**Министерство образования и науки Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего профессионального образования  
 «Севастопольский государственный университет»**

# 

**Методические указания**

**к выполнению лабораторной работы №2**

**по дисциплине**

**«Мобильные информационные технологии»**

для студентов всех форм обучения направления подготовки 09.03.02 «Информационные системы и технологии»

**Севастополь**

**2018**

Методические указания к выполнению лабораторной работы № 2 по дисциплине «Мобильные информационные технологии» для студентов всех форм обучения направления подготовки 09.03.02 «Информационные системы и технологии» /Сост. В.Е. Шишкевич ‒ Севастополь: СевГУ, 2018. ‒ 10 с.

**Лабораторная работа №2**

**Исследование особенностей интерфейса android-приложений. Основные компоненты интерфейса и их свойства.**

**1. Цель работы**

Изучить особенности создания интерфейса пользователя на Android Studio.

**2. Краткие теоретические сведения**

Два самых важных понятия в интерфейсе Android — это [Activity](https://developer.android.com/intl/ru/reference/android/app/Activity.html) и [View](https://developer.android.com/intl/ru/reference/android/view/View.html).

**Activity –** это компонент приложения, с которым может взаимодействовать пользователь для выполнения каких-нибудь действий (например, набрать номер телефона, сделать фото, отправить письмо или просмотреть карту).

Внутри Activity расположены дочерние элементы интерфейса. В конкретный момент времени в приложении отображается одно из activity и занимает весь экран, а приложение переключается между ними. В качестве примера можно рассмотреть почтовое приложение. В нем одно activity – это список писем, другое – просмотр письма, а третье – настройки ящика (рисунок 1.1)

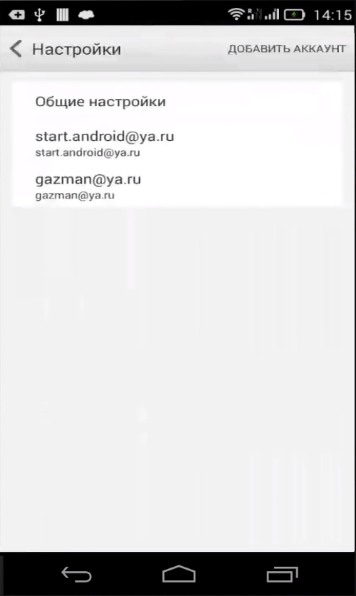
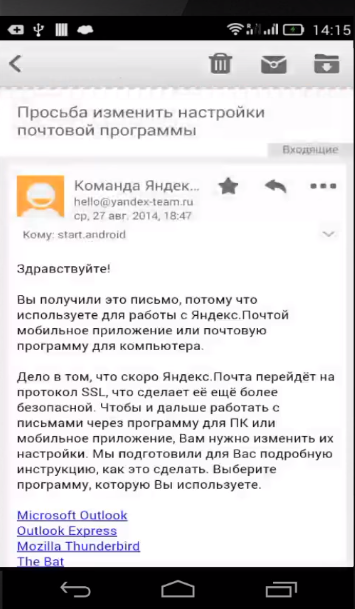
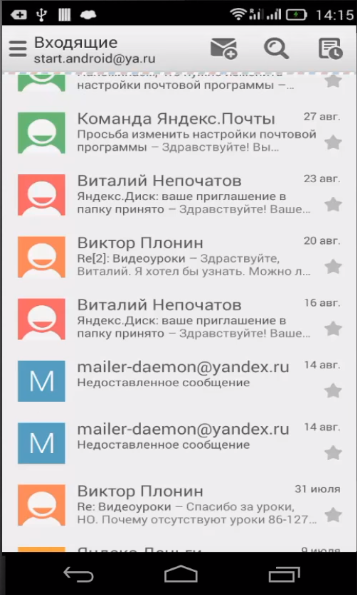


Рисунок 1.2 – Виды activity в почтовом android-приложении

**View –** элемент интерфейса. Это может быть кнопка, поле для ввода текста, контейнер для картинки, контейнер для других View и т.д. (рисунок 1.2)

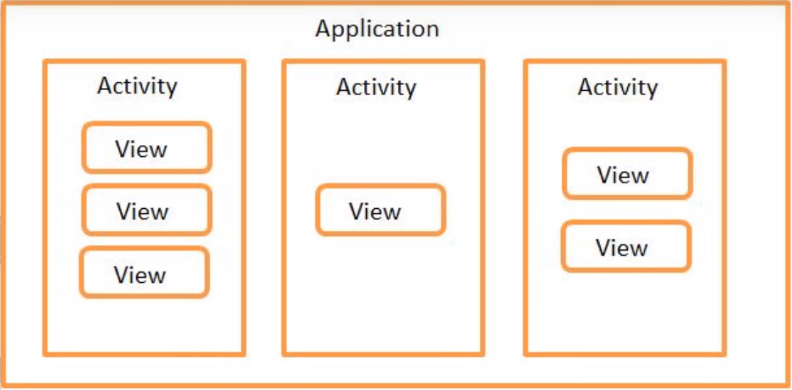


Рисунок 1.2 – Основные элементы экрана в android-приложении

Компоненты View обычно размещаются во[ViewGroup](https://developer.android.com/intl/ru/reference/android/view/ViewGroup.html).

Фактически, **ViewGroup** – это модифицированный View, созданный для того, чтобы служить контейнером для других View. Самый распространенный пример ViewGroup – это Layout.

**Layout** – общее название для нескольких наследников ViewGroup. Он служит контейнером для View и отвечает за то, как будут расположены его дочерние View на экране.

Для размещения элементов View необходимо открыть файл   
activity\_main.xml – это layout-файл, который находится в директории resources/layout. Он открывается при старте проекта по умолчанию. При запуске приложения Activity читает этот файл и отображает на экране то, что в нем описал разработчик.

**Основные параметры View-элементов**

**1) layout\_width и layout\_height**

Для ширины и высоты элементов можно указывать абсолютные значения, а можно использовать константы.

Абсолютные значения:

Используются следующие единицы измерения (ЕИ):

– *dp*или *dip* (Density-independent Pixels) – абстрактная ЕИ, позволяющая приложениям выглядеть одинаково на различных экранах и разрешениях.

– *sp***(**Scale-independent Pixels) – то же, что и dp, только используется для размеров шрифта в View элементах

– *pt*- 1/72 дюйма, определяется по физическому размеру экрана. Эта ЕИ из типографии.

– *px*– пиксел, не рекомендуется использовать т.к. на разных экранах приложение будет выглядеть по-разному.

– *mm*– миллиметр, определяется по физическому размеру экрана.

– *in*– дюйм, определяется по физическому размеру экрана.

Константы:

– *match\_parent (fill\_parent)* – означает, что элемент займет всю доступную ему в родительском элементе ширину/высоту.

– *wrap\_content* – ширина/высота элемента будет определяться его содержимым.

**2) layout\_gravity**

Параметр layout\_gravity аналогичен выравниванию из Word или Excel и может принимать следующие значения:

– *top left (top right);*

*– bottom left (top right);*

*– center.*

**3) layout\_weight**

Если родитель содержит несколько элементов и необходимо, чтобы они заняли все пространство, можно использовать параметр layout\_weight – вес. Свободное пространство распределяется между элементами пропорционально их weight-значениям.

**4) layout\_margin**

Параметры margin (отступа) полностью аналогичны margin из html. Он может быть со всех сторон сразу, либо только с необходимых сторон. Для этого используются такие параметры, как *margin top*, *margin bottom*, *margin right* и *margin left.*

**Типы Layout**

Выделяют 5 основных типа лэйаутов:

**–** AbsoluteLayout;

**–** TableLayout;

**–** LinearLayout;

**–** FrameLayout;

**–** RelativeLayout;

**AbsoluteLayout** **–** означает, что каждый элемент верстки будет иметь абсолютную позицию относительно верхнего левого угла экрана задаваемую с помощью координат x и y. Т.е. верхний левый угол экрана при AbsoluteLayout имеет координаты x = 0, y = 0. Этим типом верстки пользоваться не рекомендуется.

**TableLayout –** разметка для расположения элементов в виде таблицы. Ряды задаются в xml с помощью тега TableRow, а ячейки в каждом ряду создаются автоматически для каждого элемента. Количество колонок в таблице будет равно максимальному количеству элементов в рядах. То есть если в одном ряду 3 элемента, а во втором 1, то колонок в TableLayout будет 3. Ширина колонки определяется по самому широкому элементу в ней. Ячейки таблицы можно оставлять пустыми, или объединять.

Атрибут android:layout\_column – задает в какой столбец поместить данный элемент (нумерация столбцов начинается с 0).

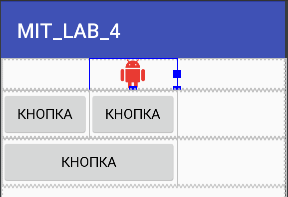
Атрибут android:layout\_span – позволяет объединить указанное количество столбцов (ряды объединить нельзя).

Например:

<TableLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<TableRow>

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/image"

android:layout\_column="1"

/>

</TableRow>

<TableRow>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Button"

/>

Рисунок 1.3 – Разметка с помощью TableLayout

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Button"

/>

</TableRow>

<TableRow>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

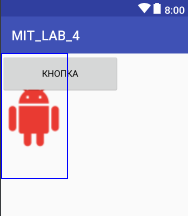
android:text="@string/Button"

android:layout\_span="2"

/>

</TableRow>

</TableLayout>

**FrameLayout** – разметка для отображения одного элемента. Дочерние View или ViewGroup в FrameLayout выравниваются по верхнему левому углу. Разметка может содержать несколько элементов, но тогда они будут перекрывать друг друга. Например:

<FrameLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<ImageView

android:layout\_width="74dp"

android:layout\_height="148dp"

android:src="@drawable/image"

/>

<Button

android:layout\_width="wrap\_content"

Рисунок 1.5 – Разметка с помощью FrameLayout

android:layout\_height="wrap\_content"

android:text="@string/Button"

/>

</FrameLayout>

FrameLayout удобно использовать для нескольких элементов, которые скрываются и показываются программно таким образом, что в каждый момент времени видимым оказывается только один из них. Для управления видимостью элемента используется атрибут android:visibility в xml или метод setVisibility (int visibility), где visibility принимает одно из трех значений:

VISIBLE — элемент видим

INVISIBLE — элемент невидим, но все еще занимает место в разметке

GONE — элемент невидим и не занимает место в разметке

**LinearLayout** – разметка для отображения одного или нескольких элементов в одну линию, горизонтально или вертикально.

Для выбора ориентации используется атрибут android:orientation с двумя возможными значениями «horizontal» и «vertical».

Ниже приведен фрагмент файла activity\_main.xml, в котором три кнопки позиционируются вертикально с помощью LinearLayout.

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical"

>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Button"

/>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

Рисунок 1.4 - Вертикальная разметка кнопок

android:text="@string/Button"

/>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

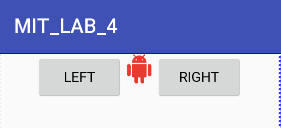
android:text="@string/Button"

/>

</LinearLayout>

**RelativeLayout –** разметка для расположения элементов относительно родителя или друг друга. Элементы начинают располагаться в указанном порядке, поэтому необходимо чтобы элемент был описан до того, как другой элемент будет на него ссылаться:

<RelativeLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<ImageView

android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/image"

android:layout\_centerHorizontal="true"

/>

<Button

Рисунок 1.6 – Разметка с помощью RelativeLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_toLeftOf="@id/image"

android:text="@string/Left"

/>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_toRightOf="@id/image"

android:text="@string/Right"

/>

</RelativeLayout>

Создавать компоновку для относительной разметки чуть сложнее, чем для линейной компоновки. Рассмотрим ещё один пример. Предположим, нам нужен такой экран (рисунок 1.7)

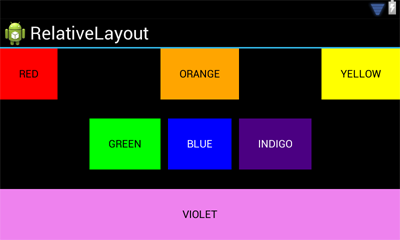


Рисунок 1.7 – Пример разметки в RelativeLayout

Шаг 1: в XML-файле задаем относительную компоновку:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

</RelativeLayout>

Шаг 2: определяем дочерние элементы. В нашем случае мы имеем семь текстовых меток разного цвета. Каждому элементу присваиваем необходимые свойства: текст, цвет, размер символов и так далее.

Шаг 3: определяем правила относительной разметки.

Красная метка должна находиться в правом верхнем углу родительского элемента.

Оранжевая метка должна располагаться по центру по горизонтали относительно родительского элемента.

Желтая метка выравнивается по правой части родительского элемента.

Зелёная метка выравнивается по центру (по вертикали) и выводится слева от синей метки.

Синяя метка выравнивается по центру (по вертикали и горизонтали) относительно родительского элемента, т.е. точно по центру.

Метка цвета индиго выравнивается по центру (по вертикали) и выводится справа от синей метки.

Фиолетовая метка выводится в нижней части родительского элемента и занимает всю её ширину.

Код разметки:

<?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" >

<TextView

android:id="@+id/TextView01"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="#f00"

android:gravity="center"

android:padding="25dp"

android:text="RED"

android:textColor="#000" >

</TextView>

<TextView

android:id="@+id/TextView02"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:background="#ffa500"

android:gravity="center"

android:padding="25dp"

android:text="ORANGE"

android:textColor="#000" >

</TextView>

<TextView

android:id="@+id/TextView03"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:background="#ffff00"

android:gravity="center"

android:padding="25dp"

android:text="YELLOW"

android:textColor="#000" >

</TextView>

<TextView

android:id="@+id/TextView04"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerVertical="true"

android:layout\_toLeftOf="@+id/TextView05"

android:background="#0f0"

android:gravity="center"

android:padding="25dp"

android:text="GREEN"

android:textColor="#000" >

</TextView>

<TextView

android:id="@+id/TextView05"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:layout\_margin="10dp"

android:background="#00f"

android:gravity="center"

android:padding="25dp"

android:text="BLUE"

android:textColor="#fff" >

</TextView>

<TextView

android:id="@+id/TextView06"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerVertical="true"

android:layout\_toRightOf="@+id/TextView05"

android:background="#4b0082"

android:gravity="center"

android:padding="25dp"

android:text="INDIGO"

android:textColor="#fff" >

</TextView>

<TextView

android:id="@+id/TextView07"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:background="#ee82ee"

android:gravity="center"

android:padding="25dp"

android:text="VIOLET"

android:textColor="#000" >

</TextView>

</RelativeLayout>

Во многих случаях грамотная относительная компоновка хорошо смотрится в альбомной и портретной ориентации.

**3. Постановка задачи**

3.1 Ознакомиться с теоретическими сведениями.

3.2 Разработать приложение в Android Studio в соответствии со своим вариантом задания (таблица 3.1).

3.3 Продемонстрировать результаты работы.

3.4 Сделать выводы по работе.

Таблица 3.1 – Варианты заданий

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Тип лэйаута** | **Эл-ты разметки** | **Макет** |
| 1 | TableLayout | Button,  ListView,  TextView, CheckBox |  |
| 2 | LinearLayout | Button,  ImageView, TextView |  |
| 3 | RelativeLayout | TextView,  Button,  ImageView,  CheckBox |  |
| 4 | FrameLayout | RadioButton, EditText,  Button,  TextView |  |
| 5 | TableLayout | ListView,  CheckBox, ImageView,  TextView |  |
| 6 | LinearLayout | Button,  TextView,  EditText,  ImageView |  |
| 7 | RelativeLayout | RadioButton, EditText,  TextView,  Button |  |
| 8 | FrameLayout | TextView,  Button,  CheckBox |  |
| 9 | TableLayout | ImageView, TextView,  Button |  |
| 10 | LinearLayout | ListView,  ImageView,  Button,  CheckBox |  |
| 11 | RelativeLayout | Button,  TextView, CheckBox |  |
| 12 | FrameLayout | ImageView, EditText,  Button,  TextView |  |

**4. Содержание отчета**

Отчет должен содержать:

– название работы;

– цель и вариант задания;

– листинг xml-кода;

– скриншот приложения;

– выводы по работе.

**5. Контрольные вопросы**

4.1 Что такое Activity и View?

4.2 Что такое Layout? Перечислите основные типы Layout и назовите их особенности.

4.3 Назовите основные свойства View-элементов.

4.4 Перечислите основные единицы измерения, используемые в Android Studio. В чем их отличия?