

```

        mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
        setTextSize(mRatio + 13);
    }
}

return true;
}

int getDistance(MotionEvent event) {
    int dx = (int) (event.getX(0) - event.getX(1));
    int dy = (int) (event.getY(0) - event.getY(1));
    return (int) (Math.sqrt(dx * dx + dy * dy));
}

public boolean onTouch(View v, MotionEvent event) {
    return false;
}
}

```

4.6. TextView с различными размерами текста

Вы можете архивировать различные размеры текста внутри текстового представления с помощью Span:

```

TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)

```

4.7. Настройка тем и стилей

MainActivity.java:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

activity_main.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.customthemattributedemo.customview.CustomTextView
        style="?mediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

```

```

        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />

    <com.customthemattributedemo.customview.CustomTextView
        style="?largeTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />
</LinearLayout>

```

CustomTextView.java:

```

public class CustomTextView extends TextView {

    private static final String TAG = "TextViewPlus";
    private Context mContext;

    public CustomTextView(Context context) {
        super(context);
        mContext = context;
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        setCustomFont(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context; setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray customFontNameTypedArray = ctx.obtainStyledAttributes(attrs,
            R.styleable.CustomTextView);
        String customFont = customFontNameTypedArray.getString(R.styleable.
            CustomTextView_font_family);
        Typeface typeface = null;
        typeface = Typeface.createFromAsset(ctx.getAssets(), customFont);
        setTypeface(typeface);
        customFontNameTypedArray.recycle();
    }
}

```

attrs.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

    <attr name="mediumTextStyle" format="reference" />
    <attr name="largeTextStyle" format="reference" />

    <declare-styleable name="CustomTextView">

        <attr name="font_family" format="string" />

```

```
<!-- Ваши другие атрибуты -->

</declare-styleable>
</resources>
```

strings.xml:

```
<resources>
    <string name="app_name">Демонстрация атрибутов темы в пользовательском стиле</string>
    <string name="message_hello">Здравствуйте! </string>
    <string name="bold_font">bold.ttf</string>
</resources>
```

styles.xml:

```
<resources>
    <!-- Базовая тема приложения. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Настроить тему можно здесь. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="mediumTextStyle">@style/textMedium</item>
        <item name="largeTextStyle">@style/textLarge</item>
    </style>

    <style name="textMedium" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Medium</item>
    </style>

    <style name="textLarge" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
    </style>

    <style name="textParentStyle">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@color/colorPrimary</item>
        <item name="android:padding">5dp</item>
    </style>
</resources>
```

4.8. Настройка TextView

```
public class CustomTextView extends TextView {
    private float strokeWidth;
    private Integer strokeColor;
    private Paint.Join strokeJoin;
    private float strokeMiter;

    public CustomTextView(Context context) {
        super(context);
```

```

    init(null);
}

public CustomTextView(Context context, AttributeSet attrs) {
    super(context, attrs);
    init(attrs);
}

public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    init(attrs);
}

public void init(AttributeSet attrs) {

    if (attrs != null) {
        TypedArray a = getContext().obtainStyledAttributes(attrs, R.styleable.
CustomTextView);

        if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
            float strokeWidth = a.getDimensionPixelSize(R.styleable.
CustomTextView_strokeWidth, 1);
            int strokeColor = a.getColor(R.styleable.CustomTextView_.
strokeColor, 0xff000000);
            float strokeMiter = a.getDimensionPixelSize(R.styleable.
CustomTextView_strokeMiter, 10);
            Paint.Join strokeJoin = null;
            switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                case (0):
                    strokeJoin = Paint.Join.MITER;
                    break;
                case (1):
                    strokeJoin = Paint.Join.BEVEL;
                    break;
                case (2):
                    strokeJoin = Paint.Join.ROUND;
                    break;
            }
            this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
        }
    }
}

public void setStroke(float width, int color, Paint.Join join, float miter) {
    strokeWidth = width;
    strokeColor = color;
    strokeJoin = join;
    strokeMiter = miter;
}

@Override
public void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    int restoreColor = this.getCurrentTextColor();
    if (strokeColor != null) {
        TextPaint paint = this.getPaint();
        paint.setStyle(Paint.Style.STROKE);

```

```
    paint.setStrokeJoin(strokeJoin);
    paint.setStrokeMiter(strokeMiter);
    this.setTextColor(strokeColor);
    paint.setStrokeWidth(strokeWidth);
    super.onDraw(canvas);
    paint.setStyle(Paint.Style.FILL);
    this.setTextColor	restoreColor);
}
}
```

Использование:

```
public class MainActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        CustomTextView customTextView = (CustomTextView) findViewById(R.id.pager_title);  
    }  
}
```

Maket:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@mipmap/background">

    <pk.sohail.gallerytest.activity.CustomTextView
        android:id="@+id/pager_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:gravity="center"
        android:text="@string/txt_title_photo_gallery"
        android:textColor="@color/white"
        android:textSize="30dp"
        android:textStyle="bold"
        app:outerShadowRadius="10dp"
        app:strokeColor="@color/title_text_color"
        app:strokeJoinStyle="miter"
        app:strokeWidth="2dp" />

</RelativeLayout>
```

```
</RelativeLayout>
```

Атрибуты:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CustomTextView">
```

```

<attr name="outerShadowRadius" format="dimension" />
<attr name="strokeWidth" format="dimension" />
<attr name="strokeMiter" format="dimension" />
<attr name="strokeColor" format="color" />
<attr name="strokeJoinStyle">
    <enum name="miter" value="0" />
    <enum name="bevel" value="1" />
    <enum name="round" value="2" />
</attr>
</declare-styleable>

```

</resources>

Программное использование:

```

CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//тогда используйте
setStroke(float width, int color, Paint.Join join, float miter);
//метод перед установкой
setText("Sample Text");

```

4.9. Одиночный TextView с двумя разными цветами

Цветной текст можно создать, передав текст и имя цвета шрифта в следующую функцию:

```

private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}

```

Затем с помощью приведенного ниже кода цветной текст может быть установлен в TextView (или даже в Button, EditText и т. д.).

Сначала определите TextView следующим образом:

```
TextView txtView = (TextView) findViewById(R.id.txtView);
```

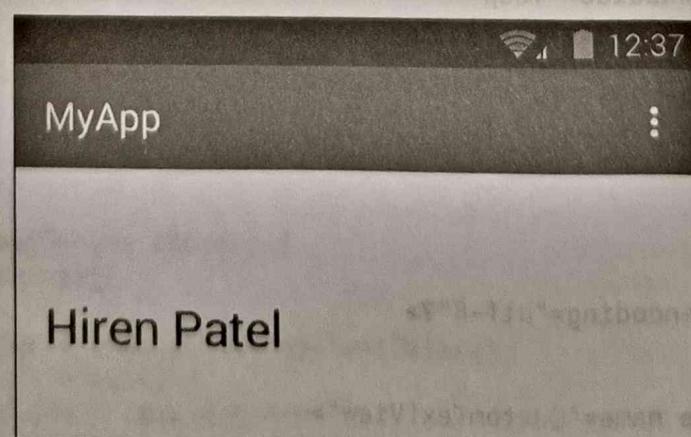
Затем создайте текст разного цвета и присвойте его строкам:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel", "#000080");
```

Наконец, установите две разноцветные строки в TextView:

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Скриншот:



Глава 5. AutoCompleteTextView

5.1. Автозаполнение с помощью CustomAdapter, ClickListener и фильтра

Основной макет: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <AutoCompleteTextView
        android:id="@+id/auto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="2"
        android:hint="@string/hint_enter_name" />
</LinearLayout>
```

Схема расположения строк row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">Введите имя</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mList = retrievePeople();
txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
txtSearch.setAdapter(adapter);
txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int pos,
    long id) {
        //это способ поиска выбранного объекта/элемента
        selectedPerson = (People) adapterView.getItemAtPosition(pos);
    }
});
}

private List<People> retrievePeople() {
    List<People> list = new ArrayList<People>();
    list.add(new People("James", "Bond", 1));
    list.add(new People("Jason", "Bourne", 2));
    list.add(new People("Ethan", "Hunt", 3));
    list.add(new People("Sherlock", "Holmes", 4));
    list.add(new People("David", "Beckham", 5));
    list.add(new People("Bryan", "Adams", 6));
    list.add(new People("Arjen", "Robben", 7));
    list.add(new People("Van", "Persie", 8));
    list.add(new People("Zinedine", "Zidane", 9));
    list.add(new People("Luis", "Figo", 10));
    list.add(new People("John", "Watson", 11));
    return list;
}
}

```

Класс модели: People.java

```

public class People {
    private String name, lastName;
    private int id;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

```

    }

    public String getLastname() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}

```

Класс адаптера: PeopleAdapter.java

```

public class PeopleAdapter extends ArrayAdapter<People> {

    Context context;
    int resource, textViewResourceId;
    List<People> items, tempItems, suggestions;

    public PeopleAdapter(Context context, int resource, int textViewResourceId,
    List<People> items)
    {
        super(context, resource, textViewResourceId, items);
        this.context = context;
        this.resource = resource; this.textViewResourceId = textViewResourceId; this.
        items = items;
        tempItems = new ArrayList<People>(items); // в этом разница
        suggestions = new ArrayList<People>();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    { View view = convertView;
    if (convertView == null) {
        LayoutInflator inflater = (LayoutInflator)
    context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        view = inflater.inflate(R.layout.row, parent, false);
    }
    People people = items.get(position);
    if (people != null) {
        TextView lblName = (TextView) view.findViewById(R.id.lbl_name);
        if (lblName != null)
            lblName.setText(people.getName());
    }
    return view;
    }

    @Override
    public Filter getFilter() {
        return nameFilter;
    }

    /**
     * Реализация пользовательских фильтров для предоставляемых нами предложений.
     */
    Filter nameFilter = new Filter() {
        @Override
        public CharSequence convertResultToString(Object resultValue) {
            String str = ((People) resultValue).getName();
        }
    }
}

```

```

        return str;
    }

    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        if (constraint != null) { suggestions.clear();
            for (People people : tempItems) {
                if (people.getName().toLowerCase().contains(constraint.toString().toLowerCase())) {
                    suggestions.add(people);
                }
            }
            FilterResults filterResults = new FilterResults();
            filterResults.values = suggestions;
            filterResults.count = suggestions.size();
            return filterResults;
        } else {
            return new FilterResults();
        }
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        List<People> filterList = (ArrayList<People>) results.values;
        if (results != null && results.count > 0) {
            clear();
            for (People people : filterList) {
                add(people);
                notifyDataSetChanged();
            }
        }
    }
}

```

5.2. Простой, жестко закодированный AutoCompleteTextView

Дизайн (верстка XML):

```

<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="65dp"
    android:ems="10" />

```

Найдите представление в коде после `setContentView()` (или его фрагмент, или эквивалент пользовательского представления):

```

final AutoCompleteTextView my.AutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);

```

Представление жестко закодированных данных через адаптер:

```

String[] countries = getResources().getStringArray(R.array.list_of_countries);

```

6.1. Granularity

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries);
myAutoCompleteTextView.setAdapter(adapter);
```

Совет: предпочтительнее было бы предоставлять данные через какой-нибудь загрузчик, а не через жестко закодированный список, как в данном случае.

Глава 6. Autosizing TextView

Это вид `TextView`, который автоматически изменяет размер текста, чтобы он идеально вписывался в его границы.

Существует возможность указать `TextView`, чтобы размер текста автоматически увеличивался или уменьшался для заполнения макета, основываясь на характеристиках и границах `TextView`.

Настроить авторазмер `TextView` можно как в коде, так и в XML.

Существует два способа установки авторазмера `TextView`: **Granularity** и **Preset Sizes**.

6.1. Granularity

На языке Java:

Вызовите метод `setAutoSizeTextTypeUniformWithConfiguration()`:

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize,
int autoSizeMaxTextSize, int autoSizeStepGranularity, int unit)
```

В XML:

С помощью атрибутов `autoSizeMinTextSize`, `autoSizeMaxTextSize` и `autoSizeStepGranularity` можно задать размеры авторазметки в XML-файле макета:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:layout_marginLeft="0dp"
    android:layout_marginTop="0dp"
    android:autoSizeMaxTextSize="100sp"
    android:autoSizeMinTextSize="12sp"
    android:autoSizeStepGranularity="2sp"
    android:autoSizeText="uniform"
    android:text="Hello World!"
    android:textSize="100sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Более подробную информацию можно найти в `AutosizingTextViews-Demo` на GitHub по адресу <https://github.com/1priyank1/AutosizingTextViews-Demo>.

6.2. Preset Sizes

На языке Java:

Вызовите метод `setAutoSizeTextTypeUniformWithPresetSizes()`:

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

В XML:

Используйте атрибут `autoSizePresetSizes` в XML-файле макета:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
    android:layout_width="wrap_content"
    android:layout_height="250dp"
    android:layout_marginLeft="0dp"
    android:layout_marginTop="0dp"
    android:autoSizeText="uniform"
    android:autoSizePresetSizes="@array/autosize_text_sizes"
    android:text="Hello World!"
    android:textSize="100sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Чтобы получить доступ к массиву как к ресурсу, определите массив в файле `res/values/arrays.xml`:

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

Более подробную информацию можно найти в `AutosizingTextViews-Demo` на GitHub по адресу <https://github.com/1priyank1/AutosizingTextViews-Demo>.

Глава 7. ListView

`ListView` – это группа просмотра (`viewgroup`), которая группирует несколько элементов из источника данных, например массива или базы данных, и отображает их в виде списка с возможностью прокрутки. Данные связываются с `ListView` с помощью класса `Adapter`.

7.1. Пользовательский `ArrayAdapter`

По умолчанию класс `ArrayAdapter` создает представление для каждого элемента массива, вызывая для каждого элемента функцию `toString()` и помещая его содержимое в `TextView`.

Чтобы создать сложное представление для каждого элемента (например, если для каждого элемента массива требуется `ImageView`), расширьте класс `ArrayAdapter` и переопределите метод `getView()`, чтобы вернуть желаемый тип представления для каждого элемента.

Например:

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflator inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflator.from(context);
    }
```

7.2. Базовый ListView с адаптером ArrayAdapter

```

@Override
public long getItemId(int position)
{
    //Это просто пример
    YourClassData data = (YourClassData) getItem(position);
    return data.ID;
}

@Override
public View getView(int position, View view, ViewGroup parent)
{
    ViewHolder viewHolder;
    if (view == null) {
        view = inflater.inflate(R.layout.custom_row_layout_design, null);
        // Выполнить инициализацию

        //Получить представление на макете элемента и установить значение.
        viewHolder = new ViewHolder(view);
        view.setTag(viewHolder);
    }
    else {
        viewHolder = (ViewHolder) view.getTag();
    }

    //Получить свой объект
    YourClassData data = (YourClassData) getItem(position);

    viewHolder.txt.setTypeface(m_Font);
    viewHolder.txt.setText(data.text);
    viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

    return view;
}
private class ViewHolder
{
    private final TextView txt;
    private final ImageView img;

    private ViewHolder(View view)
    {
        txt = (TextView) view.findViewById(R.id.txt);
        img = (ImageView) view.findViewById(R.id.img);
    }
}
}

```

7.2. Базовый ListView с адаптером ArrayAdapter

По умолчанию ArrayAdapter создает представление для каждого элемента массива, вызывая для каждого элемента функцию `toString()` и помещая его содержимое в `TextView`.

Пример:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, myStringArray);
```

где `android.R.layout.simple_list_item_1` – макет, содержащий `TextView` для каждой строки в массиве. Затем просто вызовите `setAdapter()` для `ListView`:

```
ListView listView = (ListView) findViewById(R.id.listview); listView.setAdapter(adapter);
```

Чтобы использовать для отображения массива не TextView, а что-то другое, например ImageView, или чтобы представления заполнялись какими-либо данными помимо результатов `toString()`, переопределите `getView(int, View, ViewGroup)`, чтобы вернуть нужный тип представления.

7.3. Фильтрация с помощью CursorAdapter

```
// Получите ссылку на ваш ListView
ListView listResults = (ListView) findViewById(R.id.listResults);

// Установите свой адаптер
listResults.setAdapter(adapter);

// Включите фильтрацию в ListView
listResults.setTextFilterEnabled(true);

// Подготовьте адаптер к фильтрации
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {
        // в реальной жизни делайте что-то более безопасное, чем конкатенация
        // но это будет зависеть от вашей схемы
        // Это запрос, который будет выполняться при фильтрации
        String query = "SELECT _ID as _id, name FROM MYTABLE "
            + "where name like '%" + constraint + "%' "
            + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
    }
});
```

Допустим, ваш запрос будет выполняться каждый раз, когда пользователь набирает текст в EditText:

```
EditText queryText = (EditText) findViewById(R.id.textQuery);
queryText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, final int start, final
        int count, final int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, final int start, final int
        before, final int count) {
        // Это фильтр в действии
        adapter.getFilter().filter(s.toString());
        // Не забудьте уведомить адаптер
        adapter.notifyDataSetChanged();
    }

    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

Глава 8. Макеты

Макет определяет визуальную структуру пользовательского интерфейса, например активности или виджета.

Макет объявляется в XML, включая элементы экрана, которые будут в нем отображаться. В приложение может быть добавлен код для изменения состояния экранных объектов во время выполнения, в том числе и тех, которые объявлены в XML.

8.1. LayoutParams

Каждая отдельная ViewGroup (например, LinearLayout, RelativeLayout, CoordinatorLayout) должна хранить информацию о свойствах своих дочерних элементов, о том, как располагаются дочерние элементы во ViewGroup. Эта информация хранится в объектах класса-обертки ViewGroup.LayoutParams.

Для включения параметров, характерных для конкретного типа компоновки, в группах ViewGroup используются подклассы класса ViewGroup.LayoutParams.

Например, для

- LinearLayout это LinearLayout.LayoutParams
- RelativeLayout – RelativeLayout.LayoutParams
- CoordinatorLayout – CoordinatorLayout.LayoutParams
- ...

Большинство ViewGroup используют возможность установки полей для своих дочерних элементов, поэтому они не подклассифицируются непосредственно в ViewGroup.LayoutParams, а подклассифицируются в ViewGroup.MarginLayoutParams (который сам является подклассом ViewGroup.LayoutParams).

LayoutParams в xml

Объекты LayoutParams создаются на основе «раздутого» (inflated) xml-файла макета.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_gravity="right"
        android:gravity="bottom"
        android:text="Пример текста"
        android:textColor="@android:color/holo_green_dark" />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_green_dark"
        android:scaleType="centerInside"
        android:src="@drawable/example" />

</LinearLayout>
```

Все параметры, начинающиеся с layout_, определяют, как должен работать **вложенный** в него макет. При «раздувании» макета эти параметры обрамляются в соответствующий

объект `LayoutParams`, который впоследствии будет использоваться макетом для правильного позиционирования конкретного представления в группе `ViewGroup`. Другие атрибуты представления имеют непосредственное отношение к представлению и обрабатываются самим представлением.

Для `TextView`:

- `layout_width`, `layout_height` и `layout_gravity` будут храниться в объекте `LinearLayout.LayoutParams` и использоваться `LinearLayout`
- `gravity`, `text` и `textColor` будут использоваться самим `TextView`

Для `ImageView`:

- `layout_width`, `layout_height` и `layout_weight` будут храниться в объекте `LinearLayout.LayoutParams` и использоваться `LinearLayout`
- `background`, `scaleType` и `src` будут использоваться самим `ImageView`

Получение объекта `LayoutParams`

`getLayoutParams` – метод представления, позволяющий получить текущий объект `LayoutParams`.

Поскольку объект `LayoutParams` напрямую связан с **вложенной** `ViewGroup`, данный метод будет возвращать ненулевое значение только в том случае, если `View` присоединен к `ViewGroup`. Следует иметь в виду, что этот объект может присутствовать не всегда. Особен-но не стоит полагаться на его наличие в конструкторе `View`.

```
public class ExampleView extends View {
    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }
    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }
    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }
    private void setupView(Context context) {
        if (getLayoutParams().height == 50){ // НЕ ДЕЛАЙТЕ ЭТОГО!
            // При этом может возникнуть исключение NullPointerException
            doSomething();
        }
    }
    //...
}
```

Если вы хотите зависеть от наличия объекта `LayoutParams`, то вместо него следует использовать метод `onAttachedToWindow`.

```
public class ExampleView extends View {
    public ExampleView(Context context) {
```

```

    super(context);
}
public ExampleView(Context context, AttributeSet attrs) {
    super(context, attrs);
}

public ExampleView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}

@Override
protected void onAttachedToWindow() {
    super.onAttachedToWindow();
    if (getLayoutParams().height == 50) { // getLayoutParams() здесь НЕ возвращает
        null
        doSomething();
    }
}
//...
}

```

Приведение объекта LayoutParams

Возможно, вам потребуется использовать возможности, характерные для конкретной ViewGroup (например, программно изменять правила RelativeLayout). Для этого необходимо знать, как правильно приводить объект ViewGroup.LayoutParams.

Это может немного запутать при получении объекта LayoutParams для дочернего View, который на самом деле является другой ViewGroup.

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/outer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/inner_layout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="right"/>

</LinearLayout>

```

Важно: тип объекта LayoutParams напрямую связан с типом вложенной ViewGroup.

Неправильное приведение:

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.
getLayoutParams();
// НЕПРАВИЛЬНО! Это приведет к возникновению исключения ClassCastException

```

Правильное приведение:

```

FrameLayout innerLayout = (FrameLayout) findViewById(R.id.inner_layout);

```

```
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.  
getLayoutParams();  
// ПРАВИЛЬНО! вложенный макет является LinearLayout
```

8.2. Gravity и Gravity макета

android:layout_gravity

- android:layout_gravity используется для установки положения элемента в его родителе (например, дочерний View внутри макета).
- Поддерживается LinearLayout и FrameLayout

android:gravity

- android:gravity используется для установки положения содержимого внутри элемента (например, текста внутри TextView).

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    android:orientation="vertical">  
  
<LinearLayout  
    android:layout_width="wrap_content"  
    android:layout_height="0dp"  
    android:layout_weight="1"  
    android:orientation="vertical"  
    android:layout_gravity="left"  
    android:gravity="center_vertical">  
  
<TextView  
    android:layout_width="@dimen/fixed"  
    android:layout_height="wrap_content"  
    android:text="@string/first"  
    android:background="@color/colorPrimary"  
    android:gravity="left"/>  
  
<TextView  
    android:layout_width="@dimen/fixed"  
    android:layout_height="wrap_content"  
    android:text="@string/second"  
    android:background="@color/colorPrimary"  
    android:gravity="center"/>  
  
<TextView  
    android:layout_width="@dimen/fixed"  
    android:layout_height="wrap_content"  
    android:text="@string/third"
```

```
        android:background="@color/colorPrimary"
        android:gravity="right"/>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:layout_gravity="center"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorAccent"
        android:gravity="left"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorAccent"
        android:gravity="center"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorAccent"
        android:gravity="right"/>

</LinearLayout>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical"
    android:layout_gravity="right"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimaryDark"
        android:gravity="left"/>

    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimaryDark"
```

```

        android:gravity="center" />

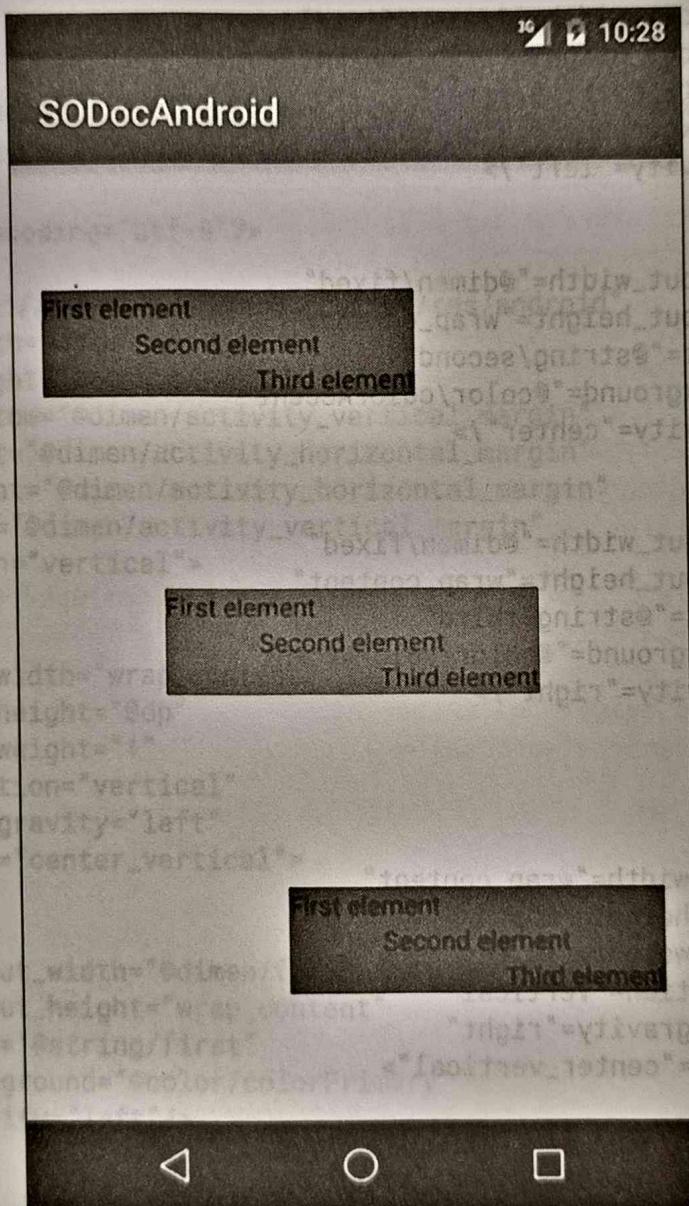
    <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimaryDark"
        android:gravity="right" />

</LinearLayout>

</LinearLayout>

```

Что дает следующий результат:



8.3. Поведение прокрутки CoordinatorLayout

Вложенный CoordinatorLayout может использоваться для достижения эффектов прокрутки материального дизайна (Material Design Scrolling Effects) при использовании внутренних макетов, поддерживающих вложенную прокрутку (Nested Scrolling), таких как NestedScrollView или RecyclerView.

Для данного примера:

- `app:layout_scrollFlags="scroll|enterAlways"` используется в свойствах панели инструментов
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` используется в свойствах ViewPager
- Во фрагментах ViewPager используется RecyclerView

Здесь представлен xml-файл макета, используемый в Activity:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
            app:elevation="0dp"
            app:layout_scrollFlags="scroll|enterAlways"
            />

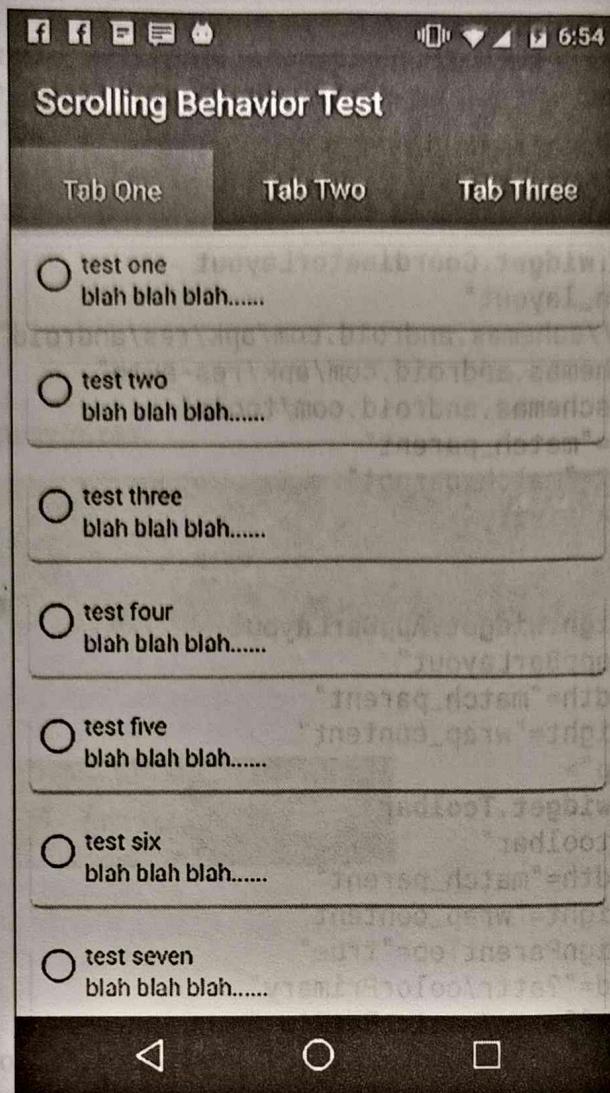
        <android.support.design.widget.TabLayout
            android:id="@+id/tab_layout"
            app:tabMode="fixed"
            android:layout_below="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="?attr/colorPrimary"
            app:elevation="0dp"
            app:tabTextColor="#d3d3d3"
            android:minHeight="?attr/actionBarSize"
            />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_below="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

</android.support.design.widget.CoordinatorLayout>
```

Результат:



8.4. Процентные макеты

Библиотека Percent Support Library (<http://developer.android.com/tools/support-library/features.html#percent>) предоставляет `PercentFrameLayout` и `PercentRelativeLayout`, где `ViewGroup`, которые обеспечивают простой способ задания **размеров и полей** представления в **процентах** от общего размера.

Вы можете использовать библиотеку Percent Support Library, добавив к своим зависимостям следующее:

```
compile 'com.android.support:percent:25.3.1'
```

Если необходимо отобразить вид, заполняющий экран по горизонтали, но только половину экрана по вертикали, то это можно сделать следующим образом.

```
<android.support.percent.PercentFrameLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <FrameLayout
```

```
        app:layout_widthPercent="100%"
        app:layout_heightPercent="50%"
```

8.5. Вес вида

```
    android:background="@android:color/black" />
```

```
<android.support.percent.PercentFrameLayout>
```

Можно также определить процентное соотношение в отдельном XML-файле с помощью такого кода:

```
<fraction name="margin_start_percent">25%</fraction>
```

Ссылаться на это в своих макетах можно при помощи @fraction/margin_start_percent.

Кроме того можно установить пользовательское **соотношение сторон** с помощью app:layout_aspectRatio. Это позволяет задать только одно измерение, например, только ширину, а высота будет определяться автоматически на основе заданного соотношения сторон, будь то 4:3, 16:9 или даже квадратное соотношение сторон 1:1.

Например:

```
<ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background" />
```

8.5. Вес вида

Одним из наиболее часто используемых атрибутов LinearLayout является **вес** (weight, см. <https://developer.android.com/guide/topics/ui/layout/linear.html#Weight>) его дочерних представлений. Вес определяет, сколько места будет занимать вид по сравнению с другими видами внутри LinearLayout.

Вес используется в тех случаях, когда необходимо выделить определенное место на экране для одного компонента по сравнению с другими.

Основные свойства:

- weightSum – это общая сумма весов всех дочерних представлений. Если не указать weightSum, то система самостоятельно вычислит сумму всех весов.
- layout_weight – задает количество места из общей суммы веса, которое будет занимать виджет.

Код:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="4">
```

<EditText

```
    android:layout_weight="2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Type Your Text Here" />
```

<Button

```
    android:layout_weight="1"
    android:layout_width="0dp"
```

```

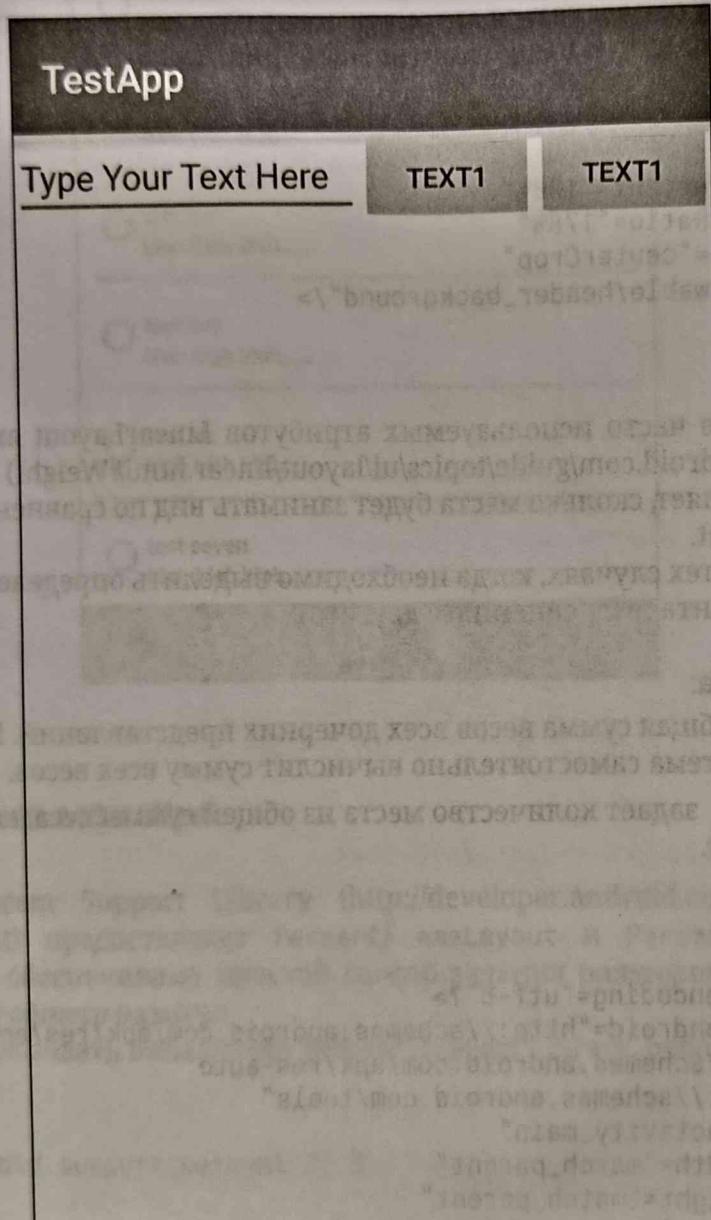
    android:layout_height="wrap_content"
    android:text="Text1" />

<Button
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Text1" />

</LinearLayout>

```

На выходе получаем:



3.4. Примеры

Библиотека `Res/layout/activity_main.xml` в папке `src/main/res/layout` предоставляет базовый макет для приложения. Установка этого макета в `setContentView` в `onCreate` приведет к следующему виду:

Вы можете увидеть, что текстовое поле занимает 2/3 экрана, а остальные 1/3 отданы двум кнопкам.

Чтобы изменить это поведение, можно изменить атрибуты `layout_width` и `layout_height`.

Теперь, даже если размер устройства больше, `EditText` будет занимать 2/4 площади экрана. Таким образом, внешний вид вашего приложения будет неизменным на всех экранах.

Примечание: здесь `layout_width` сохраняется равным `0dp`, так как пространство виджетов делится по горизонтали. Если виджеты должны быть выровнены по вертикали, то `layout_height` будет установлен в `0dp`. Это сделано для повышения эффективности кода, поскольку во время выполнения система не будет пытаться вычислить ширину или высоту соответственно, так как этим управляет вес. Если бы вместо этого использовалось значение `wrap_content`, то система попыталась бы сначала вычислить ширину/высоту, прежде чем применить атрибут `weight`, что вызвало бы еще один цикл вычислений.

8.6. Создание LinearLayout программным способом

Иерархия:

```
LinearLayout(horizontal)
    ImageView
    LinearLayout(vertical)
        TextView
        TextView
```

Код:

```
LinearLayout rootView = new LinearLayout(context);
rootView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
rootView.setOrientation(LinearLayout.HORIZONTAL);

// для просмотра изображений
ImageView imageView = new ImageView(context);
// для горизонтальной линейной раскладки
LinearLayout linearLayout2 = new LinearLayout(context);
linearLayout2.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_
PARENT, LayoutParams.WRAP_CONTENT));
linearLayout2.setOrientation(LinearLayout.VERTICAL);

TextView tv1 = new TextView(context);
TextView tv2 = new TextView(context);
// добавляем 2 текстовых вида в горизонтальный линейный макет
linearLayout2.addView(tv1);
linearLayout2.addView(tv2);

// наконец, добавляем imageView и горизонтальный linearlayout к вертикальному
linearLayout (rootView)
rootView.addView(imageView);
rootView.addView(linearLayout2);
```

8.7. LinearLayout

LinearLayout – это ViewGroup, которая располагает свои дочерние элементы в один столбец или в одну строку. Ориентация может быть задана вызовом метода `setOrientation()` или с помощью xml-атрибута `android:orientation`.

1. Верткальная ориентация: `android:orientation="vertical"`

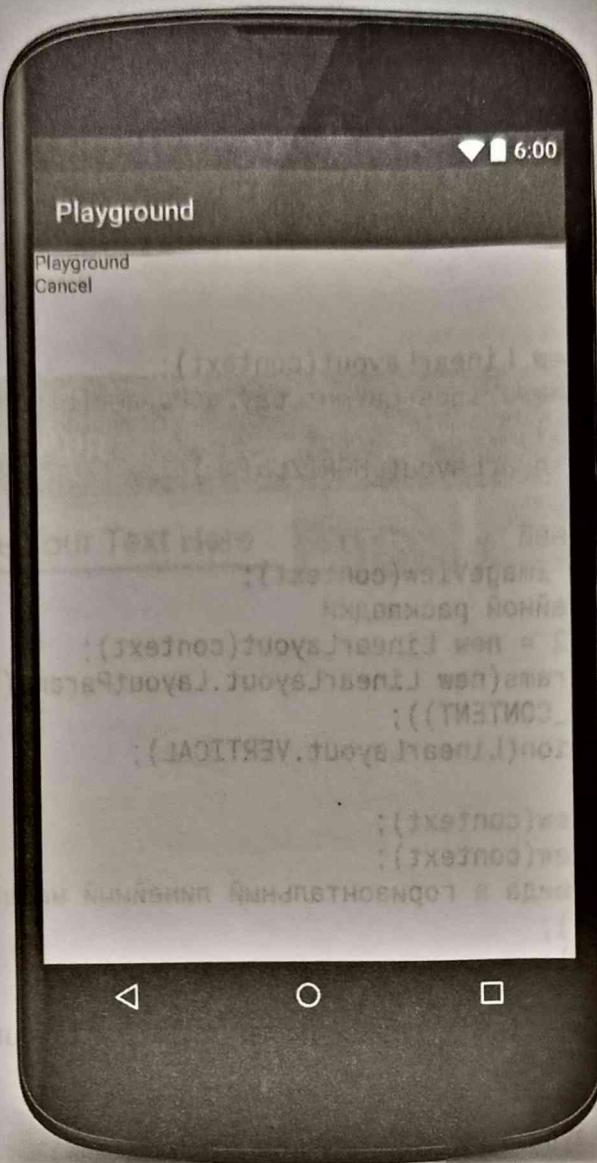
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/cancel" />

</LinearLayout>
```

Ниже приведен скриншот, как это будет выглядеть:



2. Горизонтальная ориентация: android:orientation="horizontal"

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/app_name" />  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@android:string/cancel" />
```

LinearLayout также поддерживает присвоение веса отдельным дочерним элементам с помощью атрибута android:layout_weight.

8.8. RelativeLayout

RelativeLayout – это ViewGroup, которая отображает дочерние представления в относительном положении. По умолчанию все дочерние представления рисуются на уровне в левом верхнем углу макета, поэтому необходимо определить положение каждого представления с помощью различных свойств макета, доступных в RelativeLayout.LayoutParams. Значение каждого свойства макета – это либо булево значение, позволяющее определить положение макета относительно родительского RelativeLayout, либо идентификатор, ссылаю-

щийся на другое представление в макете, относительно которого должно быть позиционировано представление.

Пример:

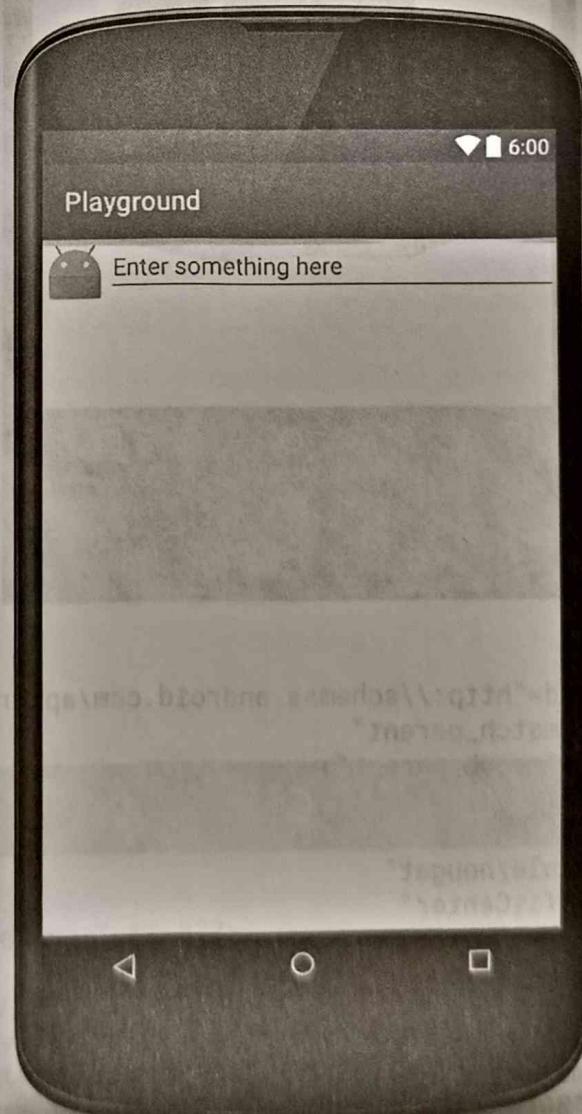
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@mipmap/ic_launcher" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_toRightOf="@+id/imageView"
        android:layout_toEndOf="@+id/imageView"
        android:hint="@string/hint" />

</RelativeLayout>
```

Ниже приведен скриншот, как это будет выглядеть:

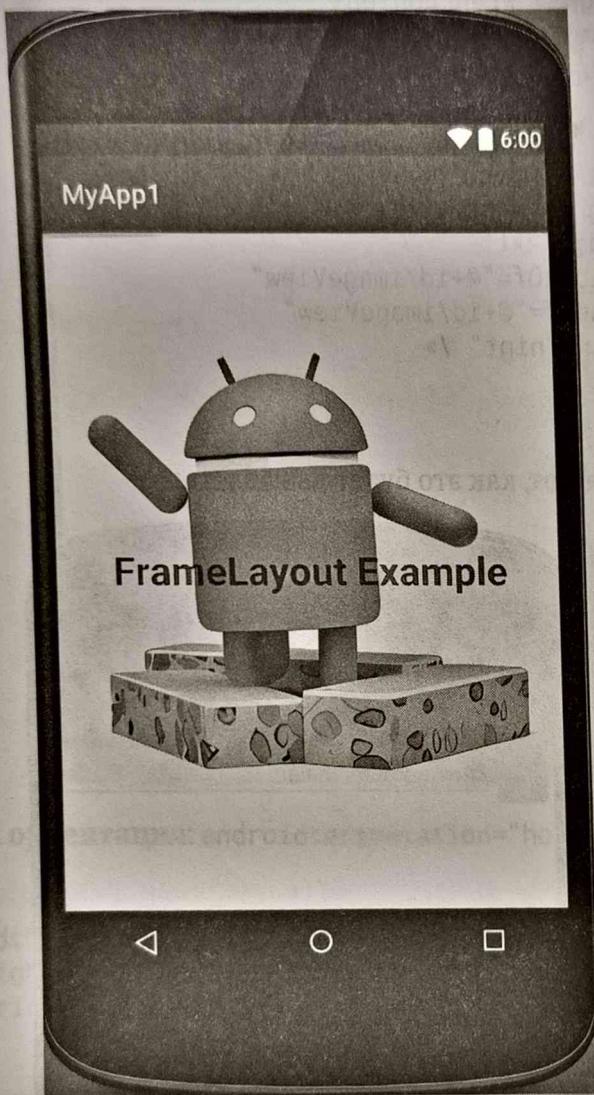


8.9. FrameLayout

FrameLayout предназначен для выделения на экране области для отображения одного элемента. Однако можно добавить несколько дочерних элементов в FrameLayout и управлять их положением внутри FrameLayout, назначив Gravity каждому из них с помощью атрибута `android:layout_gravity`.

Как правило, FrameLayout используется для размещения одного дочернего представления. Частыми случаями использования являются создание держателей для «раздувания» фрагментов в Activity, наложение представлений друг на друга или применение переднего плана к представлениям.

Приведем пример и его код:



```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:src="@drawable/nougat"
        android:scaleType="fitCenter"
        android:layout_height="match_parent"
        android:layout_width="match_parent"/>

    <TextView
        android:text="FrameLayout Example"
        android:textSize="30sp"
```

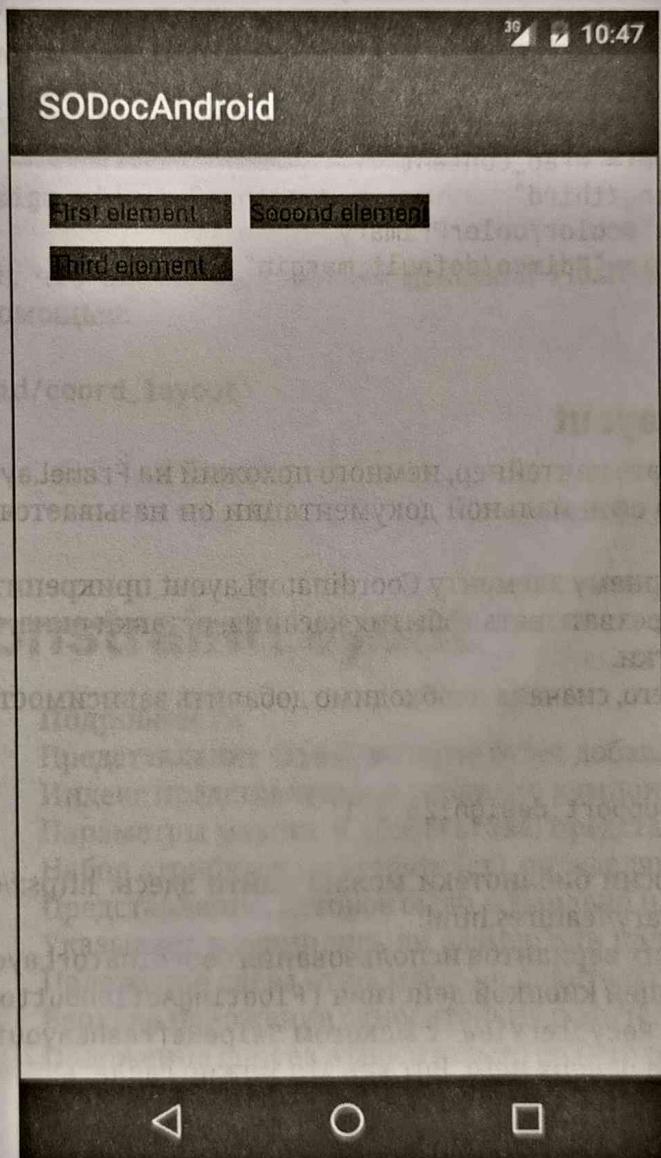
```
    android:textStyle="bold"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:gravity="center"/>"

</FrameLayout>
```

8.10. GridLayout

GridLayout, как следует из названия, представляет собой макет, используемый для расположения представлений в виде сетки. GridLayout делится на столбцы и строки. Как видно из приведенного ниже примера, количество столбцов и/или строк задается свойствами `columnCount` и `rowCount`. При добавлении представлений в этот макет первое представление будет добавлено в первый столбец, второе представление – во второй столбец, а третье представление – в первый столбец второй строки.

Приведем скриншот и его код:



```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:columnCount="2"
    android:rowCount="2">

    <Button
        android:id="@+id/button1"
        android:layout_column="0"
        android:layout_row="0"/>
    <Button
        android:id="@+id/button2"
        android:layout_column="1"
        android:layout_row="0"/>
    <Button
        android:id="@+id/button3"
        android:layout_column="0"
        android:layout_row="1"/>
</GridLayout>
```

```

    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:columnCount="2"
    android:rowCount="2">

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/first"
    android:background="@color/colorPrimary"
    android:layout_margin="@dimen/default_margin" />

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/second"
    android:background="@color/colorPrimary"
    android:layout_margin="@dimen/default_margin" />

<TextView
    android:layout_width="@dimen/fixed"
    android:layout_height="wrap_content"
    android:text="@string/third"
    android:background="@color/colorPrimary"
    android:layout_margin="@dimen/default_margin" />

</GridLayout>

```

8.11. CoordinatorLayout

CoordinatorLayout – это контейнер, немного похожий на FrameLayout, но с дополнительными возможностями, в официальной документации он называется super-powered (сверхмощный) FrameLayout.

Если к прямому дочернему элементу CoordinatorLayout прикрепить CoordinatorLayout.Behavior, вы сможете перехватывать события касания, вставки окна, измерения, компоновки и вложенной прокрутки.

Чтобы использовать его, сначала необходимо добавить зависимость для библиотеки поддержки в файл gradle:

```
compile 'com.android.support:design:25.3.1'
```

Номер последней версии библиотеки можно найти здесь: <https://developer.android.com/topic/libraries/support-library/features.html>.

Одним из практических вариантов использования CoordinatorLayout является создание представления с плавающей кнопкой действия (FloatingActionButton). В данном конкретном случае мы создадим RecyclerView с макетом SwipeRefreshLayout и плавающей кнопкой FloatingActionButton поверх него. Вот как это можно сделать:

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coord_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
```

```

<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/swipe_refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/recycler_view"/>

</android.support.v4.widget.SwipeRefreshLayout>

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:clickable="true"
    android:color="@color/colorAccent"
    android:src="@mipmap/ic_add_white"
    android:layout_gravity="end|bottom"
    app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>

```

Обратите внимание, что плавающая кнопка действия FloatingActionButton привязана к CoordinatorLayout с помощью:

`app:layout_anchor="@+id/coord_layout"`

Глава 10. Textview и TextView

Глава 9. ConstraintLayout

Параметр

child
index
params
attrs
view
changed
left
top
right
bottom
widthMeasureSpec

heightMeasureSpec

layoutDirection
a
widthAttr
heightAttr

Подробности

Представление (View), которое будет добавлено к макету
Индекс представления в иерархии компоновки
Параметры макета (LayoutParams) представления
Набор атрибутов (AttributeSet), определяющий LayoutParams
Представление, которое было добавлено или удалено
Указывает, изменились ли размер или положение представления
Положение слева относительно родительского представления
Верхнее положение относительно родительского представления
Положение справа относительно родительского представления
Нижнее положение относительно родительского представления
Требования к горизонтальному пространству, накладываемые родительским представлением
Требования к вертикальному пространству, накладываемые родительским представлением
—
—
—
—

`ConstraintLayout` – это группа `ViewGroup`, позволяющая гибко позиционировать и изменять размеры виджетов. Она совместима с Android 2.3 (уровень API 9) и выше.

`ConstraintLayout` позволяет создавать большие и сложные макеты с плоской иерархией представлений. Сходство с `RelativeLayout` в том, что все представления располагаются в соответствии с отношениями между дочерними представлениями и родительским макетом, но `ConstraintLayout` более гибкий, чем `RelativeLayout`, и его проще использовать в редакторе макетов Android Studio.

9.1. Добавление ConstraintLayout в проект

Для работы с `ConstraintLayout` требуется Android Studio версии 2.2 или новее и наличие `Android Support Repository` не ниже 32-й версии.

1. Добавьте библиотеку `Constraint Layout` в качестве зависимости в файл `build.gradle`:

```
dependencies {
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Синхронизируйте проект:

Чтобы добавить в проект новый `ConstraintLayout`, выполните следующие действия:

1. Щелкните правой кнопкой мыши на каталоге макетов вашего модуля, затем выберите `New > XML > Layout XML`.
2. Введите имя для макета и введите `"android.support.constraint.ConstraintLayout"` для корневого тега.
3. Нажмите `Finish` (Готово).

Или просто добавьте в файл макета:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout>
```

9.2. Цепочки (Chains)

Начиная с версии `ConstraintLayout alpha 9` стали доступны **цепочки** (Chains). Цепочка – это набор представлений внутри `ConstraintLayout`, которые связаны между собой двунаправленно, то есть **A** связано с **B** ограничением, а **B** связано с **A** другим ограничением.

Пример:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- это представление связано с bottomTextView -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView" />
```

```

    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_chainPacked="true" />

<!-- это представление одновременно связано с topTextView -->
<TextView
    android:id="@+id/bottomTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bottom\nMkay"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/topTextView" />

</android.support.constraint.ConstraintLayout>

```

В данном примере два вида расположены один под другим и оба центрированы по вертикали. Вертикальное положение этих представлений можно изменить, настроив **смещение (bias)** цепочки. Добавьте следующий код в первый элемент цепочки:

```
app:layout_constraintVertical_bias="0.2"
```

В вертикальной цепочке первым элементом является крайний верхний вид, а в горизонтальной – крайний левый вид. Первый элемент определяет поведение всей цепочки.

Цепочки – это новая функция, которая часто обновляется. Официальная документация Android Documentation по цепочкам: <https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html>.

Глава 10. TextInputLayout

TextInputLayout был введен для отображения плавающей метки на EditText. Для отображения плавающей метки EditText должен быть обернут TextInputLayout.

10.1. Базовое использование

Рассмотрим базовое использование TextInputLayout.

Обязательно добавьте зависимость в файл build.gradle, как описано в разделе замечаний. Пример:

```

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/username" />

</android.support.design.widget.TextInputLayout>

```

10.2. Переключатели видимости пароля

При вводе пароля можно также включить пиктограмму, которая может показывать или скрывать весь текст с помощью атрибута passwordToggleEnabled (см. <https://material.google.com/components/text-fields.html#text-fields-password-input>).

Возможные настройки:

- `passwordToggleDrawable`: для изменения значка глаза по умолчанию.
- `passwordToggleTint`: применение оттенка к отрисовываемому элементу переключения видимости пароля.
- `passwordToggleTintMode`: для указания режима смешивания, используемого для применения фонового оттенка.

Пример:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleContentDescription="@string/description"
    app:passwordToggleDrawable="@drawable/another_toggle_drawable"
    app:passwordToggleEnabled="true">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

10.3. Добавление подсчета символов

В `TextInputLayout` имеется **счетчик символов** (см. <https://material.google.com/components/text-fields.html#text-fields-character-counter>) для определенного в нем `EditText`. Счетчик будет отображаться под `EditText`.

Достаточно воспользоваться методами `setCounterEnabled()` и `setCounterMaxLength()`:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

или атрибуты `app:counterEnabled` и `app:counterMaxLength` в xml.

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

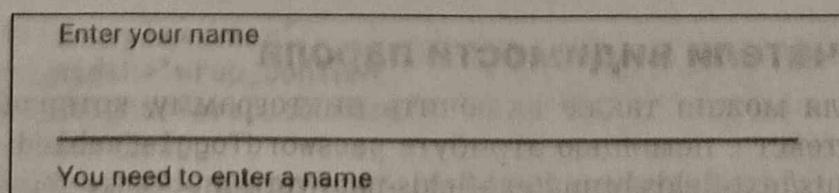
10.4. Обработка ошибок

Вы можете использовать `TextInputLayout` для отображения сообщений об ошибках в соответствии с рекомендациями (<https://material.google.com/patterns/errors.html#error-user-input-errors>), используя методы `setError` и `setErrorEnabled`.

Для того чтобы вывести ошибку под `EditText`, используйте:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Для включения ошибки в `TextInputLayout` можно использовать `app:errorEnabled="true"` в xml, либо `til.setErrorEnabled(true)`; как показано выше. Вы получите:



10.5. Настройка внешнего вида TextInputLayout

Вы можете настроить внешний вид TextInputLayout и встроенного в него EditText, определив пользовательские стили в файле styles.xml. Определенные стили могут быть добавлены как стили или темы к TextInputLayout.

Пример настройки внешнего вида подсказки:

styles.xml:

```
<!--Стиль текста плавающей метки-->
<style name="MyHintStyle" parent="TextAppearance.AppCompat.Small">
    <item name="android:textColor">@color/black</item>
</style>

<!--Стиль поля ввода-->
<style name="MyEditText" parent="Theme.AppCompat.Light">
    <item name="colorControlNormal">@color/indigo</item>
    <item name="colorControlActivated">@color/pink</item>
</style>
```

Для применения стиля обновите TextInputLayout и EditText следующим образом:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:hintTextAppearance="@style/MyHintStyle">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string>Title"
        android:theme="@style/MyEditText" />
</android.support.design.widget.TextInputLayout>
```

Пример настройки акцентного цвета в TextInputLayout. Цвет акцента влияет на цвет базовой линии текста EditText и цвет текста плавающей подсказки:

styles.xml:

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

файл макета:

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/textInputLayout_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/TextInputLayoutWithPrimaryColor">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/textInputEditText_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/login_hint_password"
        android:inputType="textPassword" />
</android.support.design.widget.TextInputLayout>
```

10.6. TextInputEditText

`TextInputEditText` – это `EditText` с дополнительным исправлением для отображения подсказки в IME в режиме 'extract' ('extract' mode) (см. <http://developer.android.com/reference/android/inputmethodservice/InputMethodService.html#FullscreenMode>).

Extract mode (Режим извлечения) – это режим, в который переключается клавиатурный редактор при нажатии на `EditText`, когда пространство слишком мало (например, альбомная ориентация на смартфоне).

В данном случае, используя `EditText` во время редактирования текста, можно увидеть, что IME не дает подсказки о том, что вы редактируете.

`TextInputEditText` устраняет эту проблему, предоставляя текст подсказки, когда IME устройства пользователя находится в режиме извлечения. Пример:

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Description"
    >
    <android.support.design.widget.TextInputEditText
        android:id="@+id/description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.design.widget.TextInputLayout>
```

Глава 11. CoordinatorLayout и поведение

`CoordinatorLayout` – это супермощный (super-powered) `FrameLayout`; цель этой группы представлений – координировать представления, находящиеся внутри нее.

Основная привлекательность `CoordinatorLayout` заключается в возможности координировать анимацию и переходы представлений в самом XML-файле.

`CoordinatorLayout` предназначен для двух основных вариантов использования:

- В качестве декора верхнего уровня приложения или Chrome-макета.
- Как контейнер для конкретного взаимодействия с одним или несколькими дочерними представлениями.

11.1. Создание простого поведения

Для создания поведения (Behavior) достаточно расширить класс `CoordinatorLayout.Behavior`.

Расширение `CoordinatorLayout.Behavior`

Пример:

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {

    /**
     * Конструктор по умолчанию.
     */
    public MyBehavior() {
    }

    /**
     * Конструктор по умолчанию для "раздувания" MyBehavior из layout.
     */
```

```

*
@param context Контекст {@link Context}.
@param attrs Набор атрибутов {@link AttributeSet}.
*/
public MyBehavior(Context context, AttributeSet attrs) {
    super(context, attrs);
}
}

```

Для вызова этого поведения необходимо прикрепить его к дочернему представлению CoordinatorLayout.

Присоединение поведения программным способом

```

MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) view.
getLayoutParams(); params.setBehavior(myBehavior);

```

Вложение поведения в XML

Для закрепления поведения в XML можно использовать атрибут `layout_behavior`:

```

<View
    android:layout_height="..." android:layout_width="..."
    app:layout_behavior=".MyBehavior" />

```

Автоматическое присоединение поведения

Если вы работаете с пользовательским представлением, то можете прикрепить поведение, используя аннотацию `@CoordinatorLayout.DefaultBehavior`:

```

@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends....{
}

```

11.2. Использование поведения SwipeDismissBehavior

Поведение `SwipeDismissBehavior` работает на любом представлении и реализует функциональность смахивания для закрытия (отклонения) в макетах с `CoordinatorLayout`. Просто используйте:

```

final SwipeDismissBehavior<MyView> swipe = new SwipeDismissBehavior();

//Устанавливает направление смахивания для данного поведения.
swipe.setSwipeDirection(
    SwipeDismissBehavior.SWIPE_DIRECTION_ANY);

//Установка слушателя, который будет использоваться при наступлении события
//отклонения (dismiss event)
swipe.setListener(
    new SwipeDismissBehavior.OnDismissListener() {
        @Override public void onDismiss(View view) {
            //.....
        }
        @Override
        public void onDragStateChanged(int state) {
            //.....
        }
    });

```

```
//Присоединение поведения SwipeDismissBehavior к представлению
LayoutParams coordinatorParams = (LayoutParams) mView.getLayoutParams();
coordinatorParams.setBehavior(swipe);
```

11.3. Создание зависимостей между представлениями

Для создания зависимостей между представлениями можно использовать CoordinatorLayout.Behavior. Вы можете привязать представление к другому представлению:

- с помощью атрибута `layout_anchor`.
- с помощью создания пользовательского Behavior и реализации метода `layoutDependsOn`, возвращающего `true`.

Например, чтобы создать поведение для перемещения ImageView при перемещении другого (например Toolbar), выполните следующие действия.

- Создайте пользовательское поведение (Behavior):

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> { ... }
```

- Переопределите метод `layoutDependsOn`, возвращающий значение `true`. Этот метод вызывается каждый раз, когда происходит изменение макета:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent, ImageView child, View dependency) {
    // Возвращает true для добавления зависимости.
    return dependency instanceof Toolbar;
}
```

- Всякий раз, когда метод `layoutDependsOn` возвращает `true`, вызывается метод `onDependentViewChanged`:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, ImageView child, View dependency) {
    // Реализуйте здесь анимации, переводы или движения, всегда связанные
    // с предоставленной зависимостью.
    float translationY = Math.min(0, dependency.getTranslationY() - dependency.
        getHeight()); child.setTranslationY(translationY);
}
```

Глава 12. TabLayout

12.1. Использование TabLayout без ViewPager

Чаще всего TabLayout используется вместе с ViewPager, чтобы получить функциональность пролистывания, которая поставляется вместе с ним. Но возможно использование TabLayout и без ViewPager, с помощью TabLayout.OnTabSelectedListener.

Для начала добавьте TabLayout в XML-файл вашей активности:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Для навигации внутри Activity вручную заполните пользовательский интерфейс в зависимости от выбранной вкладки:

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        switch (tab.getPosition()) {
            case 1:
                getSupportFragmentManager().beginTransaction()
                    .replace(R.id.fragment_container, new ChildFragment()).commit();
                break;
            // Продолжить для каждой вкладки в TabLayout
        }
    }
    @Override
    public void onTabUnselected(TabLayout.Tab tab) {
    }
    @Override
    public void onTabReselected(TabLayout.Tab tab) {
    }
});
```

Глава 13. ViewPager

ViewPager – это менеджер макетов, позволяющий пользователю листать влево и вправо страницы с данными. Чаще всего он используется совместно с фрагментами, что представляет собой удобный способ предоставления и управления жизненным циклом каждой страницы.

13.1. ViewPager с точечным индикатором



Все, что нам нужно, это ViewPager, TabLayout и два drawable-объекта для выделенных точек и точек по умолчанию.

Прежде всего нам необходимо добавить TabLayout в макет экрана и связать его с ViewPager. Это можно сделать двумя способами:

Вложенный TabLayout в ViewPager

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

В этом случае TabLayout будет автоматически связан с ViewPager, но будет находиться рядом с ViewPager, а не над ним.

Отдельная вкладка TabLayout

```
<android.support.v4.view.ViewPager
    android:id="@+id/photos_viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
```

В этом случае мы можем разместить TabLayout в любом месте, но нам необходимо связать его с ViewPager программно.

```
ViewPager pager = (ViewPager) view.findViewById(R.id.photos_viewpager);
PagerAdapter adapter = new PhotosAdapter getChildFragmentManager(), photosUrl);
pager.setAdapter(adapter);
```

```
TabLayout tabLayout = (TabLayout) view.findViewById(R.id.tab_layout);
tabLayout.setupWithViewPager(pager, true);
```

После того как мы создали макет, необходимо подготовить точки. Для этого мы создаем три файла: selected_dot.xml, default_dot.xml и tab_selector.xml.

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@color/colorAccent"/>
        </shape>
    </item>
</layer-list>
```

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            android:innerRadius="0dp" android:shape="ring" android:thickness="8dp"
            android:useLevel="false">
                <solid android:color="@android:color/darker_gray"/>
            </shape>
        </item>
    </layer-list>
```

tab_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/selected_dot"
          android:state_selected="true"/>

    <item android:drawable="@drawable/default_dot"/>
</selector>
```

Теперь нам нужно добавить всего три строки кода в TabLayout в нашем xml-макете, и все будет готово.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

13.2. Базовое использование ViewPager с фрагментами

ViewPager позволяет отображать в работе несколько фрагментов, навигация по которым осуществляется с помощью пролистывания влево или вправо. Передавать ему нужно либо представления, либо фрагменты с помощью PagerAdapter.

Существуют две более специфические реализации, которые будут наиболее полезны при использовании фрагментов, – FragmentPagerAdapter и FragmentStatePagerAdapter. Когда фрагмент необходимо инстанцировать в первый раз, для каждой позиции, требующей инстанцирования, будет вызван метод getItem(position). Метод getCount() возвращает общее количество страниц, чтобы ViewPager знал, сколько фрагментов необходимо показать.

Как FragmentPagerAdapter, так и FragmentStatePagerAdapter хранят кэш фрагментов, которые необходимо показать ViewPager. По умолчанию ViewPager будет стараться хранить максимум три фрагмента, которые соответствуют текущему видимому фрагменту, а также следующим справа и слева. Также FragmentStatePagerAdapter будет хранить состояние каждого из ваших фрагментов.

Необходимо иметь в виду следующее: обе реализации предполагают, что фрагменты будут сохранять свои позиции, поэтому если вы храните список фрагментов, а не имеете их статическое количество, как это видно из метода getItem(), то вам необходимо создать подкласс PagerAdapter и переопределить, по крайней мере, методы instantiateItem(), destroyItem() и getItemPosition().

Просто добавьте ViewPager в свой макет, как описано в базовом примере:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager>
```

```

    android:id="@+id/vpPager">
</android.support.v4.view.ViewPager>
</LinearLayout>

```

Затем определите адаптер, который будет определять, сколько страниц существует и какой фрагмент отображать для каждой страницы адаптера.

```

public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        // Применение адаптера
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter {

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

        // Возвращает фрагмент для отображения на данной странице
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();
                case 1:
                    return new Fragment2();
                case 2:
                    return new Fragment3();
                default:
                    return null;
            }
        }

        // Возвращает общее количество страниц
        @Override
        public int getCount() {
            return 3;
        }
    }
}

```

Если вы используете android.app.Fragment, то необходимо добавить эту зависимость:

```
compile 'com.android.support:support-v13:25.3.1'
```

Если вы используете android.support.v4.app.Fragment, то необходимо добавить эту зависимость:

```
compile 'com.android.support:support-fragment:25.3.1'
```

13.3. ViewPager с фрагментом PreferenceFragment

До недавнего времени использование android.support.v4.app.FragmentPagerAdapter не позволяло использовать PreferenceFragment в качестве одного из фрагментов, используемых в FragmentPagerAdapter.

В последних версиях библиотеки support v7 появился класс PreferenceFragmentCompat (информацию о нем см. на ресурсе <http://developer.android.com/intl/es/reference/android/support/v7/preference/PreferenceFragmentCompat.html>), который будет работать с ViewPager и v4-версией FragmentPagerAdapter.

Пример фрагмента, расширяющего PreferenceFragmentCompat:

```
import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {
        // Требуется пустой открытый конструктор
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.fragment_settings_pref);
    }

    @Override
    public void onPreferenceSelected(Preference preference) {
    }

    @Override
    public void onCreatePreferences(Bundle bundle, String s) {
    }
}
```

После этого данный фрагмент можно использовать в подклассе android.support.v4.app.FragmentPagerAdapter:

```
private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new FragmentOne();
            case 1:
                return new FragmentTwo();
            case 2:
                return new MySettingsPrefFragment();
        }
    }
}
```

```

    default:
        return null;
    }
}

// .....
}

```

13.4. Добавление ViewPager

Убедитесь, что в файл build.gradle вашего приложения в разделе зависимостей добавлена следующая зависимость:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Затем добавьте ViewPager в макет активности:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

После этого определите свой PagerAdapter:

```

public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {
        // Создать страницу для заданной позиции. Например:
        LayoutInflater inflater = LayoutInflater.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection,
            false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
        // Удаление страницы для заданной позиции. Например:
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        // Возвращает количество доступных просмотров.
        return NumberOfPages;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {

```

```

    // Определяет, связана ли страница View с конкретным ключевым объектом
    // как возвращает instantiateItem(ViewGroup, int). Например:
    return view == object;
}
}

```

Наконец, настройте ViewPager в вашей активности:

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}

```

13.5. Настройка OnPageChangeListener

Если необходимо прослушивать изменения выбранной страницы, можно реализовать слушатель (listener) ViewPager.OnPageChangeListener (см. <https://developer.android.com/reference/android/support/v4/view/ViewPager.OnPageChangeListener.html>) на ViewPager:

```

viewPager.addOnPageChangeListener(new OnPageChangeListener() {

    // Этот метод будет вызываться, когда будет выбрана новая страница.
    // Анимация не должна быть обязательно завершена.
    @Override
    public void onPageSelected(int position) {
        // Ваш код
    }

    // Этот метод будет вызываться при прокрутке текущей страницы, либо как часть
    // инициированной программно плавной прокрутки, либо прокрутки, инициированной
    // пользователем.
    @Override
    public void onPageScrolled(int position, float positionOffset, int
    positionOffsetPixels) {
        // Ваш код
    }

    // Вызывается при изменении состояния прокрутки. Полезно для определения того,
    // когда пользователь начинает перетаскивание, при котором ViewPager
    // автоматически перемещается на текущую страницу или при полной остановке.
    @Override
    public void onPageScrollStateChanged(int state) {
        // Ваш код
    }
});

```

13.6. ViewPager с TabLayout

Для упрощения навигации можно использовать TabLayout.

Установить вкладки для каждого фрагмента в адаптере можно с помощью метода TabLayout.newTab(), однако для этой задачи существует другой, более удобный и про-

стой метод – `TabLayout.setupWithViewPager()` (см. [https://developer.android.com/reference/android/support/design/widget/TabLayout.html#setupWithViewPager\(android.support.v4.view.ViewPager\)\)](https://developer.android.com/reference/android/support/design/widget/TabLayout.html#setupWithViewPager(android.support.v4.view.ViewPager)).

Этот метод синхронизирует, создавая и удаляя вкладки в соответствии с содержимым адаптера, связанного с вашим `ViewPager` при каждом его вызове.

Также будет установлен обратный вызов, чтобы каждый раз, когда пользователь перелистывает страницу, выбиралась соответствующая вкладка.

Просто определите макет при помощи:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        app:tabMode="scrollable" />

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="0px"
        android:layout_weight="1" />

</LinearLayout>
```

Затем реализуйте `FragmentPagerAdapter` и примените его к `ViewPager`:

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mPagerAdapter;
    private ViewPager mViewPager;
    private TabLayout mTabLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        // Получение ViewPager и применение PagerAdapter
        mPagerAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mPagerAdapter);

        // связываем вместе tabLayout и viewpager
        mTabLayout = (TabLayout) findViewById(R.id.tab_layout); mTabLayout.
        setupWithViewPager(mViewPager);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter {

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

        // Возвращает фрагмент для отображения на данной странице
        @Override
        public Fragment getItem(int position) {
            switch(position) {
```

```

        case 0:
            return new Fragment1();

        case 1:
            return new Fragment2();

        case 2:
            return new Fragment3();

        default:
            return null;
    }

}

// Будет отображаться в качестве метки вкладки
@Override
public CharSequence getPageTitle(int position) {
    switch(position) {
        case 0:
            return "Заголовок фрагмента 1";

        case 1:
            return "Заголовок фрагмента 2";

        case 2:
            return "Заголовок фрагмента 3";

        default:
            return null;
    }
}

// Возвращает общее количество страниц
@Override
public int getCount() {
    return 3;
}
}

</android.support.v7.widget.CardView>

```

Глава 14. CardView

Параметр

cardBackgroundColor
cardCornerRadius
cardElevation
cardMaxElevation
cardPreventCornerOverlap
cardUseCompatPadding

Подробности

Цвет фона для CardView
Радиус угла для CardView
Возышение для CardView
Максимальное возышение (elevation) для CardView
Добавление подкладок в CardView на v20 и более ранних версиях для предотвращения пересечения содержимого карточки с закругленными углами
Добавление подкладок в API v21+, чтобы иметь одинаковые измерения с предыдущими версиями. Может быть булевым значением, например "true" или "false".

Параметр	Подробности
contentPadding	Внутренняя подкладка между краями карточки и дочерними элементами CardView
contentPaddingBottom	Внутренняя подкладка между нижним краем карточки и дочерними элементами CardView
contentPaddingLeft	Внутренняя подкладка между левым краем карточки и дочерними элементами CardView
contentPaddingRight	Внутренняя подкладка между правым краем карточки и дочерними элементами CardView
contentPaddingTop	Внутренняя подкладка между верхним краем карточки и дочерними элементами CardView

В сущности, это FrameLayout с фоном и тенью со скругленными углами. CardView использует для теней свойство возвышения elevation на Lollipop и возвращается к собственной эмульсированной реализации теней на более старых платформах.

Из-за ресурсоемкости обрезки скругленных углов на платформах до Lollipop CardView не обрезает свои дочерние элементы, пересекающиеся со скругленными углами. Вместо этого он добавляет подложку, чтобы избежать такого пересечения (для изменения этого поведения см. параметр setPreventCornerOverlap(boolean)).

14.1. Начало работы с CardView

CardView является составной частью библиотеки Android Support Library и предоставляет макет для карточек. Чтобы добавить CardView в свой проект, добавьте следующую строку в зависимость build.gradle:

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Номер последней версии можно уточнить здесь: <https://developer.android.com/topic/libraries/support-library/features.html>.

Затем в макете можно добавить следующее:

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <!-- один дочерний макет, содержащий другие макеты или представления -->

</android.support.v7.widget.CardView>
```

Внутри можно добавлять другие макеты, которые будут заключены в карточку. Кроме того, CardView можно заполнить любым элементом пользовательского интерфейса и манипулировать им из кода.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_margin="5dp"
    card_view:cardBackgroundColor="#81C784"
    card_view:cardCornerRadius="12dp"
    card_view:cardElevation="3dp"
    card_view:contentPadding="4dp">
```

```

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp" >

    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:id="@+id/item_image"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="16dp"
    />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/item_title"
        android:layout_toRightOf="@+id/item_image"
        android:layout_alignParentTop="true"
        android:textSize="30sp"
    />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/item_detail"
        android:layout_toRightOf="@+id/item_image"
        android:layout_below="@+id/item_title"
    />

</RelativeLayout>
</android.support.v7.widget.CardView>

```

14.2. Добавление пульсирующей анимации

Чтобы включить пульсирующую анимацию в CardView, добавьте следующие атрибуты:

```

<android.support.v7.widget.CardView
    ...
    android:clickable="true" android:foreground="?android:attr/
    selectableItemBackground">
    ...
</android.support.v7.widget.CardView>

```

14.3. Настройка CardView

CardView задает возвышение и радиус углов по умолчанию, чтобы карточки имели единообразный вид на разных платформах.

Вы можете настроить эти значения по умолчанию, используя следующие атрибуты в xml-файле:

1. Атрибут `card_view:cardElevation` добавляет возвышение в CardView.
2. Атрибут `card_view:cardBackgroundColor` используется для настройки цвета фона CardView (можно задать любой цвет).
3. Атрибут `card_view:cardCornerRadius` используется для закругления четырех краев CardView.
4. Атрибут `card_view:contentPadding` добавляет подкладку между картой и ее дочерними элементами.

Примечание: `card_view` – это пространство имен, определенное в самом верхнем родительском представлении макета. `xmlns:card_view="http://schemas.android.com/apk/res-auto"`.

Приведем пример:

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardElevation="4dp"
    card_view:cardBackgroundColor="@android:color/white"
    card_view:cardCornerRadius="8dp"
    card_view:contentPadding="16dp">

    <!-- один дочерний макет, содержащий другие макеты или представления -->

</android.support.v7.widget.CardView>
```

Это можно сделать и программно, используя:

```
card.setCardBackgroundColor( ... );
card.setCardElevation( ... );
card.setRadius( ... );
card.setContentPadding();
```

Дополнительную информацию можно найти по адресу <https://developer.android.com/reference/android/support/v7/widget/CardView.html>.

14.4. Анимирование цвета фона CardView с помощью TransitionDrawable

```
public void setCardColorTran(CardView card) {
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
    TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
        card.setBackground(trans);
    } else {
        card.setBackgroundDrawable(trans);
    }
    trans.startTransition(5000);
}
```

</android.support.v7.widget.CardView>

Глава 15. NavigationView

15.1. Как добавить NavigationView

Чтобы использовать NavigationView, просто добавьте зависимость в файл build.gradle, как описано в разделе замечаний. Затем добавьте NavigationView в макет.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
```