Модуль 2. Объектно-ориентированное программирование

Тема 2.5. Простейший интерфейс пользователя. Язык разметки XML

4 часа

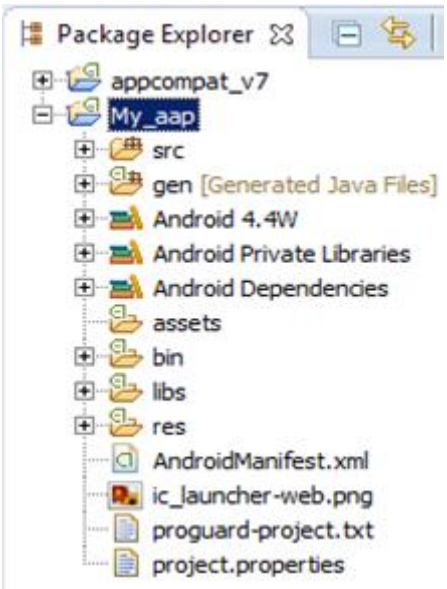
Оглавление

2.5. Построение интерфейса приложения

2.5.1. Структура проекта

На предыдущем уроке было создано наше первое Android-приложение. Давайте еще раз посмотрим его структуру. Итак, проект содержит папки:

- src — папка, в которой хранятся java-файлы, называемые файлами кода. Как правило, для каждого класса создается свой файл кода, но иногда разработчики располагают все классы в одном файле (не рекомендуем так делать, поскольку код становится неудобочитаемым);
- gen — папка, содержащая файл R.java для связи ресурсов с кодом. Данный файл создается системой при первой же компиляции проекта и не нуждается в редактировании, поскольку обновляется автоматически. В этом файле для каждого android-ресурса создается своя **целочисленная константа** (id), таким образом, мы получаем возможность из кода обращаться к ресурсу. Дело в том, что в Android-приложении написанном на java, код и ресурсы размещаются отдельно, что дает возможность независимого редактирования каждого из этих файлов;
- res — папка, в которой хранятся все ресурсы приложения и разметки экрана. Ресурсами являются текстовые константы, картинки, конфигурации экрана устройства (включая управляющие элементы), меню приложения, анимации, описание стилей объектов и подобные элементы. Все файлы ресурсов представляют собой гипертекстовые файлы xml.



Структура и настройки приложения описываются в файле AndroidManifest.xml, так же как и файлы ресурсов, написанном на языке гипертекстовой разметки xml.

Заметим, что IDE создает только «заготовки» файлов проекта, заполнять же их предстоит разработчику самостоятельно. Для этого необходимо знать язык программирования Java для кодирования приложения и язык разметки гипертекста XML для описания ресурсов и разметки экрана.

2.5.2. Язык разметки XML

Что же собой представляет язык разметки XML(англ. *Extensible Markup Language* - расширяемый язык разметки, описывающий класс объектов, называемых XML- документами) и как на нем описать ресурсы Android-приложения?

Гипертекст— термин, введённый Тедом Нельсоном в 1963 году для обозначения текста «ветвящегося или выполняющего действия по запросу».
Обычно гипертекст представляется набором текстов, имеющим свойства и содержащим узлы перехода между текстами, которые позволяют избирать читаемые сведения или последовательность чтения. Общеизвестным и ярко выраженным примером гипертекста служат веб-страницы— документы HTML (язык разметки гипертекста), размещённые в сети.
В компьютерной терминологии, гипертекст— текст, сформированный с помощью языка разметки, потенциально содержащий в себе гиперссылки.

Начнем с того, что XML – это язык свободного описания структур документов. То есть, если мы хотим, чтобы в документе присутствовал какой-либо элемент, то мы для него определяем некоторый тег (маркер в тексте). Например, для описания элемента «текстовая строка» можно условиться использовать тег <string></string>, где первая метка указывает начало описания элемента, а вторая (со знаком /) - конец описания. Между парой тегов помещается сам элемент. Для каждого элемента применяется своя пара тегов, при этом однотипные элементы описываются одинаковой парой тегов. Таким образом, для описания двух строк нам нужны две пары тегов:

<string>это первая строка</string> и

<string>это вторая строка</string>.

В открывающем теге можно поместить атрибуты описываемого элемента, такие как цвет, размер, начертание, выравнивание и т.п., то есть описать особенности формируемого элемента.

Атрибут — это свойство описываемого элемента. При этом у однотипных элементов полный набор атрибутов будет совпадать, но в описании можно использовать не все свойства. Каждому имени атрибута присваивается значение, записанное в виде текстовой строки, то есть заключенное в двойные кавычки. Разделяются свойства пробелом либо переносом строки.

Вернемся к нашему примеру: разметка

< string color = “red” align = “center”>это первая строка</string>

описывает текстовую строку, написанную красным шрифтом (начертание и размер установлены по умолчанию, поскольку эти свойства не указаны при описании) с выравниванием в центре страницы

это первая строка

Правила построения XML-документа

Каким бы свободным не был стиль XML-документа, все-таки существуют правила его формирования.

- В языке XML все теги парные. Это значит, что у каждого открывающего тега обязательно должен присутствовать закрывающий тег. Это правило позволяет описывать вложенные экземпляры, то есть помещать внутри одного элемента другие. Если тело тега пусто, то два тега записываются в один, который завершается косой чертой:

  < string color = “red” align = “center”/>.

- Документ должен содержать строку заголовка, в которой указывается версия языка и используемая текстовая кодировка:

<?xml version="1.0" encoding="utf8"?>

- Имена тегов должны начинаться с буквы или символа «_» с соблюдением регистра, поскольку XML различает регистры.
- Для реализации возможности одинаковых имен элементов для различающихся структур используют понятие пространства имен. Чтобы различать схемы документов, для каждой из них ставится в соответствие специальный уникальный идентификатор ресурса или URI. В результате схемы будут считаться тождественными только в том случае, если уникальные идентификаторы будут совпадать. В связи с этим в качестве идентификатора чаще всего используется адрес своего (возможно, несуществующего) ресурса. *Пространство имен* определяется благодаря *атрибуту xmlns* в начальном теге элемента:

< string xmlns:string="http://my_strings/styles/new" … > … </string>.

- В XML-тексте комментарии выделяются тегами <!-- и --> :

<!-- *текст, не читаемый анализатором документа* --> .

Конструкции языка

Содержимое XML-документа представляет собой набор элементов, секций CDATA, директив анализатора, комментариев, спец символов и текстовых данных.

Секции CDATA выделяют части документа, внутри которых текст не должен восприниматься как разметка. CDATA означает буквально "character data" — символьные данные.
Разделы CDATA представляют собой способ включения в документ текста, который иначе бы интерпретировался анализатором как разметка. Обычно такая потребность возникает для включения в текст документа примеров, содержащих разметку, или кода функций на каком-нибудь языке программирования.

Элементы данных

Элемент - это структурная единица XML-документа. Каждый документ содержит один или несколько элементов. Границы элементов представлены начальным и конечным тегами. Имя элемента в начальном и конечном тегах элемента должно совпадать. Элемент может быть также представлен пустым тегом, то есть не включающего в себя другие элементы и символьные данные.

Например, заключая число 167 в теги <school> и</school>, мы определяем непустой элемент, называемый <school>, содержимым которого является число 167. В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Любой непустой элемент должен состоять из начального, конечного тегов и данных, заключенных между ними. Например, следующие фрагменты будут являться элементами:

```
<school>167</school>
<city>Samara</city>
```

А вот эти элементами не являются:

```
<school>
<city>
Samara
```

Вложение элементов

Вложение— это размещение элементов внутри других элементов. Эти новые элементы называются дочерними элементами, а элементы, которые их окружают, — родительскими.

Типичная синтаксическая ошибка связана с вложенностью родительского и дочернего элементов. Каждый дочерний элемент должен быть целиком расположен между открывающим и замыкающим тегами своего родительского элемента. Дочерние элементы должны заканчиваться до начала следующего дочернего элемента.

Набором всех элементов, содержащихся в документе, задается его структура и определяются все иерархическое соотношения. Например, можно описывать месторасположение Самарских университетов. Указываем, что Самарские университеты расположены в городе Самара, который, в свою очередь, находится в России, используя для этого вложенность элементов XML:

```
<?xml version="1.0" encoding="utf-8"?>
<country>
        <name>Russia</name>
        <city>
                <name>Samara</name>
                <universities>
                <university>
                        <name>SSU</name>
                </university>
                <university>
                        <name>SSAU</name>
                </university>
                </universities>
        </city>
</country>
```

Производя в последствии поиск в таком документе, специальная программа будет опираться на информацию, заложенную в структуру документа - используя элементы документа. Т.е. если, например, требуется найти нужный университет в нужном городе, используя приведенный фрагмент, то необходимо просмотреть содержимое конкретного элемента <university>, находящегося внутри конкретного элемента <city>. Поиск при этом гораздо более эффективен, чем нахождение нужной последовательности по всему документу.

В XML документе определяется хотя бы один элемент, называемый корневым и с него программы-анализаторы начинают просмотр документа. В приведенном примере этим элементом является <country> .

В случае, если элемент не имеет содержимого, т.е. нет данных, которые он должен определять, он называется пустым.

Еще одним примером может быть описание кулинарного рецепта:

```
<?xml version="1.0" encoding="utf-8"?>
<recipe name="хлеб" preptime="5min" cooktime="180min">
        <title>          Простой хлеб</title>
        <composition>
                <ingredient amount="3" unit="стакан">Мука</ingredient>
                <ingredient amount="0.25" unit="грамм">Дрожжи</ingredient>
                <ingredient amount="1.5" unit="стакан">
                Тёплая вода
                </ingredient>
                <ingredient amount="1" unit="чайная ложка">Соль</ingredient>
        </composition>
        <instructions>
                <step>
                Смешать все ингредиенты и тщательно замесить.
                </step>
                <step>
                 Закрыть тканью и оставить на один час в тёплом
                помещении.
                </step>
                <step>
                 Замесить ещё раз, положить на противень и поставить
                в духовку.
                </step>
        </instructions>
</recipe>
```

Специальные символы

Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать специальный символьный, либо числовой идентификатор. Например:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

или десятичная форма записи:

```
&#34; - апостроф
```

Строковые обозначения спецсимволов могут определяться в XML документе при помощи компонентов (entity).

Сущности (entity) могут представлять собой фрагменты текста или специальные символы. Они могут указываться внутри документа или вне его. Во избежание ошибок и для правильности отображения сущности должны быть надлежащим образом объявлены и выражены.
В качестве сущностей можно определить фразы, такие как название компании, а затем использовать их по всему тексту. Чтобы создать сущность, назначьте ей имя и вставляйте это имя в текст после знака амперсанда (&), и заканчивая точкой с запятой — например, &coname; (или другое имя). Затем укажите этот код в строке DOCTYPE в квадратных скобках([]). Этот код определяет текст, который подставляется вместо сущности.

```
<!DOCTYPE MyDocs SYSTEM "filename.dtd" [ <!ENTITY coname
"RabidTurtleIndustries" ]>
```

Использование сущностей помогает избежать многократного повторения одной и той же фразы или информации. Оно может также облегчить редактирование текста (например, если компания изменит название) сразу во многих местах с помощью простой настройки строки определения сущности.

Структура XML-документа

Получив сведения о различных синтаксических конструкциях, можно рассмотреть общую логическую структуру документа. Любой XML-документ оформляется согласно следующей структуре:

пролог

тело документа

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="show-book.xsl"?>
<!DOCTYPE recipe SYSTEM "recipe.dtd">
<!-- recipe last update 2014-09-23 -->
<recipe name="хлеб" preptime="5min" cooktime="180min">
    <title>
        Простой хлеб
    </title>
    <composition>
        <ingredient amount="3" unit="стакан">Мука</ingredient>
        <ingredient amount="0.25" unit="грамм">Дрожжи</ingredient>
        <ingredient amount="1.5" unit="стакан">Тёплая вода</ingredient>
        <ingredient amount="1" unit="чайная ложка">Соль</ingredient>
    </composition>
    <instructions>
        <step>
            Смешать все ингредиенты и тщательно замесить.
        </step>
        <step>
            Закрыть тканью и оставить на один час в тёплом помещении.
        </step>
        <step>
            Замесить ещё раз, положить на противень и поставить в духовку.
        </step>
        <!-- <step> Почитать вчерашнюю газету. </step> - это сомнительный шаг... -->
    </instructions>
</recipe>
```

Т.е. состоит из трёх разделов:

- Пролог
- Тело документа
- Эпилог (на рисунке выше не представлен)

Пролог

Любой XML документ начинается с пролога. В пролог помещается описательная информация для всего документа в целом, которую требуется получить анализатору ещё до начала какой-нибудь обработки документа. К ней относятся:

- Команды обработки
- Декларация типа документа
- Комментарий

Пролог не является обязательной частью XML-документа, в том смысле, что его отсутствие не должно вызывать каких-либо проблем у анализаторов. Необязательность пролога продиктована лишь совместимостью с GML и HTML в версиях, существовавших до выхода XML. Однако, правилом хорошего тона считается всегда помещать пролог в любой XML-документ.

Тело документа

Тело документа включает в себя прикладную информацию и должно состоять ровно из одного узла, называемого корневым элементом. С точки зрения иерархической структуры данных, все другие прикладные данные обязаны быть дочерними этого одного единственного элемента документа.

Эпилог

Эпилог завершает XML-документ и не является обязательным, однако если он имеется в документе, то может состоять только из комментариев.

Упражнение 2.5.1

Рассмотрим пример иерархии плоских геометрических фигур:

Как подобную иерархию можно описать с помощью языка XML?

Первым шагом должен стать выбор родительского элемента, и в данном примере – это элемент «фигура». Для этого элемента придумываем название соответствующего ему тега, например, <figure>. тег с таким названием будет корневым элементом документа:

```
<?xml version="1.0" encoding="utf-8"?>
<figure>
 …
</figure>
```

По отношению к элементу «фигура» дочерними являются «открытая» и «закрытая». С точки зрения XML-документа это означает вложенность тегов. Пусть тег для элемента «открытая» называется <opened>, а для «закрытая» - <closed>. Тогда XML-документ выглядит следующим образом:

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

```
        …
    </opened>
    <closed>
        …
    </closed>
</figure>
```

Стоит обратить внимание, что несмотря на то, что элементов <opened></opened> и <closed></closed> с точки зрения самого XML-документа может быть неограниченное количество, с точки зрения логики, таких элементов может быть только по одному. И действительно, вряд ли в реальном мире можно отыскать открытые фигуры, чья «открытость» отличалась бы.

Рассмотрим теперь отдельно фигуры «открытые» и «закрытые».

В «открытые» фигуры попало две – точка (будем обозначать <point>) и ломанная (<polyline>). Про каждую из этих фигур можно сказать, что у них могут быть следующие свойства: толщина и цвет. Точке могут соответствовать координаты, а ломанной – длина.

Как точек, так и ломанных может быть сколь угодно много, и на первый взгляд их все можно просто разместить внутри элемента <opened>:

```
<?xml version="1.0" encoding="utf-8"?>
<figure>
    <opened>
        <point color=”red” thick=”2”>(0;0)</point>
        <point color=”blue” thick=”1”>(1;5,4)</point>
        <polyline color=”green” thick=”1”>23,1</polyline>
        <polyline color=”green” thick=”10”>1,09</polyline>
    </opened>
    <closed>
        …
    </closed>
</figure>
```

Однако, элементы, принадлежащие одному классу, принято группировать. Например:

```
<pointList>
    <point color=”red” thick=”2”>(0;0)</point>
    <point color=”blue” thick=”1”>(1;5,4)</point>
</pointList>
```

Тогда XML-документ приобретет вид:

```
<?xml version="1.0" encoding="utf-8"?>
<figure>
    <opened>
        <pointList>
        <point color=”red” thick=”2”>(0;0)</point>
        <point color=”blue” thick=”1”>(1;5,4)</point>
        </pointList>
        <polylineList>
```

```
                <polyline color=”green” thick=”1”>23,1</polyline>
                <polyline color=”green” thick=”10”>1,09</polyline>
                </polylineList>
        </opened>
        <closed>
                …
        </closed>
</figure>
```

Подобные рассуждения верны и для классов многоугольник и эллипс и их подклассов. Причем каждый конкретный представитель, например, класса эллипс, не обязан быть кругом. Это означает, что у элемента эллипс может не быть дочерних элементов.

```
<ellipseList>
        <ellipse filling=”red” border=”2” />
        <ellipse filling=”blue” border=”1”>5</ellipse>
        <ellipse filling=”green” border=”1”>
                <circleList>
                <circle>12,5</circle>
                <circle>0,152</circle>
                </circleList>
        </ellipse>
</ellipseList>
```

В результате XML-документ может иметь следующий вид:

```
<?xml version="1.0" encoding="utf-8"?>
<figure>
        <opened>
                <pointList>
                <point color=”red” thick=”2”>(0;0)</point>
                <point color=”blue” thick=”1”>(1;5,4)</point>
                </pointList>
                <polylineList>
                <polyline color=”green” thick=”1”>23,1</polyline>
                <polyline color=”green” thick=”10”>1,09</polyline>
                </polylineList>
        </opened>
        <closed>
                <ellipseList>
                <ellipse filling=”red” border=”2” />
                <ellipse filling=”blue”border=”1”>
                        фокальноерасстояние = 5
                </ellipse>
                <ellipse filling=”none” border=”1”>
                        <circleList>
                                <circle>радиус = 12,5</circle>
                                <circle>радиус = 0,152</circle>
                        </circleList>
                </ellipse>
                </ellipseList>
```

```
                <polygonList>
                <polygon angle="4">
                        <rectangleList type="square">
                                <rectangle measurement="sm">сторона =
                                        20</rectangle>
                                <rectangle measurement="m">сторона =
                                        0,45</rectangle>
                        </rectangleList>
                        <rectangleList type="rectangle">
                                <rectangle measurement="sm">площадь =
                                        17</rectangle>
                                <rectangle measurement="sm">площадь =
                                        5,7</rectangle>
                        </rectangleList>
                </polygon>
                <polygon angle="5" />
                </polygonList>
        </closed>
</figure>
```

2.5.3. Описание ресурсов Android с помощью XML

Ресурсы - один из основных компонентов, с которыми приходится работать. В Android принято держать объекты такие, как изображения, строковые константы, цвета, анимацию, стили и т.п. за пределами исходного кода. Система поддерживает хранение ресурсов во внешних файлах, например, xml. Внешние ресурсы легче поддерживать, обновлять и редактировать.

Каждое приложение на Android содержит каталог для ресурсов res/. Информация в каталоге ресурсов будет доступна в приложении через класс R, который автоматически генерируется средой разработки. То есть хранение файлов и данных в ресурсах (в каталоге res/) делает их легкодоступными для использования в коде программы.

Ресурсы в Android являются декларативными, т.е. они описывают свойства ресурсов, а не как их создавать. В основном ресурсы хранятся в виде XML-файлов в каталоге res с такими подкаталогами как, например, values, drawable-ldpi, drawable-mdpi, drawable-hdpi, layout.

Для удобства система создает идентификаторы ресурсов и использует их в файле R.java (класс R, который содержит ссылки на все ресурсы проекта), что позволяет ссылаться на ресурсы внутри кода программы. Статический класс R генерируется на основе заданных ресурсов и создается во время компиляции проекта. При создании класс содержит статические подклассы для всех типов ресурсов, для которых был описан хотя бы один экземпляр. Так как файл R генерируется автоматически, то нет смысла его редактировать вручную, потому что все изменения будут утеряны при повторной генерации.

В общем виде ресурсы представляют собой файл (например, изображение) или значение (например, заголовок программы), связанные с создаваемым приложением. Удобство использования ресурсов заключается в том, что их можно изменять без повторной компиляции

или новой разработки приложения. Имена файлов для ресурсов должны состоять исключительно из букв в нижнем регистре, чисел и символов подчеркивания.

Самыми распространенными ресурсами являются: разметка (layout), строки (string), цвета (color) и графические рисунки (bitmap, drawable).

В Android используются два подхода к процессу создания ресурсов - первый подход заключается в том, что ресурсы задаются в файле, при этом имя файла значения не имеет. Второй подход - ресурс задается в виде самого файла, и тогда имя файла уже имеет значение.

Общая структура каталогов, содержащих ресурсы выглядит следующим образом:

```
/res
        /drawable
                /*.png
                /*.jpg
                /*.gif
        /layout
                /*.xml
        /values
                /strings.xml
                /colors.xml
```

Чтобы создать xml файл ресурсов в Eclipse нужно проследовать по цепочке File > New > Other > Android и выбрать графу Android XML File.
В зависимости от того, какой элемент (папка/файл) проекта был в этот момент выделен, Eclipse может сам определить корневой элемент файла XML, либо предложить список подходящих вариантов.
В Android Studio цепочка действий выглядит так: File > New > Values resource file

2.5.4. Строковые ресурсы

Любой используемый в программе текст – это отдельный ресурс, такой же как изображение или звук.

При создании нового приложения, среда разработки (например, Eclipse) создает файл strings.xml, в котором хранятся строки для заголовка приложения и выводимого им (приложением) сообщения. Можно редактировать данный файл, добавляя новые строковые ресурсы. Также можно создавать новые файлы с любым (осмысленным) именем, которые будут содержать строковые ресурсы. Все эти файлы должны находиться в подкаталоге /res/values. Число файлов может быть любым. Строковые ресурсы обозначаются тегом <string>.

Типичный файл выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="app_name"> Hello world!</string>
        <string name="hello_world">Hello world!</string>
</resources>
```

где name – имя строкового ресурса, с помощью которого можно отличить один ресурс от другого, а также обратиться к данному ресурсу в программе или в другом файле xml.

Подход, при котором строковые ресурсы хранятся в отдельном файле, удобен по двум причинам:

1. Если одна и та же строка используется в нескольких частях программы, то благодаря использованию ссылки на текстовый ресурс в дальнейшем правку (если, например, захотели изменить текст какого-то заголовка) нужно будет делать только в одном исходном файле ресурса.
2. Создание отдельных файлов с текстовыми значениями удобно и для локализации приложения на несколько языков. Все, что нужно будет сделать — это создать отдельный файл с переводом, например, на французский язык, и создать для него отдельную директорию res/values-fr (fr – сокращение для French).

При именовании строковых ресурсов необходимо придерживаться нескольких правил:

- название строкового ресурса должно состоять из строчных букв;
- в случае, если название состоит из двух или трех слов, то их рекомендуется разделять нижним подчеркиванием;
- в названиях рекомендуется использовать только латиницу.

Во многих случаях можно задействовать системные строковые ресурсы - это строки типа OK, Cancel и др. В таких ситуациях используется схожий формат (добавляется слово android):

```
android:text="@android:string/cancel"
```

Здесь часть строки вида @android:string – обращение к системному строковому ресурсу android.

Вот список некоторых системных строковых ресурсов android:

```
@android:string/cancel
@android:string/no
@android:string/ok
@android:string/yes
```

Теперь строковые ресурсы можно использовать в xml файле разметки формы. Например, если открыть файл activity_main.xml, расположенный в res/layout/, то можно увидеть:

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
```

Среда разработки Eclipse автоматически создает код для Activity, в котором размещается текстовый виджет TextView. В качестве значения для текста помещена ссылка на текстовый ресурс:

```
android:text="@string/hello_world"
```

Сам ресурс находится в файле res/values/strings.xml:

```
<string name="hello_world">Hello world!</string>
```

В приложении можно применять жестко написанные строки кода. Но такой подход не рекомендуется - вместо жестко написанных строк следует использовать соответствующие идентификаторы, что позволяет изменять текст строкового ресурса, не изменяя исходного кода.

Пример жестко написанной строки:

```
android:text="нажми на меня"
```

Пример строки с использованием строкового ресурса:

```
android:text="@string/close"
```

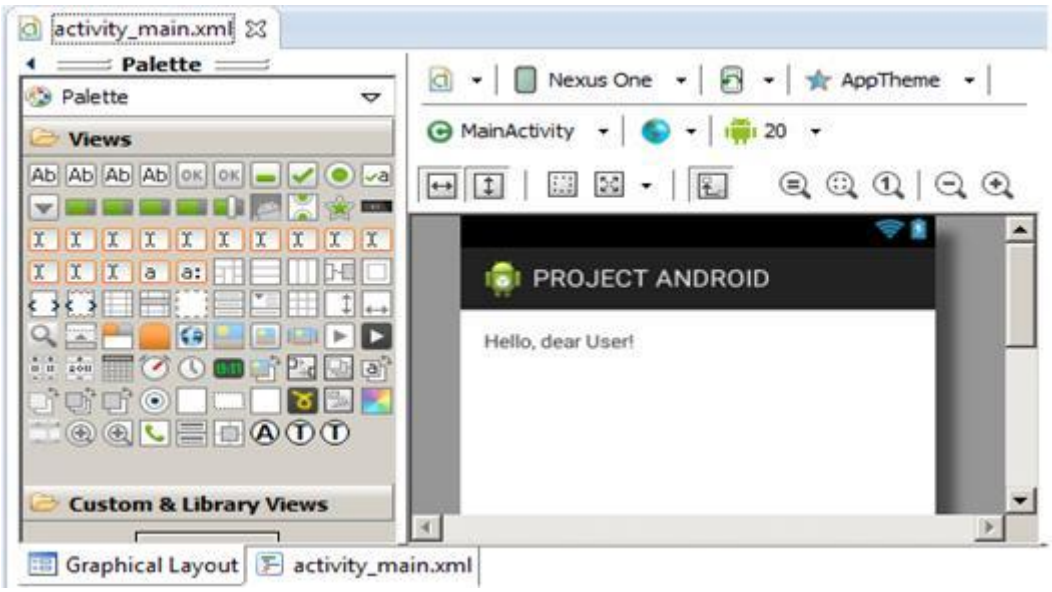
Применение ресурсов позволяет использовать механизм динамического выбора нужного ресурса в программе. Можно задать определенную структуру каталогов в проекте, чтобы создавать ресурсы для конкретных языков, регионов и аппаратных конфигураций. Во время выполнения программы Android выберет нужные значения для конкретного телефона пользователя.

Когда IDE Eclipse открывает Android XML файлы, внизу можно заметить несколько вкладок (их количество и названия зависят от самого XML файла). Так, например, для файла strings.xml их две: Resources и strings.xml. Если открыть вторую, то можно увидеть текст, содержащийся в файле. Если нужно отредактировать файл в таком представлении, то придется вручную печать всё необходимое. Первая же вкладка (Resources) показывает ресурсы, расположенные в файле, в виде объектов. В таком представлении ресурсы гораздо удобнее добавлять, удалять и редактировать.
IDE Eclipse позволяет быстро создавать строковые ресурсы. Для этого нужно щелкнуть правой кнопкой мыши по строке и выбрать Quick Fix.

В Android Studio вкладок нет и сразу видно весь текст со всеми тегами. Быстрое создание строковых ресурсов в данной IDE осуществляется по щелчку правой кнопки мыши и выбору пункта Generate, или сочетанием клавиш Alt + Insert

2.5.5. Интерфейс пользователя. Разметка (layout)

Графический интерфейс создается с помощью представлений (View) и групп представлений (ViewGroup). Эти элементы размещаются на активности, их описания помещаются в файл манифеста, а действия с объектами прописываются программно в файле кода MainActivity.java в виде методов классов, наследуемых от классов View и ViewGroup или атрибутивно в файле разметки layout/activity_main.xml. У файла разметки так же имеется графический вид Graphical layout - системная имитация мобильного устройства.



В окне Palette можно выбрать вид представления объектов: значки, названия, названия и значки. Выберите удобный для себя вид для работы с графическими представлениями.

Eclipse при создании нашего проекта разместил в активности единственный объект — относительную разметку RelativeLayout с дочерним элементом - текстовой строкой TextView, значение которой находится в файле ресурсов res/value/strings.xml в текстовой константе string.

Мы уже знаем, как изменять значения таких констант. А как же изменить параметры самого представления? Во-первых, это можно сделать в графическом представлении, во-вторых - редактируя xml-файл разметки. Графический редактор очень прост в использовании, поэтому мы не будем останавливаться на его рассмотрении, а рассмотрим содержимое текстового файла.

Переключимся на вкладку activity_main.xml и посмотрим свойства объектов, которые присутствуют в нашей активности.

```
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns: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="com.example.my_aap.MainActivity" >

    <TextView
           android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/hello" />
</RelativeLayout>
```

Layout — это разметка окна. Разметка может быть одного из четырех типов:

- LinearLayout — линейная разметка, при которой виджеты (Views) размещаются вертикально или горизонтально. Расположение объектов в данной разметке осуществляется строго друг под другом в один столбец, если ориентация разметки

вертикальная (android:orientation=”vertical”) или в одну строку рядом, если ориентация горизонтальная (android:orientation=”horizontal”). Линейная разметка позволяет объектам пропорционально заполнять пространство (android:layout_weight).

- FrameLayout — разметка в виде фреймов, когда все объекты выравниваются по левому верхнему углу экрана. Очень неудобна в качестве корневой, но можно применять как вложенную для размещения в ней одного объекта и резервирования свободного пространства рядом с ним.
- TableLayout — табулированная разметка, позволяющая располагать представления по строкам и столбцам. Каждый столбец таблицы в xml-файле ресурсов выделяется парным тегом <TableRow></TableRow>. В данной разметке во всех строках выделяется одинаковое количество столбцов. Если в строках было описано разное количество объектов, то длина строки измеряется по максимальному количеству элементов в TableRow. В строках меньшей длины крайние правые ячейки остаются свободными. Для того, чтобы оставить пустой не крайнюю правую ячейку, в нее помещают пустой объект, например, TextView без текста и форматирования.
- RelativeLayout — относительная разметка, при которой позиции объектов можно определять относительно родительского виджета, а так же относительно друг друга. Данный вид разметки используется по умолчанию.

С Android 4.0 в проектах появился новый вид разметки - GridLayout - табличная разметка, в которой элементы фиксируются в определенных ячейках таблицы, независимо от заполненности других ячеек. Все дочерние элементы в этой разметке имеют размеры по размеру данных, по этому один объект может занимать несколько ячеек. Это свойство регулируется атрибутами android:layout_columnSpan и android:layout_rowSpan.

Для всех видов разметок обязательно имеются атрибуты высоты и ширины, при этом каждая разметка имеет свой набор атрибутов.

Разметка является корневым элементом активности. У окна может быть ровно один корневой элемент, в котором размещаются дочерние объекты. Потомком может быть представление любого типа, включая и тип родительского объекта.

Свойства описываются внутри парных тегов представления. Атрибут описывается пространством имен ”android:” и названием свойства с присвоением ему значения.

```
android:property = “property_value”
```

Описание свойств можно прочитать, выделив соответственную строку в ниспадающем списке, который появляется при нажатии комбинации клавиш <Ctrl> + <Пробел> в процессе написания имени атрибута.

В составе Android SDK имеется утилита Hierarchy Viewer, позволяющая отследить разметку интерфейса пользователя. Запустить утилиту можно после запуска приложения на устройстве. Запустите файл tool/hierarchyviewer.bat, находящийся в папке установки Android SDK и после его запуска выберите устройство, затем <Focused Window>, и нажмите кнопку Load View Hierarchy.

Часто используемые атрибуты объектов приведены в таблице

атрибут	описание	возможные значения
android:id	маркировка объекта по имени	*“@+id/name”* Знак + говорит о добавлении нового id объекта.
android:orientation	выравнивание разметки на экране	vertical, horizontal
android:layout_width	ширина объекта	fill_parent (match_parent) - родительский размер; wrap_content- по размеру содержимого; или фиксированный размер с указанием единиц измерения
android:layout_height	высота объекта	fill_parent (match_parent) - родительский размер; wrap_content- по размеру содержимого; или фиксированный размер с указанием единиц измерения
android:layout_weight	весовой коэффициент объекта	по умолчанию 0, или принимает натуральные значения - доля объекта в линии разметки
android:gravity android:layout_gravity	выравнивание содержимого потомков или самих дочерних объектов	*center, left, right*

Более подробно о видах разметок и тонкостей работы с ними можно почитать в цикле статей:

- *Android. Всё о LinearLayout - 1*
- *Android. Всё о LinearLayout - 2*
- *Android. Всё о LinearLayout - 3*
- *Android. Обзор RelativeLayout*
- *Android. Обзор TableLayout*
- *Android. Обзор FrameLayout*

Упражнение 2.5.1.

Изучение атрибутов разметки Layout

1. Измените относительную разметку на линейную, внутри нее создайте еще две разметки по ширине родительской разметки и по высоте относящиеся 1:2 сверху вниз. Нижнюю разметку разделите на две равные вертикальные. Самым нижним элементом корневой разметки должно остаться текстовое поле. Установите для всех объектов различные значения цветового атрибута. Изучите возможности остальных типов разметок, изменяя корневую и дочерние разметки экрана.
2. Используя табличную разметку, нарисуйте экран игры «Пятнашки».

2.5.6. Представления (Виджеты, Views)

Разметка является основой для расположения других полей (виджетов) приложения. В xml-файле разметки описание объекта помещается в парные теги <Тип_объекта> </Тип_объекта>, а в java-файле необходимо импортировать стандартный класс из пакета `android.widget.Тип_объекта`, а затем создать константу или переменную этого типа.
В программном коде доступ к объекту осуществляется посредством его id, установленного атрибутом android:id, поэтому данный атрибут необходимо устанавливать всем создаваемым представлениям, к которым планируется обращение. У EditText имеется возможность установить подсказки (hint) о вводимой информации. Для этого необходимо добавить атрибут android:hint = “текст подсказки”.

В файле разметки:

```
android:id = “@+id/name”
android:hint = “@string/text_hint”
```

В java-файле:

```
Type_object var_name = (Type_object)findViewById(R.id.name);
```

Здесь инструкция findViewById() находит виджет по его id, считывая адрес из класса R.

Класс R, описание которого хранится в файле Проект/gen/Имя_пакета/R.java, создается в процессе компиляции. В этом классе создаются константы для каждого вновь созданного (в разметке или в программном коде) объекта. При каждом запуске проекта данный класс создается снова. Для очистки класса R достаточно выполнить команду меню Project|Clean… При следующей компиляции файл R.java снова будет заполнен константами.

Текстовые объекты

Различают два вида текстовых полей: текстовая константа (TextView) и редактируемый текст (EditText).

TextView предназначается для отображения на экране стационарного фрагмента текста. Если содержимое не помещается в размеры объекта, то автоматически появляется полоса прокрутки. После создания в программе экземпляра текстового класса, здесь же можно установить его атрибуты. Например, программным аналогом атрибута файла разметки android:text = “значение” является вызов метода setText(string):

```
android:text = “Hello!” <=> txt.setText (“Hello!”).
```

Мы видим, что для отображения текста в Textview в файле разметки используется атрибут android:text, например:

```
android:text="Погладь кота, ...!"
```

Как уже упоминалось, такой подход является нежелательным. Вместо этого лучше создать строковый ресурс в файле strings.xml:

```
<string name="cat">Погладь кота, ...!!!</string>
```

А затем уже использовать этот строковый ресурс:

```
android:text="@string/cat"
```

У элемента TextView есть многочисленные методы и XML атрибуты для работы с текстом. Например, основные XML атрибуты, отображающие свойства элемента TextView:

1. android:textSize— размер текста. При установке размера текста используются несколько единиц измерения:

- px — пикселы;
- dp — независимые от плотности пикселы. Это абстрактная единица измерения, основанная на физической плотности экрана;
- sp — независимые от масштабирования пикселы, т.е. зависимы от пользовательских настроек размеров шрифтов;
- in — дюймы, базируются на физических размерах экрана;
- pt — 1/72 дюйма, базируются на физических размерах экрана;
- mm — миллиметры, также базируются на физических размерах экрана.

Обычно при установке размера текста используются единицы измерения sp, которые наиболее корректно отображают шрифты.

```
android:textSize="48sp"
```

2. android: textStyle — стиль текста. Используются константы:

- normal;
- bold;
- italic.

Пример установки стиля через атрибуты:

```
android:textStyle="bold"
```

3. android:textColor— цвет текста. Используются четыре формата в шестнадцатеричной кодировке:

- #rgb
- #argb
- #rrggbb
- #aarrggbb

где r, g, b — соответствующий цвет, a — прозрачность (alpha channel). Значение a, установленное в 0, означает прозрачность 100%. Значение по умолчанию, без указания значения alpha, — непрозрачно.

Например, можно создать цветовой ресурс в файле colors.xml:

```
<color name="text">#87CEFA</color>
```

И тогда в атрибуте указываем:

```
android:textColor="@color/text"
```

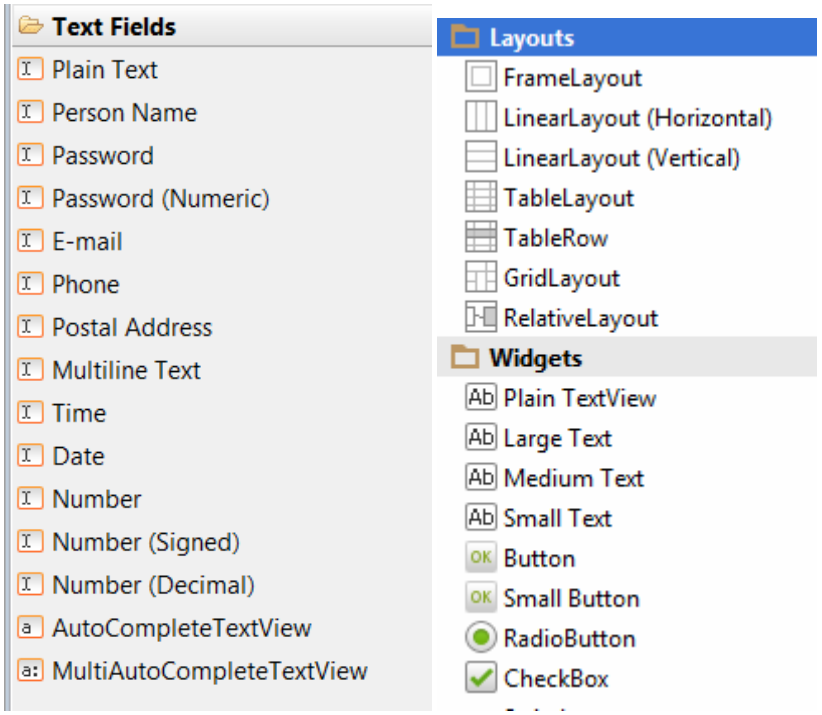
Если же мы хотим реализовать ввод и редактирование текста, нам придется использовать объект класса EditText, основным методом которого является получение вводимого значения — метод getText(). Данный метод возвращает значение типа Editable - надстройка над String, обладающая возможностью динамического изменения в процессе выполнения программы. Приведем описание текстовых полей: простой текст с надписью “Hello!” и текстовое поле для ввода с подсказкой (hint) “Input your text”.

```
<TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentTop="true"
      android:layout_marginTop="31dp"
      android:text="Hello!" />

<EditText
      android:id="@+id/etext1"
     android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/textView1"
      android:layout_below="@+id/textView1"
      android:hint="Input your text" >
</EditText>
```

Элемент интерфейса EditText — это текстовое поле для ввода пользовательского текста, которое используется, если необходимо редактирование текста. EditText является наследником TextView.

Как и для файлов ресурсов (например, strings.xml), так и для файлов разметки (activity_main.xml), когда открываем его в Eclipse, внизу можно заметить несколько вкладок: GraphicalLayout и activity_main.xml.
В AndroidStudio вкладки называются Design и Text. Открыв первую, на панели инструментов слева в папке TextFields можно найти текстовые поля под разными именами. Например, PlainText, Password, E-mail и т.д.



Eclipse **AndroidStudio**

Для быстрой и удобной разработки текстовые поля снабдили различными свойствами и дали разные имена. Рассмотрим некоторые из них.

PlainText

PlainText- самый простой вариант текстового поля. При добавлении в разметку его XML-представление будет следующим:

```
<EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

Password

При использовании типа Password в свойстве inputType используется значение textPassword. При вводе текста сначала показывается символ, который заменяется на звёздочку или точку.

```
<EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />
```

E-mail

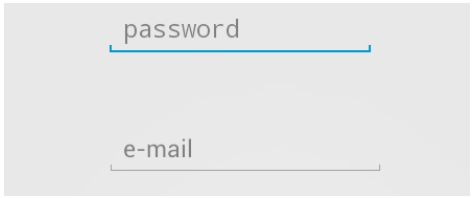
У элемента E-mail используется атрибут android:inputType="textEmailAddress". В этом случае на клавиатуре появляется дополнительная клавиша с символом @, который обязательно используется в любом электронном адресе.

Текст-подсказка

Подсказка видна, если текстовый элемент не содержит пользовательского текста. Как только пользователь начинает вводить текст, то подсказка исчезает. Соответственно, если удалить пользовательский текст, то подсказка появляется снова. Это очень удобное решение во многих случаях, когда на экране мало места для элементов.

В Android у многих элементов есть свойство Hint (атрибут hint), который работает аналогичным образом. Установите у данного свойства нужный текст и у вас появится текстовое поле с подсказкой.

```
android:hint="подсказка"
```



Кнопки

Кнопка (Button) - это управляющий элемент. Кнопки позволяют управлять процессом выполнения приложения в ходе его работы.

Кнопка описывается тегами <Button></Button> и является наследником класса android.widget.Button. Сам по себе объект Button ничего не делает, то есть при нажатии на кнопку ничего не происходит. Чтобы действия выполнялись, необходим обработчик события нажатия кнопки. Методом, устанавливающим обработчик событий для кнопки, является метод setOnClickLictener().

Создадим относительную разметку экрана приложения и поместим в нее следующие текстовые поля и кнопки:

```
<TextView
android:id="@+id/txt1"
android:layout_width="wrap_content”
android:layout_height="wrap_content”
android:text=“Текст" />
<EditText
android:layout_width="fill_parent"
android:layout_height="wrap_content”
android:id="@+id/etxt1“
android:layout_below="@id/txt1"
android:hint="Вводите текст сюда" />
<Button
android:layout_width="wrap_content“
android:layout_height="wrap_content“
android:id="@+id/bt1“
android:layout_below="@id/etxt1“
android:text="Ввести" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/bt2"
android:layout_below="@id/etxt1"
android:layout_toRightOf="@id/bt1"
android:text="Отменить" />
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/bt3"
android:layout_below="@id/bt1"
android:text="Выход" />
```

Откроем файл java-кода (если вы его не переименовали, то по умолчанию это MainActivity.java)

и создадим объекты для виджетов:

```
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;

public class MainActivity extends Activity{
    TextView txt;
    EditText etxt;
    Button bt1;
    Button bt2;
    Button bt3;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SetContentView(R.layout.main_activity);
        txt = (TextView)findViewById(R.id.txt);
        etxt = (EditText)findViewById(R.id.etxt);
        bt1 = (Button)findViewById(R.id.bt1);
        bt2 = (Button)findViewById(R.id.bt2);
        bt3 = (Button)findViewById(R.id.bt3);
```

А теперь напишем обработчики событий для каждой из кнопок (продолжаем работать в методе onCreate()):

```
import android.view.View.OnClickListener;//импортируем класс обработчика
...
OnClickListener l1 = new OnClickListener(){//первый объект-обработчик
@Override //переопределяем родительский метод по своему усмотрению
public void onClick(View v){
            txt.setText(etxt.getText());}//этот обработчик копирует текст // из
EditText в TextView
};
OnClickListener l2 = new OnClickListener(){//второй объект-обработчик
@Override
public void onClick(View v){
            txt.setText("ничего не напишу!");}//этот обработчик меняет
// содержимое TextView на константу "ничего не напишу!"
};
OnClickListener l3 = new OnClickListener(){//третий объект-обработчик
@Override
public void onClick(View v){//этот обработчик останавливает работу Activity
            finish();}
};
}
//и вызовем каждый обработчик к определенной кнопке
bt1.setOnClickListener(l1);
bt2.setOnClickListener(l2);
bt3.setOnClickListener(l3);
```

Из кода видно, что мы создали два текстовых поля, три кнопки и три обработчика, то есть всего 8 объектов. И это для такого простого приложения. Чем сложнее приложение, тем больше объектов, соответственно, больше занятой оперативной памяти (сравнительно небольшой у мобильных устройств). По этому необходимо стараться уменьшить количество создаваемых объектов. Так, мы можем объединить все три обработчика (l1, l2, l3) в один, используя оператор выбора при его определении:

```
OnClickListener l = new OnClickListener(){//создаем один обработчик
@Override
public void onClick(View v){
    switch(v.getId()){
        case R.id.bt1: txt.setText(etxt.getText()); break;
        case R.id.bt2: txt.setText("ничего не напишу!"); break;
        case R.id.bt3: finish(); break;}
    }
};//и вызываенм его к каждой кнопке
bt1.setOnClickListener(l);
bt2.setOnClickListener(l);
bt3.setOnClickListener(l);
```

Выполняемое действие зависит от того, какой id у нажатой кнопки. Для идентификации кнопки применяется метод getId(), считывающий атрибут из класса R.

Ну и самый простой способ вызова обработчика - это вызов его из атрибутов кнопки в xml - файле. Для этого нам не нужно создавать объект-обработчик, а достаточно просто определить метод onClick() для различных случаев (назовем методы l):

```
В файле разметки activity_main.xml:
...
<Button
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content“
android:id="@+id/bt1“
android:onClick="l1“
android:layout_below="@id/etxt1“
android:text="Ввести" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/bt2"
android:onClick="l2“
android:layout_below="@id/etxt1"
android:layout_toRightOf="@id/bt1"
android:text="Отменить" />
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/bt3"
android:onClick="l3“
android:layout_below="@id/bt1"
android:text="Выход" />

В файле кода MainActivity.java
...
public void l1(View v){
txt.setText(etxt.getText());
}
public void l2(View v){
txt.setText("нет текста");
}
public void l3(View v){
finish();
}
```

Обратите внимание на то, что в данном случае для обработчиков никаких объектов не создается, мы обходимся только самими виджетами. Такой же экономный способ написания обработчиков событий нажатия кнопок получается, если в качестве обработчика использовать Activity. Эту возможность мы рассмотрим далее при знакомстве с Интерфейсами.

Если же при описании кнопки мы применим теги <ImageButton></ImageButton>, то на кнопке вместо текста будет изображение, например, иконка приложения, находящаяся в папке ресурсов drawable-* (в зависимости от параметров экрана будет выбрана одна из папок). В файле разметки установка изображения на кнопку осуществляется командой

```
android:src = “@drawable/имя_файла_с_картинкой”
```

в программе – методом setImageResource(R.drawable.ic_launcher.png), где ic_launcher.png — имя файла с картинкой из папки drawable-*.

```
<ImageButton
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/abc_menu_dropdown_panel_holo_dark"  />
```

Переключатели/включатели

RadioButton. Представления этого типа позволяют выбирать одну из предложенных позиций и относятся к java-классу RadioGroup. Группа переключателей в файле разметки выделяется тегами <RadioGroup></RadioGroup>, каждый из переключателей описывается тегами <RadioButton></RadioButton>. В группе активным может быть только один переключатель. Если теги <RadioGroup> не ставить, то кнопки будут независимы друг от друга. Так делать не стоит, поскольку для независимых переключателей предусмотрен класс CheckBox. Атрибуты переключателей ничем не отличаются от атрибутов остальных виджетов. Определение группы переключателей из двух кнопок выглядит так:

```
<RadioGroup
       android:id="@+id/radioGroup1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content">
 <RadioButton
       android:id="@+id/radio1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:checked="true"
       android:text="RadioButton 1"/>
 <RadioButton
       android:id="@+id/radio2"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="RadioButton 2"/>
</RadioGroup>
```

Для программирования поведения приложения при выборе определенной кнопки применяется метод `isChecked()`. При создании в приложении текстового поля с `id = textView1` и кнопки `button1` можно задать событие на вывод текстовой строки в поле *textView1* при выборе переключателя и нажатия кнопки `button1`.

```
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.TextView;
public class MainActivity extends ActionBarActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       final RadioButton r1 = (RadioButton)findViewById(R.id.radio1);
       final RadioButton r2 = (RadioButton)findViewById(R.id.radio2);
       final Button b = (Button)findViewById(R.id.button1);
        final TextView t = (TextView)findViewById(R.id.textView1);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (r1.isChecked()) t.setText("Case 1");
                if (r2.isChecked()) t.setText("Case 2"); }
       });
    }
```

```
}
```

CheckBox. Об этом классе объектов уже упоминалось ранее. Класс CheckBox не привязывает свои объекты к какой-либо группе, то есть представления работают независимо. Для описания в разметке используются теги <CheckBox></CheckBox> . Проверка включения кнопки проверяется тем же методом isChecked(), что и у RadioButton.

Другие представления. В активность приложения можно добавлять шкалу прогресса (ProgressBar), часы (AnalioClock, DigitalClock), календарь (Calendar), секундомер (Hronometer) и другие объекты. Такие объекты проще добавлять в графическом представлении разметки. Ознакомиться с полным списком виджетов можно на сайте разработчика http://developer.android.com/reference/android/.

Упражнение 2.5.2

Добавление и редактирование объектов в активности

Создайте активность со следующими элементами:

- полями ввода логина и пароля: они должны задаваться в коде в виде констант и содержать подсказку (hint);

- кнопкой «Вход»;

- текстовым полем, отображающим верно ли введен пароль: если верно, то вывести зеленым цветом «Верно», если не верно - красным «Вы ошиблись в логине или пароле», при этом поля ввода очищаются.

Задание 2.5.1

Создание простого интерфейса пользователя.

Создайте приложение “Анкета”, содержащее относительную разметку. Разместите внутри разметки следующие виджеты:

1. Слева в столбец расположите TextView с пунктами анкеты:
   - Фамилия
   - Имя
   - Отчество
   - Год рождения
   - Город
   - Школа
   - Опыт программирования
   - Знание языков программирования
2. В центре экрана для каждого поля установите виджеты для ввода данных:
   - EditText для полей “Фамилия”, “Имя”, “Отчество”, “Год рождения”.
   - RadioButton для полей “Город” и “Опыт программирования”.
   - CheckBox для поля “Знание языков программирования”.
3. В правой части экрана расположите календарь и часы.
4. В конце анкеты добавьте кнопки “Отправить”, “Отмена” и “Выход”.

Задание 2.5.2

Создать Activity с следующими элементами:

- полями ввода логина и пароля: они должны задаваться в коде в виде констант и содержать подсказку (hint);

- кнопкой «Вход».

- текстовым полем, отображающим верно ли введен пароль: если верно вывести зеленым цветом «Верно», если не верно красным «Вы ошиблись в логине или пароле», при этом поля ввода очищаются.

Для текстового поля использовать метод visibly.

Задание 2.5.3*

Добавление и редактирование переключателей.

Создайте приложение с экраном тестовой системы, содержащим два вопроса: один вопрос с выбором ровно одного ответа из четырех предложенных, второй — с множественным выбором, то есть среди приведенных ответов есть более одного правильного. Предусмотрите кнопку ввода ответа. Результат проверки правильности ответа выводите текстовой строкой, расположенной сразу же под вариантами ответа: зеленого цвета строка «верно!» при выборе правильного ответа или красного цвета строка «не верно!», если ответ был ошибочным.

Увеличьте количество вопросов. Доработайте приложение так, чтобы вопросы выводились по очереди. Разместите на активности еще две кнопки: одну для продолжения теста, вторую для его завершения.

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателям IT ШКОЛЫ SAMSUNG Покровской Валерии Павловне и Дружинской Елене Владимировне.