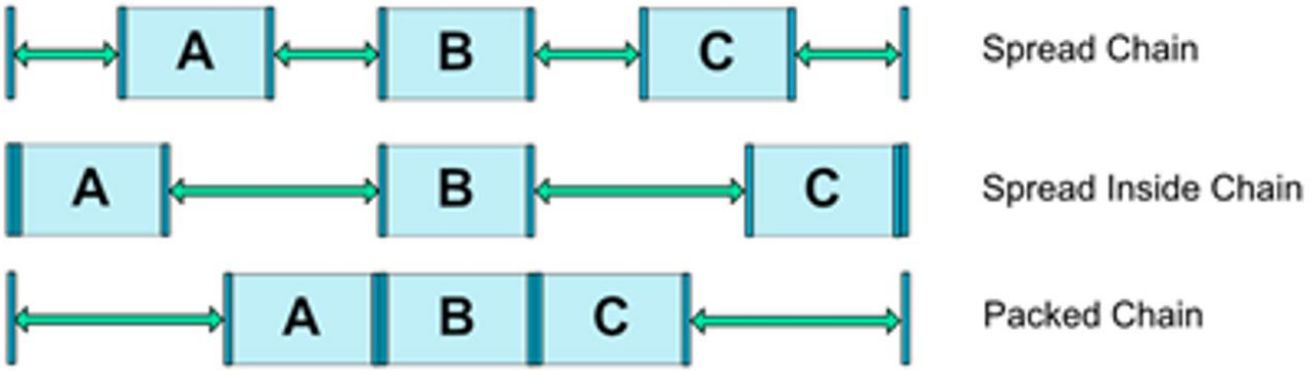
```
app:layout_constraintBottom_toTopOf="@+id/guideline"
app:layout_constraintEnd_toStartOf="@+id/button2"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

- 倾向

```
app:layout_constraintHorizontal_bias="0.5" // 横向倾向
app:layout_constraintVertical_bias="0.8" // 纵向倾向
```

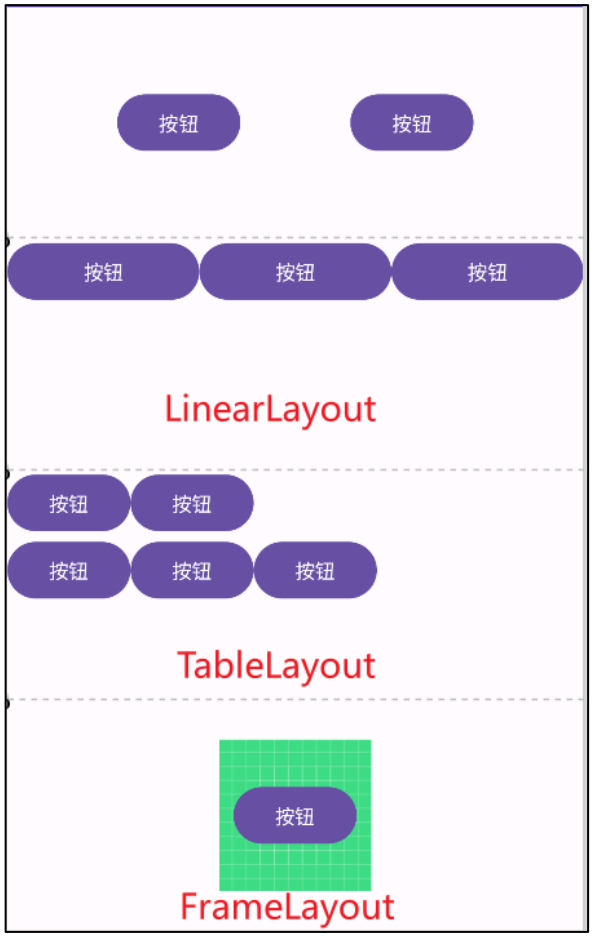
约束布局 ConstraintLayout

- 链（chain）



约束布局 ConstraintLayout

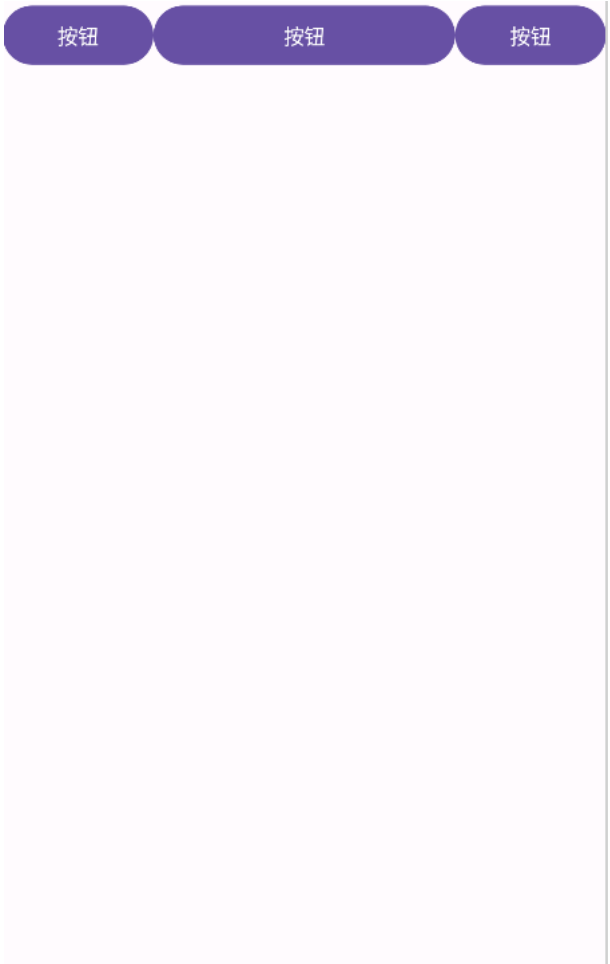
- 布局嵌套



练习：使用四种界面布局

技能要点：

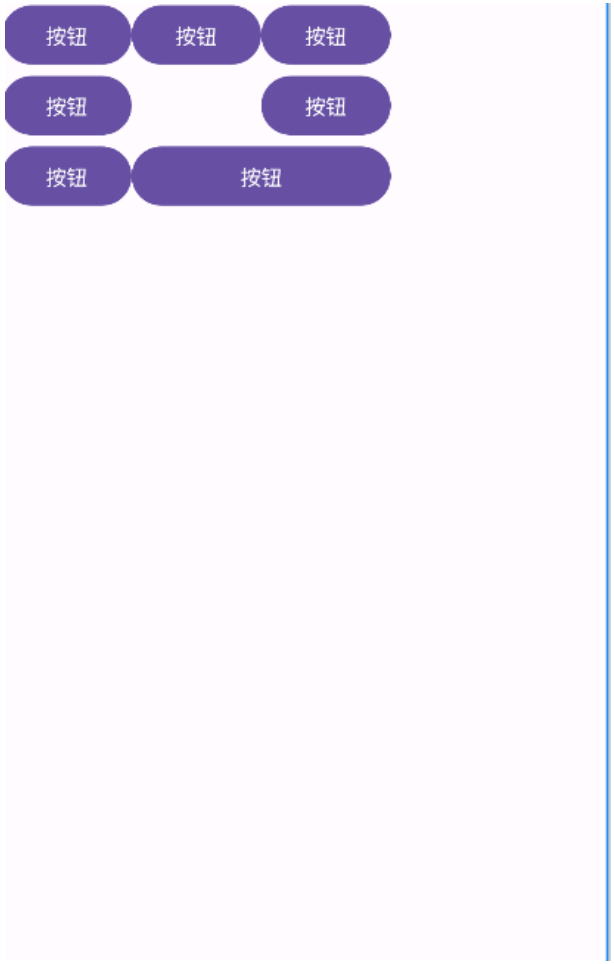
- 使用LinearLayout
- 使用LinearLayout的属性
  - orientation
- 使用LinearLayout中控件的属性
  - gravity
  - layout_gravity
  - weight



练习：使用四种界面布局

技能要点：

- 使用TableLayout
- 使用TableLayout的属性
  - stretchColumns
  - shrinkColumns
  - collapseColumns
- 使用TableLayout中控件的属性
  - layout_span
  - layout_column
  - layout_weight



练习：使用四种界面布局

技能要点：

- 使用FrameLayout
- 使用FrameLayout中控件的属性
  - layout_gravity

练习：使用四种界面布局

技能要点：

- 使用ConstraintLayout
  - 相对定位
  - 使用链
  - 使用辅助线
  - 使用布局嵌套

学习内容

3.1 常见界面布局

- LinearLayout
- TableLayout
- FrameLayout
- ConstraintLayout

3.2 RecyclerView列表视图

3.3 菜单

- OptionsMenu
- PopupMenu

主要内容

RecyclerView列表视图

- RecyclerView是ViewGroup的子类。
- 可以通过**RecyclerView**创建滚动列表视图
  - 因为列表中的每个子项是可回收的（recyclable），所以RecyclerView创建的列表视图适用于为大型数据集显示元素。

RecyclerView列表视图

- 效果展示

RecyclerView列表视图

- 使用RecyclerView涉及到以下要素：
  - **包含RecyclerView的布局文件**。该布局文件用于设置整个滚动列表的布局。
  - **显示子项视图的布局文件**。该布局文件用于设置滚动列表中每个子项的布局。
  - **适配器（Adapter）**。Adapter是数据与RecyclerView之间的桥梁，用来管理RecyclerView。
  - **视图持有者（ViewHolder）**。ViewHolder用来管理RecyclerView中的每个子项的视图。

RecyclerView列表视图

- 使用RecyclerView的步骤包括：
  1. 创建包含RecyclerView的布局文件，如：activity_main.xml。
  2. 创建用于显示RecyclerView中每个子项视图的布局文件，如：item_view_linear.xml。
  3. 创建自定义ViewHolder，继承于RecyclerView.ViewHolder，获取子项视图中每个控件。
  4. 创建自定义Adapter，继承于RecyclerView.Adapter<MyRecyclerViewAdapter.MyViewHolder>，重写构造函数、onCreateViewHolder、onBindViewHolder、getItemCount。
  5. 在Activity中获取recyclerView控件，并为recyclerView设置LayoutManager和Adapter。

练习：创建RecyclerView列表视图

功能描述：

- 使用RecyclerView创建水果滚动列表

技术要点：

- 定义水果类
- 对象数据的传递
- 创建RecyclerView
  - 创建布局文件
  - 创建自定义ViewHolder
  - 创建自定义Adapter
- 切换视图
- 设置子项的点击效果
- 设置ActionBar上显示的文字

练习：创建RecyclerView列表视图

学习内容

主要内容

3.1 常见界面布局

- LinearLayout
- TableLayout
- FrameLayout
- ConstraintLayout

3.2 RecyclerView列表视图

3.3 菜单

- OptionsMenu
- PopupMenu

选项菜单 OptionsMenu

- 选项菜单（OptionMenu）显示在操作栏上。
- 创建选项菜单的步骤：
  1. 创建设置选项菜单视图的布局文件，如：options_menu.xml。
  2. 重写Activity的onCreateOptionsMenu(Menu menu)方法创建选项菜单。
  3. 重写Activity的onOptionsItemSelected(MenuItem mi)方法对选项菜单的控件实现事件监听。

选项菜单 OptionsMenu

- 效果展示

选项菜单 OptionsMenu

- 代码展示

```
1  @Override
2  public boolean onCreateOptionsMenu(Menu menu) {
3      getMenuInflater().inflate(R.menu.options_menu, menu);
4      return super.onCreateOptionsMenu(menu);
5  }
```

```
1  @Override
2  public boolean onOptionsItemSelected(@NonNull MenuItem item) {
3      // 代码块：事件监听
4      return super.onOptionsItemSelected(item);
5  }
```

弹出式菜单 PopupMenu

- 弹出式菜单（PopupMenu）会在指定组件的上方或下方弹出。
- 弹出式菜单的创建方式和对话框的创建方式类似。
- 创建选项菜单的步骤：
  1. 创建设置弹出式菜单视图的布局文件，如：popup_menu.xml。
  2. 调用new PopupMenu(Context context, View anchor)创建下拉菜单，anchor代表要激发弹出菜单的组件。
  3. 调用MenuInflater的inflate()方法设置PopupMenu的界面布局。
  4. 对PopupMenu中的控件实现事件监听。
  5. 调用PopupMenu的show()方法显示弹出式菜单。

弹出式菜单 PopupMenu

- 效果展示

弹出式菜单 PopupMenu

➢代码展示

```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 创建popupMenu
        PopupMenu popupMenu = new PopupMenu(context, holder.itemView);
        popupMenu.inflate(R.menu.popup_menu);
        popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                // 事件监听：处理popupMenu中点击事件
                // 代码块
                return false;
            }
        });
        popupMenu.setGravity(Gravity.END);
        popupMenu.show();
    }
});
```

针对每个子项

设置布局

事件监听

设置显示位置

练习：创建RecyclerView列表视图

功能描述：

- 为RecyclerView添加optionsMenu
- 为RecyclerView中每个子项添加PopupMenu
- 实现事件监听（以Toast为例）
- 实现切换视图的功能

课堂总结

3.1 常见界面布局

- LinearLayout
- TableLayout
- FrameLayout
- ConstraintLayout

3.2 RecyclerView列表视图

3.3 菜单

- OptionsMenu
- PopupMenu

作业

本章作业

- 创建RecyclerView，添加OptionsMenu和PopupMenu，并实现事件监听
  - “视频：运行后全过程
    - **“水果图鉴”改成姓名 + 学号**

作业提交方式

- 视频，文件名：学号+姓名+第3课作业
- 邮件给助教，主题：学号+姓名+第3课作业