**Лекция №2.**

**Создание интерфейса.**

# Все элементы интерфейса в приложении Android создаются с помо- щью объектов *View* и *ViewGroup*. Объект *View* формирует на экране элемент, с которым пользователь может взаимодействовать. Объект *ViewGroup* содержит другие объекты *View* (и *ViewGroup*) для определения макета интерфейса.

Android предоставляет коллекцию подклассов *View* и *ViewGroup*, ко- торая включает в себя обычные элементы ввода и различные модели ма- кета. Каждая группа представляет собой невидимый контейнер, в кото- ром объединены дочерние виды. Эта древовидная иерархия может быть простой или сложной (чем проще, тем лучше для производительности). Для отладки макетов можно воспользоваться инструментом *Hierarchy Viewer*. С его помощью можно просмотреть значения свойств, рамки с индикаторами заполнения или поля, а также полностью отри- сованные представления прямо во время отладки приложения на эму-

ляторе или на устройстве.

### Разработка Layout

Макет определяет визуальную структуру пользовательского ин- терфейса. Существует два способа объявить макет (рис. 2.1):

* + - * oбъявление элементов пользовательского интерфейса в XML;
      * cоздание экземпляров элементов во время выполнения (приложе- ние может программным образом создавать объекты *View* и *ViewGroup*). В приложениях Android визуальный интерфейс загружается из спе- циальных файлов XML, которые хранят разметку. Эти файлы являются

ресурсами разметки.

Объявление пользовательского интерфейса в файлах XML позво- ляет отделить интерфейс приложения от кода. В приложении могут быть определены разметки в файлах XML для различных ориентаций монитора, размеров устройств, языков и т. д. Кроме того, объявление разметки в XML позволяет легче визуализировать структуру интер- фейса и облегчает отладку.

Активность – специальный класс Java, ко- торый решает, какой макет следует исполь- зовать, и описывает, как приложение должно реагировать на действия пользователя

Макеты могут включать компо- ненты графических интерфейсов: кнопки, текстовые поля, подписи и т. д.

public class MainActivity extends AppCompatActivity {

public void functionPress(View view) { EditText username = (EditText)findView-

ById(R.id.*userName*);

Log.*i*("Username", username.getText().toString());

Toast.*makeText*(getApplicationCon- text(),username.getText().toString(), Toast.*LENGTH\_LONG*).show();

ImageView imageView = (Image-View)findView- ById(R.id.*imageIcon*);

imageView.setImageResource R.drawable.*penguins*);

}

@Override

protected void onCreate(Bundle savedInstance- State) {

super.onCreate(savedInstanceState); setContentView(R.layout.*acti-vity\_main*);

}

}

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

<RelativeLayout xmlns:android=["ht](http://schemas.an-/)t[p://schemas.an-](http://schemas.an-/) droid.com/apk/res/android"

xmlns:tools="[http://schemas.android.com/tools"](http://schemas.android.com/tools) 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" tools:context="by.bstu.pnv.activitydemo.MainActivity">

<TextView

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Input name" android:id="@+id/textView" android:textSize="15sp" android:layout\_alignParentTop="true" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="35dp" />

<EditText

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:id="@+id/userName" android:textSize="15sp" android:layout\_below="@+id/textView" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="47dp" />

<Button

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Enter" android:id="@+id/button" android:layout\_below="@+id/userName" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="57dp" android:onClick="functionPress" />

<ImageView

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:id="@+id/imageIcon" android:src="@android:drawable/sym\_def\_app\_icon" android:scaleType="fitXY" android:layout\_alignParentTop="true" android:layout\_centerHorizontal="true" />RelativeLayout>

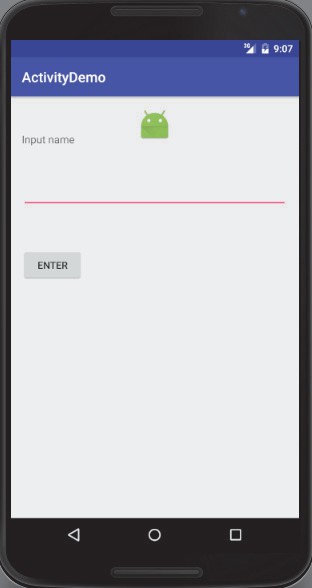


Рис. 1. Разработка макета пользовательского интерфейса Android-приложения

Файлы разметки графического интерфейса располагаются в про- екте в каталоге *res*/*layout*.

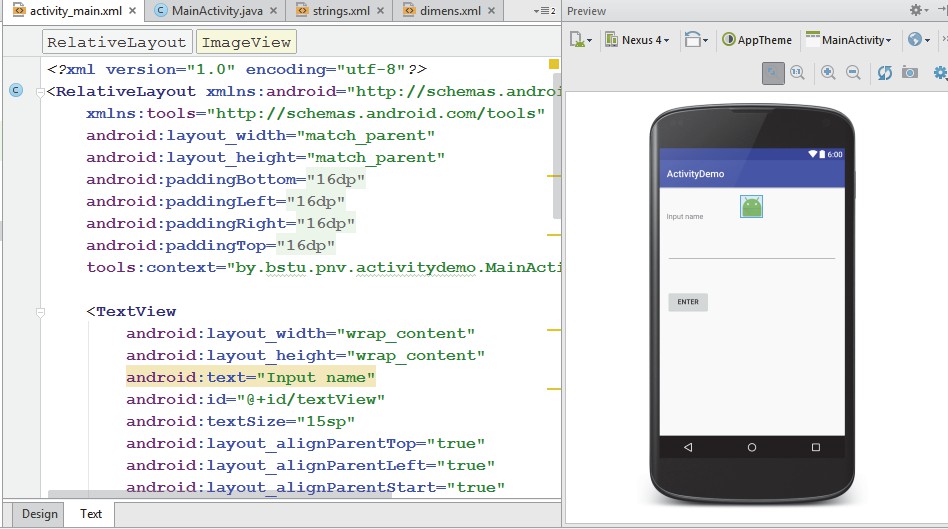


Рис. 2. Разработка макета пользовательского интерфейса на основе XML

### Разработка интерфейса в режиме дизайнера

Android Studio имеет инструментарий, который облегчает разра- ботку графического интерфейса. Можно открыть файл XML и с помо- щью кнопки *Design* переключиться в режим дизайнера к графическому представлению интерфейса в виде эскиза устройства (рис. 2.3).

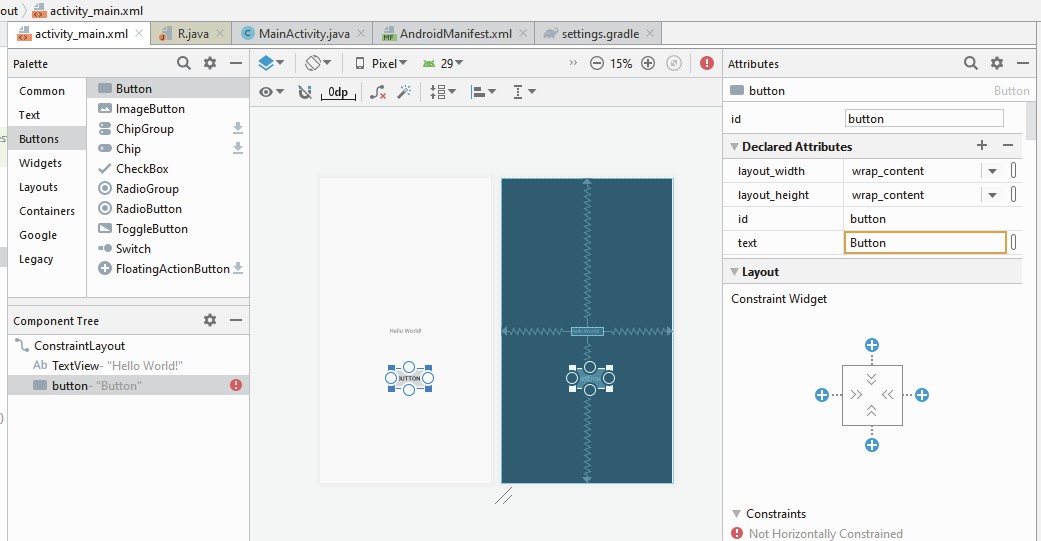


Рис. 3. Работа в режиме *Design*

Слева будет находиться панель инструментов, из которой можно перенести нужный элемент мышкой на эскиз. Все перенесенные эле- менты будут автоматически добавляться в файл XML.

При выделении элемента справа появится окно *Properties* – панель свойств выделенного элемента. Здесь можно изменить значения свойств элемента.

# Установка размеров

В операционной системе Android можно использовать различные типы измерений:

* *px* – пиксели текущего экрана. Эта единица измерения не реко- мендуется, так как реальное представление внешнего вида может изме- няться в зависимости от устройства;
* *dp* (device-independent pixels) – независимые от плотности экрана пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi (точек на дюйм). В этом случае 1 dp = 1 px. Если размер экрана больше или меньше, чем 160 dpi, коли- чество пикселей, которые применяются для отрисовки 1 dp, соответ- ственно увеличивается или уменьшается. Общая формула для получе- ния количества физических пикселей из dp: px = dp · (dpi / 160);
* *sp* (scale-independent pixels) – независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользовате- лем. Рекомендуются для работы со шрифтами;
* *pt* – 1/72 дюйма, величина базируется на физических размерах экрана;
* *mm* – миллиметры;
* *in* – дюймы.

Предпочтительными единицами для использования являются dp.

Для упрощения работы с размерами все они разбиты на несколь- ко групп:

* ldpi (low) ~120 dpi;
* mdpi (medium) ~160 dpi;
* hdpi (high) ~240 dpi;
* xhdpi (extra-high) ~320 dpi;
* xxhdpi (extra-extra-high) ~480 dpi;
* xxxhdpi (extra-extra-extra-high) ~640 dpi.

Все визуальные элементы, которые используются в приложении, как правило, упорядочиваются на экране с помощью контейнеров.

В Android подобными контейнерами служат классы *RelativeLayout*, *Lin- earLayout*, *GridLayout*, *TableLayout*, *ConstraintLayout*, *FrameLayout* и др. Все они по-разному располагают элементы и управляют ими, но есть некоторые общие моменты при компоновке визуальных компонентов. Для организации элементов внутри контейнера используются параметры разметки. Для их задания в файле XML используются атри- буты, которые начинаются с префикса *layout*\_. К таким параметрам относятся атрибуты *layout*\_*height* и *layout*\_*width*, которые использу- ются для установки размеров и могут принимать одно из следующих

значений:

* *точные размеры* элемента, например 96 dp (рис. 4);

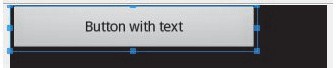


Рис. 4. Определение элемента с точными размерами

* *значение wrap\_content*: элемент растягивается до тех границ, ко- торые достаточны, чтобы вместить все его содержимое (рис. 5);

Рис. 5. Определение элемента в соответствии с содержимым

* *значение match\_parent*: элемент заполняет всю область роди- тельского контейнера (рис. 6).



Рис. 6. Определение элемента в соответствии с родительским контейнером

Можно дополнительно ограничить минимальные и максималь- ные значения с помощью атрибутов *minWidth*/*maxWidth* и *minHeight*/*maxHeight*:

android:minWidth="200dp" android:maxWidth="250dp" android:minHeight="100dp" android:maxHeight="200dp" android:layout\_height="wrap\_content" android:layout\_width="wrap\_content"

Если элемент создается в коде Java, то для установки высоты и ши- рины можно использовать метод *setLayoutParams*():

TextView textView1 = new TextView(this); textView1.setText("Hello Android"); textView1.setTextSize(26);

*// Устанавливаем размеры*

textView1.setLayoutParams(new ViewGroup.LayoutParams (ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT));

Параметры разметки позволяют задать отступы как от внешних границ элемента до границ контейнера, так и внутри самого элемента между его границами и содержимым.

Для установки внутренних отступов применяется атрибут *android*:*padding*. Он устанавливает отступы контента от всех четырех сторон контейнера. Можно устанавливать отступы только от одной стороны кон- тейнера, применяя атрибуты: *android*:*paddingLeft*, *android*:*paddingRight*, *android*:*paddingTop* и *android*:*paddingBottom* (рис. 7).

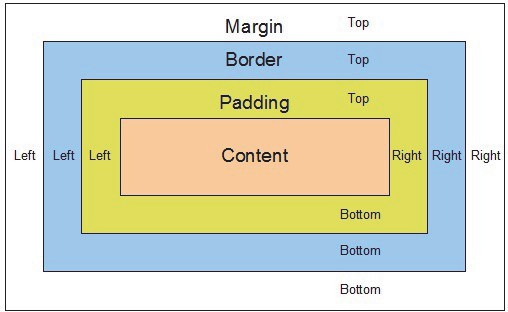


Рис. 7. Отступы элементов управления

Для установки внешних остступов используется атрибут *layout*\_*margin*. Он имеет модификации, которые позволяют задать отступ только от од- ной стороны: *android*:*layout*\_*marginBottom*, *android*:*layout*\_*marginTop*, *android*:*layout*\_*marginLeft* и *android*:*layout*\_*marginRight* (рис. 2.7):

android:layout\_marginTop="50dp" android:layout\_marginBottom="60dp" android:layout\_marginLeft="60dp" android:layout\_marginRight="60dp"

Для программной установки внутренних отступов вызыватся ме- тод *setPadding*(*left*, *top*, *right*, *bottom*), в который передаются четыре зна- чения для каждой из сторон.