КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»

Факультет «Информатика и управление"

Кафедра "Программное обеспечение ЭВМ, информационные технологии"

Меню и виджеты графического интерфейса

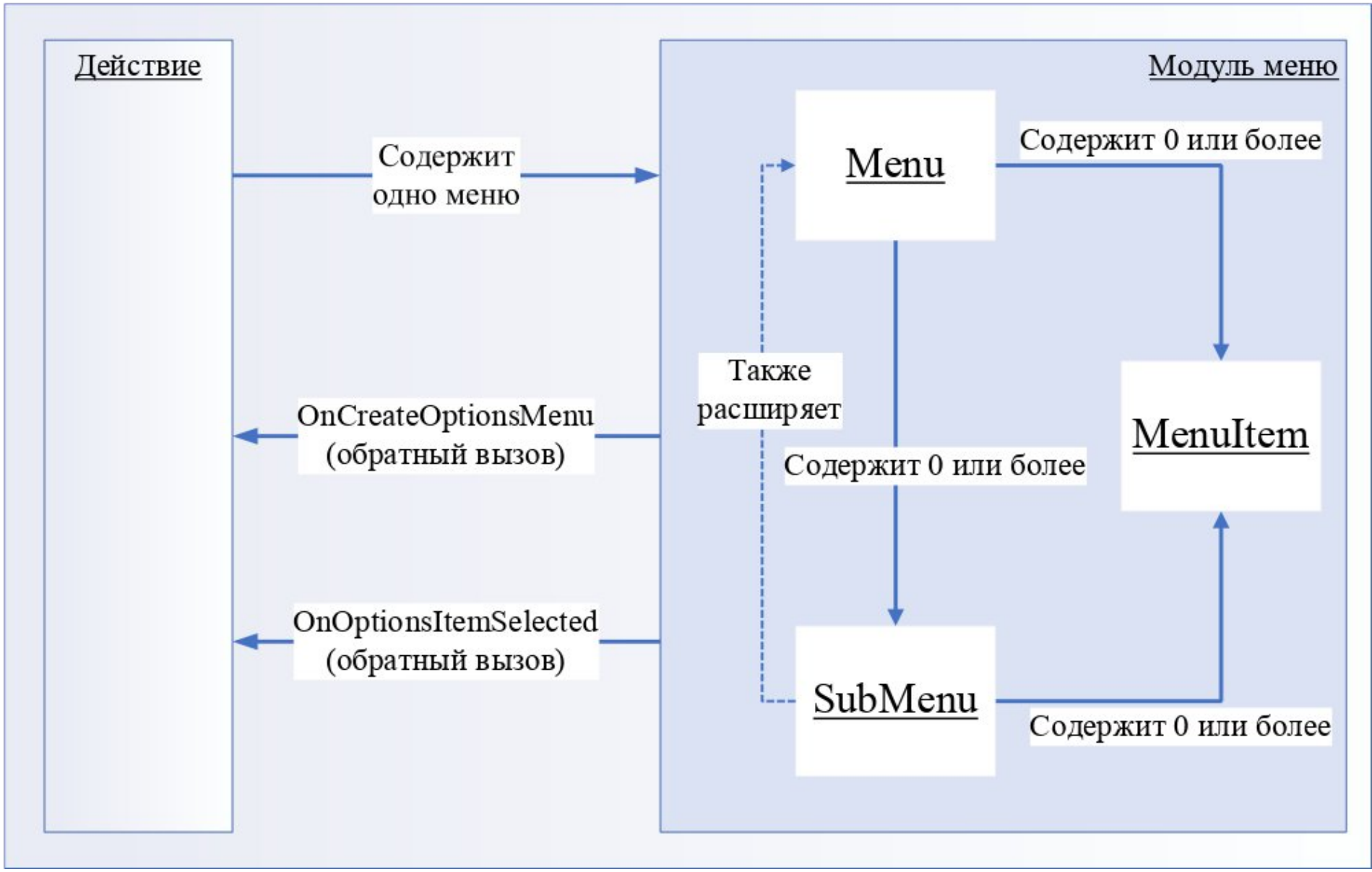
Калуга

Меню в Android представляют собой **объекты Java**, однако они представляются и **как ресурсы**. А поскольку меню — это ресурсы, Android SDK позволяет **загружать их из XML-файлов**, как и все другие ресурсы, генерируя идентификаторы ресурсов для каждого из загруженных пунктов меню.

Основой поддержки меню в Android является класс **android.view.Menu**. Каждое действие в Android связано с одним объектом меню, который содержит **ряд пунктов и подменю**.

Пункты меню представлены классом **android.view.Menultem**, а подменю — классом **android.view.SubMenu**. Эти взаимосвязи графически изображены на рисунке. На этом рисунке представлена структурная диаграмма для визуализации взаимосвязей между различными классами и функциями, которые имеют дело с меню. На рис. видно, что объект Menu содержит набор пунктов меню.

У пункта меню есть имя (название), **идентификатор** пункта меню, **порядок** сортировки (в SDK называется просто "order" — упорядоченность) и идентификатор (или номер). Эти идентификаторы позволяют задать порядок пунктов в меню. Например, если у одного пункта меню порядок сортировки равен 4, а у другого — 6, то первый пункт будет находиться в меню выше второго.


Действие
Модуль меню
Содержит
одно меню
Menu
Содержит 0 или более
Также
расширяет
MenuItem
OnCreateOptionsMenu
(обратный вызов)
Содержит 0 или более
OnOptionsItemSelected
(обратный вызов)
SubMenu
Содержит 0 или более

Пункты меню можно объединять в группы, назначая им одинаковый идентификатор группы, который является атрибутом объекта пункта меню. Несколько пунктов меню с одинаковым идентификатором группы считаются входящими в одну группу. На рис. показаны **два метода обратного вызова**, которые позволяют создавать пункты меню и реагировать на их выбор: **onCreateOptionsMenu** и **onOptionsItemSelected**.

Операционная система Android предлагает **три вида меню**:
OptionsMenu — меню выбора опций, появляется внизу экрана при нажатии кнопки Menu на мобильном устройстве;
ContextMenu — контекстное меню, появляется при долгом касании (2 или более секунды) сенсорного экрана;
SubMenu — подменю, привязывается к конкретному пункту меню (меню выбора опций или контекстному меню). Пункты подменю не поддерживают вложенного меню.

В случае с OptionsMenu существует два типа меню:
IconMenu — меню со значками, добавляет значки к тексту в пункты меню. Это единственный тип меню, поддерживающий значки.
ExpandedMenu — Для старых версий Android, расширенное меню появляется автоматически, если количество пунктов меню больше шести. В поздних версиях, в случае, если все пункты меню не помещаются на экран, появляется полоса прокрутки.

Меню выбора опций

Начнем с создания самого часто использующегося меню — меню выбора опций, которое появляется, когда пользователь нажмет кнопку **Menu** на мобильном устройстве.

Меню можно создать в файле разметки или с помощью метода add(). Мы будем использовать второй способ, оставив файл разметки без изменений.

Первым делом нужно определить идентификаторы создаваемых пунктов меню, делается это так:

```
//Java
public static final int IDM_NEW = 101;
public static final int IDM_OPEN = 102;
public static final int IDM_SAVE = 103;
public static final int IDM_EXIT = 104;
//Kotlin
val IDM_NEW = 101
val IDM_OPEN = 102
val IDM_SAVE = 103
val IDM_EXIT = 104
```

В нашем меню будут четыре пункта с идентификаторами IDM_NEW, IDM_OPEN, IDM_SAVE, IDM_EXIT.

Далее для каждого пункта меню нужно вызвать метод add():
В данном случае мы не только добавляем новый пункт меню, но и устанавливаем для него клавишу быстрого доступа. Если клавиша быстрого доступа не нужна, то оператор добавления пункта меню можно записать короче:

```
menu.add(Menu.NONE, IDM_NEW, Menu.NONE, "New game");
```

```
//Java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Вызов базового класса для включения системных меню
super.onCreateOptionsMenu(menu);
menu.add(0 // Группа
          , 1 // Идентификатор элемента
          , 0 // Порядок
          , "append"); // Название
menu.add(0, 2, 1, "item2");
menu.add(0, 3, 2, "clear");
// Для видимости меню важно возвратить true
return true;
}
```

```
//Kotlin
override fun onCreateOptionsMenu(menu: Menu):
Boolean {
    // Вызов базового класса для включения системных меню
    super.onCreateOptionsMenu(menu)
    menu.add(0 // Группа
         , 1 // Идентификатор элемента
         , 0 // Порядок
         , "append") // Название
        menu.add(0, 2, 1, "item2")
    menu.add(0, 3, 2, "clear")
    // Для видимости меню важно возвратить true
    return true
}
```

Методу add() нужно передать **четыре параметра**:

идентификатор группы меню — позволяет связать данный пункт меню с группой других пунктов этого меню. Используется для создания сложных меню. В нашем случае можно воспользоваться значением Menu.NONE, потому что идентификатор группы задавать не нужно;

идентификатор меню — позволяет однозначно идентифицировать пункт меню, далее идентификатор поможет определить, какой пункт меню был выбран пользователем;

порядок расположения пункта в меню — по умолчанию пункты размещаются в меню в порядке их добавления методом add(), поэтому в качестве значения этого параметра можно тоже указать Menu.NONE;

заголовок — задает заголовок пункта меню, видимый пользователем. Вы можете задать как текстовую константу, так и указать строковый ресурс.

```
menu.add(Menu.NONE, IDM_NEW, Menu.NONE, "New game");
```

```java
package com.example.test65;
//Java
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
import android.widget.Toast;
import android.view.Gravity;
public class MainActivity extends Activity {
    // Описываем идентификаторы пунктов меню
    public static final int IDM_NEW = 101;
    public static final int IDM_OPEN = 102;
    public static final int IDM_SAVE = 103;
    public static final int IDM_EXIT = 104;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
            // Создаем пункты меню
      menu.add(Menu.NONE, IDM_NEW, Menu.NONE, "New
game");
      menu.add(Menu.NONE, IDM_OPEN, Menu.NONE, "Open
game");
      menu.add(Menu.NONE, IDM_SAVE, Menu.NONE, "Save
game");
      menu.add(Menu.NONE, IDM_EXIT, Menu.NONE,
"Exit");
      return(super.onCreateOptionsMenu(menu));
    }
```

```kotlin
//Kotlin
import android.os.Bundle
import android.view.Menu
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    val IDM_NEW = 101
    val IDM_OPEN = 102
    val IDM_SAVE = 103
    val IDM_EXIT = 104
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
    override fun onCreateOptionsMenu(menu:
Menu): Boolean {
        // Создаем пункты меню
        menu.add(Menu.NONE, IDM_NEW, Menu.NONE,
"New game")
        menu.add(Menu.NONE, IDM_OPEN,
Menu.NONE, "Open game")
        menu.add(Menu.NONE, IDM_SAVE,
Menu.NONE, "Save game")
        menu.add(Menu.NONE, IDM_EXIT,
Menu.NONE, "Exit")
        return super.onCreateOptionsMenu(menu)
    }
}
```

```java
//Java
@Override
    public boolean onOptionsItemSelected(MenuItem
item) {
    CharSequence t;
    switch (item.getItemId()) {
    case IDM_NEW: t = "New game selected"; break;
    case IDM_OPEN: t = "Open game selected";
break;
    case IDM_SAVE: t = "Save game selected";
break;
    case IDM_EXIT: t = "Exit selected"; break;
    default: return false;
    }
    Toast toast = Toast.makeText(this, t,
Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    // Выводим уведомление
    toast.show();
    return true;
    }
```


```kotlin
//Kotlin
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val t: CharSequence
    t = when (item.getItemId()) {
        IDM_NEW -> "New game selected"
        IDM_OPEN -> "Open game selected"
        IDM_SAVE -> "Save game selected"
        IDM_EXIT -> "Exit selected"
        else -> return false
    }
    val toast = Toast.makeText(this, t, Toast.LENGTH_LONG)
    toast.setGravity(Gravity.CENTER, 0, 0)     // Выводим
уведомление
    toast.show()
    return true
}
```

Работа с контекстными меню

Пользователи, работающие на настольных компьютерах, без сомнения, знакомы с контекстными меню. Например, в Windows-приложениях контекстные меню открываются в результате щелчка правой кнопкой мыши на элементе пользовательского интерфейса. Android поддерживает такую же идею с помощью **операции *длинного щелчка* (long click).** Длинный щелчок — это щелчок кнопкой мыши на любом представлении Android, который длится несколько дольше, чем обычно.

В карманных устройствах, таких как мобильные телефоны, щелчки кнопкой мыши имитируются целым рядом способов, в зависимости от механизма навигации. **Независимо от реализации щелчка кнопкой мыши на конкретном устройстве, более долгое удерживание щелчка считается длинным щелчком.**

Контекстные меню структурно отличается от стандартных меню, и для них характерны некоторые нюансы, не присущие меню выбора. Как и Menu, класс ContextMenu может содержать ряд пунктов меню. Для добавления пунктов в контекстное меню используется тот же набор методов, что и в классе Menu.

Наибольшее различие между Menu и ContextMenu состоит во владельце меню. **Владельцами обычных меню выбора являются действия, а владельцами контекстных меню являются представления (View-объекты).** Это понятно, т.к. длинные щелчки, активизирующие контекстные меню, применяются к *представлению,* на котором выполнен щелчок. Поэтому действие может иметь только одно меню выбора, но несколько контекстных меню: ведь действие может содержать несколько представлений, а каждое представление может иметь собственное контекстное меню, и поэтому действие может иметь столько контекстных меню, сколько в нем представлений.

Несмотря на то что владельцем контекстного меню является представление, **метод для заполнения контекстных меню находится в классе Activity**. Этот метод называется activity.onCreateContextMenu (), и его функции аналогичны методу activity.onCreateOptionsMenu (). Этот метод обратного вызова также несет информацию о представлении, для которого нужно заполнить пункты контекстного меню.

С контекстными меню связан еще один существенный момент. Метод onCreateOptionsMenu () автоматически вызывается для каждого действия, но в отношении метода onCreateContextMenu () это не так. Представление в действии *не обязано* иметь контекстное меню.

Например, в действии может быть три представления, но программист может активизировать контекстное меню лишь для одного из них. **Если понадобится, чтобы у какого-либо представления было контекстное меню, это представление необходимо зарегистрировать вместе с его действием специально для обретения контекстного меню. Для этого предназначен метод**

activity.registerForContextMenu (view).

Создать контекстное меню чуть сложнее, чем обычное.

В начале необходимо подключить дополнительные Java-пакеты. С появлением контекстного меню список импортируемых пакетов увеличился

```
import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import
android.view.ContextMenu.ContextMenuInfo;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.Toast;
```

Помимо пакетов, непосредственно относящихся к контекстному меню, необходимо добавить пакеты, относящиеся к линейной разметке (к ней мы будем привязывать наше меню), и пакет android.view.View — необходимый при создании меню.

Далее необходимо определить идентификаторы контекстного меню:

```
//Java
public static final int IDM_RESTORE = 201;
public static final int IDM_PAUSE = 202;
```

```
//Kotlin
companion object {
    const val IDM_RESTORE = 201
    const val IDM_PAUSE = 202
}
```

Чтобы не путать элементы контекстного меню с элементами меню опций, идентификаторы начинаются с двойки.

Далее создаем само контекстное меню:

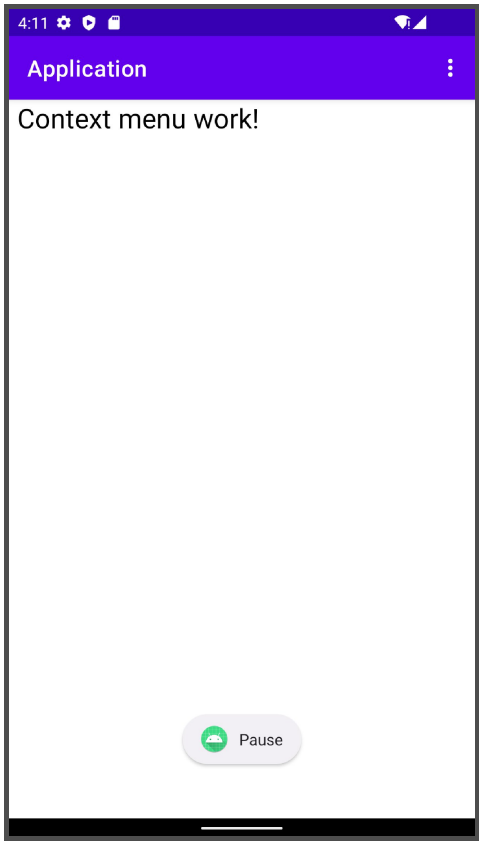
```
//Java
@Override
    public void onCreateContextMenu(ContextMenu menu,
View v, ContextMenuInfo info) {
    super.onCreateContextMenu(menu, v, info);
    menu.add(Menu.NONE, IDM_PAUSE, Menu.NONE, "Pause game");
    menu.add(Menu.NONE, IDM_RESTORE, Menu.NONE, "Restore game");
    }

//Kotlin
override fun onCreateContextMenu(menu: ContextMenu, v: View?, info:
ContextMenu.ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, info)
    menu.add(Menu.NONE, IDM_PAUSE, Menu.NONE, "Pause game")
    menu.add(Menu.NONE, IDM_RESTORE, Menu.NONE, "Restore game")
}
```

Тип View требует пакет android.view.View, поэтому мы его и импортировали. Далее создаем обработчик выбора пункта меню:

```
//Java
@Override
    public boolean onContextItemSelected(MenuItem item) {
    CharSequence t;
    t = "";
    switch (item.getItemId()) {
    case IDM_PAUSE: t = "Pause"; break;
    case IDM_RESTORE: t = "Restore"; break;
    default: super.onContextItemSelected(item);
    }
    Toast toast = Toast.makeText(this, t, Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
    return true;
    }
   //Kotlin
   override fun onContextItemSelected(item: MenuItem): Boolean {
       var t: CharSequence

       t = ""
       when (item.itemId) {
           IDM_PAUSE -> t = "Pause"
           IDM_RESTORE -> t = "Restore"
           else -> super.onContextItemSelected(item)
       }
       val toast = Toast.makeText(this, t, Toast.LENGTH_LONG)
       toast.setGravity(Gravity.CENTER, 0, 0)
       toast.show()
       return true
   }
```

Если сейчас запустить приложение, то меню ни при каких условиях не появится — оно не привязано к какому-либо виджету. Для привязки контекстного меню к нашей разметке используются операторы:

```
//Java
final LinearLayout game = (LinearLayout)findViewById(R.id.Main);
registerForContextMenu(game);
//Kotlin
val game: LinearLayout = findViewById(R.id.Main) as LinearLayout
registerForContextMenu(game)
```



```java
//Java
package com.example.test65;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.SubMenu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
import android.widget.Toast;
import android.view.Gravity;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.LinearLayout;
import android.view.View;
public class MainActivity extends Activity {
public static final int IDM_RESTORE = 201;
public static final int IDM_PAUSE = 202;
    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final LinearLayout game =
(LinearLayout)findViewById(R.id.Main);
        registerForContextMenu(game);
    }
}
```

```kotlin
//Kotlin
import android.os.Bundle
import android.view.*
import android.widget.LinearLayout
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    public override fun
onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val game =
findViewById<View>(R.id.Main) as LinearLayout
        registerForContextMenu(game)
    }
    companion object {
        const val IDM_RESTORE = 201
        const val IDM_PAUSE = 202
    }
}
```

```java
//Java
@Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo info) {
    super.onCreateContextMenu(menu, v, info);
    menu.add(Menu.NONE, IDM_PAUSE, Menu.NONE, "Pause game");
    menu.add(Menu.NONE, IDM_RESTORE, Menu.NONE, "Restore game");
    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {
    CharSequence t;
    t = "";
    switch (item.getItemId()) {
    case IDM_PAUSE: t = "Pause"; break;
    case IDM_RESTORE: t = "Restore"; break;
    default: super.onContextItemSelected(item);
    }
    Toast toast = Toast.makeText(this, t, Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
    return true;
    }
```

```xml
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/Main"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

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

</LinearLayout>
```

```kotlin
//Kotlin
override fun onCreateContextMenu(menu: ContextMenu, v: View?,
info: ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, info)
    menu.add(Menu.NONE, IDM_PAUSE, Menu.NONE, "Pause game")
    menu.add(Menu.NONE, IDM_RESTORE, Menu.NONE, "Restore game")
}
override fun onContextItemSelected(item: MenuItem): Boolean {
    var t: CharSequence
    t = ""
    when (item.getItemId()) {
        IDM_PAUSE -> t = "Pause"
        IDM_RESTORE -> t = "Restore"
        else -> super.onContextItemSelected(item)
    }
    val toast = Toast.makeText(this, t, Toast.LENGTH_LONG)
    toast.setGravity(Gravity.CENTER, 0, 0)
    toast.show()
    return true
}
```

Контекстное меню для нескольких View-объектов

```
//Java
    TextView tvColor, tvSize;
      @Override
      public void onCreate(Bundle
    savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            tvColor = (TextView)
    findViewById(R.id.tvColor);
            tvSize = (TextView)
    findViewById(R.id.tvSize);
            // для tvColor и tvSize необходимо
    создавать контекстное меню
            registerForContextMenu(tvColor);
            registerForContextMenu(tvSize);
      }
```

```
//Kotlin
override fun onCreate(savedInstanceState:
Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    tvColor = findViewById(R.id.tvColor) as
TextView
    tvSize = findViewById(R.id.tvSize) as
TextView
    registerForContextMenu(tvColor)
    registerForContextMenu(tvSize)
}
```

```
//Java
final int MENU_COLOR_RED = 1;
final int MENU_COLOR_GREEN = 2;
final int MENU_COLOR_BLUE = 3;
final int MENU_SIZE_22 = 4;
final int MENU_SIZE_26 = 5;
final int MENU_SIZE_30 = 6;
```

```
//Kotlin
val MENU_COLOR_RED = 1
val MENU_COLOR_GREEN = 2
val MENU_COLOR_BLUE = 3
val MENU_SIZE_22 = 4
val MENU_SIZE_26 = 5
val MENU_SIZE_30 = 6
```

```java
//Java
@Override
public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo
menuInfo) {
  switch (v.getId()) {
case R.id.tvColor:
  menu.add(0, MENU_COLOR_RED, 0, "Red");
  menu.add(0, MENU_COLOR_GREEN, 0, "Green");
  menu.add(0, MENU_COLOR_BLUE, 0, "Blue");
  break;
case R.id.tvSize:
  menu.add(0, MENU_SIZE_22, 0, "22");
  menu.add(0, MENU_SIZE_26, 0, "26");
  menu.add(0, MENU_SIZE_30, 0, "30");
  break;
}
}
```

```kotlin
//Kotlin
override fun onCreateContextMenu(menu:ContextMenu, v:View, menuInfo:ContextMenuInfo) {
    when(v.getId()) {
        R.id.tvColor->{
            menu.add(0, MENU_COLOR_RED, 0, "Red")
            menu.add(0, MENU_COLOR_GREEN, 0, "Green")
            menu.add(0, MENU_COLOR_BLUE, 0, "Blue")
        }
        R.id.tvSize->{
            menu.add(0, MENU_SIZE_22, 0, "22")
            menu.add(0, MENU_SIZE_26, 0, "26")
            menu.add(0, MENU_SIZE_30, 0, "30")
        }
    }
}
```

Подменю

Подменю можно добавить в любое другое меню, кроме контекстного. Подменю полезно при создании очень сложных приложений, где пользователю доступно много функций.

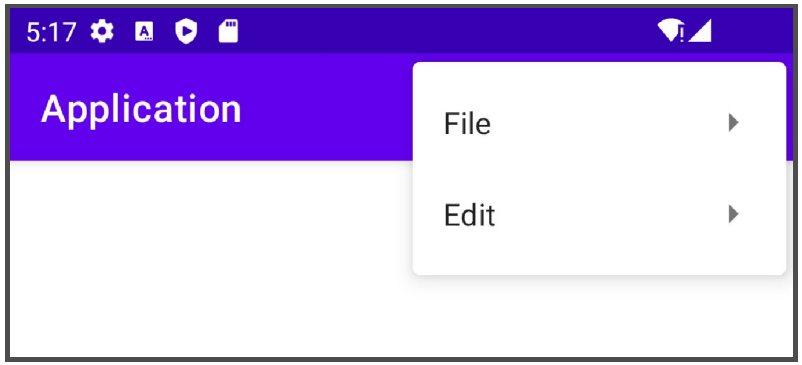
Рассмотрим, как можно добавить подменю в меню опций. Для создания подменю используется метод **addSubMenu()**. Создадим два подменю в меню опций - file и edit - с соответствующими пунктами подменю:

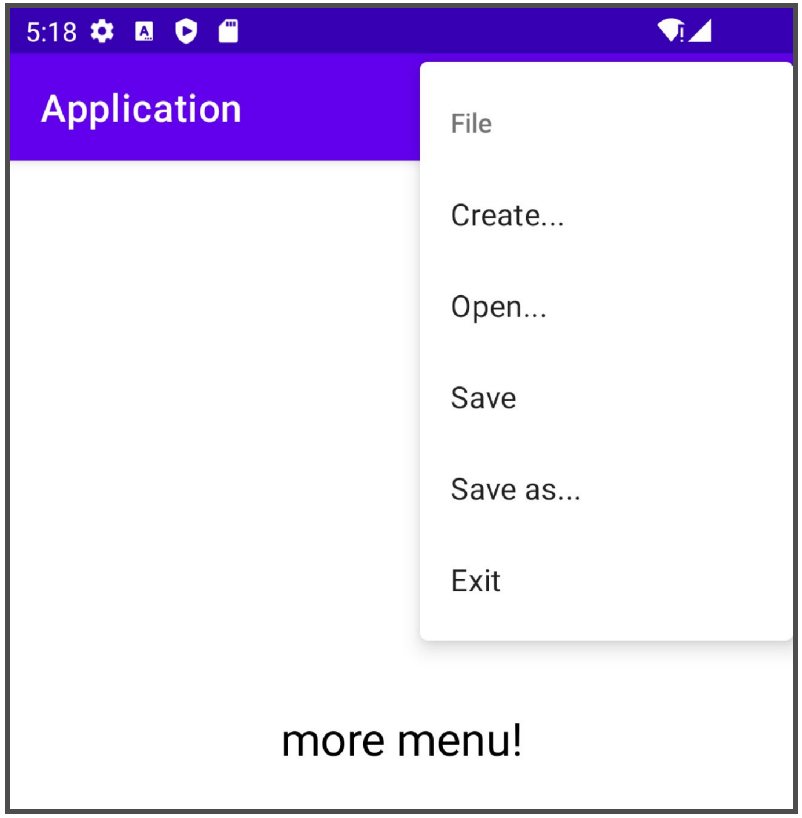
```
//Java
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
    // Создаем пункты меню
    SubMenu subFile=menu.addSubMenu("File");
    subFile.add(Menu.NONE, IDM_NEW, Menu.NONE, "Create...");
    subFile.add(Menu.NONE, IDM_OPEN, Menu.NONE, "Open...");
    subFile.add(Menu.NONE, IDM_SAVE, Menu.NONE, "Save");
    subFile.add(Menu.NONE, IDM_SAVEAS, Menu.NONE, "Save as...");
    subFile.add(Menu.NONE, IDM_EXIT, Menu.NONE, "Exit");
    SubMenu subEdit = menu.addSubMenu("Edit");
    subEdit.add(Menu.NONE, IDM_COPY, Menu.NONE, "Copy");
    subEdit.add(Menu.NONE, IDM_CUT, Menu.NONE, "Cut");
    subEdit.add(Menu.NONE, IDM_PASTE, Menu.NONE, "Paste");
    return(super.onCreateOptionsMenu(menu));
}           menu.add(Menu.NONE, IDM_HELP, Menu.NONE, "Help");
```

Обработка команд подменю осуществляется аналогично обработке команд меню опций.

```kotlin
//Kotlin
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    // Создаем пункты меню
    val subFile = menu.addSubMenu("File")
    subFile.add(Menu.NONE, IDM_NEW, Menu.NONE, "Create...")
    subFile.add(Menu.NONE, IDM_OPEN, Menu.NONE, "Open...")
    subFile.add(Menu.NONE, IDM_SAVE, Menu.NONE, "Save")
    subFile.add(Menu.NONE, IDM_SAVEAS, Menu.NONE, "Save
as...")
    subFile.add(Menu.NONE, IDM_EXIT, Menu.NONE, "Exit")
    val subEdit = menu.addSubMenu("Edit")
    subEdit.add(Menu.NONE, IDM_COPY, Menu.NONE, "Copy")
    subEdit.add(Menu.NONE, IDM_CUT, Menu.NONE, "Cut")
    subEdit.add(Menu.NONE, IDM_PASTE, Menu.NONE, "Paste")
    return super.onCreateOptionsMenu(menu)
}
```

```java
//Java
package com.example.test65;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.SubMenu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
import android.widget.Toast;
import android.view.Gravity;
import android.widget.LinearLayout;
import android.view.View;

public class MainActivity extends Activity {
// Описываем идентификаторы пунктов меню
public static final int IDM_NEW = 101;
public static final int IDM_OPEN = 102;
public static final int IDM_SAVE = 103;
public static final int IDM_EXIT = 104;
public static final int IDM_HELP = 105;
public static final int IDM_ABOUT = 106;
public static final int IDM_UPDATE = 107;

public static final int IDM_SAVEAS = 108;
public static final int IDM_COPY = 109;
public static final int IDM_CUT = 110;
public static final int IDM_PASTE = 111;

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





Загрузка меню из XML-файлов

До сих пор мы создавали все наши меню программным образом. Конечно, это утомительное занятие, т.к. для каждого меню приходится указывать несколько идентификаторов и определять константы для каждого такого идентификатора.
Но вместо этого можно определять меню с помощью XML-файлов — в Android меню также являются ресурсами. Такой подход к созданию меню имеет несколько преимуществ: возможность именования меню, автоматическое их упорядочение и присваивание идентификаторов. Кроме того, для текста меню поддерживается локализация.
Для работы с XML-меню нужно выполнить следующие шаги.
1. Определите XML-файл с дескрипторами меню.
2 Поместить этот файл в подкаталог /res/menu. Имя файла может быть произвольным, и файлов может быть сколько угодно. Android автоматически генерирует идентификатор ресурса для такого файла меню.
3. Воспользуйтесь идентификатором ресурса для файла меню, чтобы загрузить XML-файл в меню.
4. Описать реагирование на пункты меню, используя идентификаторы ресурсов, сгенерированные для каждого пункта.
Далее рассмотрим каждый из этих шагов и соответствующие фрагменты кода.

Структура ресурсного XML-файла меню

Вначале рассмотрим пример XML-файла с определениями меню. Все файлы меню начинаются с одного и того же высокоуровневого **дескриптора menu**, за которым следуют **несколько дескрипторов group** Каждый из этих дескрипторов group соответствует группе пунктов меню, о которых шла речь в начале. Идентификаторы групп можно указывать с помощью ***@+id.*** В каждой группе имеется ряд пунктов меню со своими идентификаторами пунктов, связанными с символьными именами. Все возможные аргументы этих XML-дескрипторов можно посмотреть в документации по Android SDK.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <!-- В этой группе используется стандартная категория -->
    <group android:id="@+id/menuGroup_Main" >
        <item
            android:id="@+id/menu_testPick"
            android:orderInCategory="5"
            android:title="Test Pick"/>
        <item
            android:id="@+id/menu_testGetContent"
            android:orderInCategory="5"
            android:title="Test Get Content"/>
        <item
            android:id="@+id/menu_clear"
            android:orderInCategory="10"
            android:title="clear"/>
        <item
            android:id="@+id/menu_dial"
            android:orderInCategory="7"
            android:title="dial"/>
        <item
            android:id="@+id/menu_test"
            android:orderInCategory="4"
            android:title="@+string/test"/>
        <item
            android:id="@+id/menu_show_browser"
            android:orderInCategory="5"
            android:title="show browser"/>
    </group>

</menu>
```

Реагирование на пункты меню, созданных из XML

Реагирование на пункты XML-меню выполняется так же, как и на программно созданные меню — однако есть небольшое отличие. Как и раньше, щелчки на пунктах меню обрабатываются в методе обратного вызова onOptionsItemSelected.

Android генерирует не только идентификатор ресурса для XML-файла, но и необходимые идентификаторы пунктов меню для их отличия друг от друга. При реагировании на пункты меню это удобно, т.к. отпадает необходимость в явном создании идентификаторов для пунктов меню и управлении ими.
Более того, при работе с XML-меню нет необходимости и в определении констант для этих идентификаторов, и не нужно заботиться об их уникальности, поскольку все это обеспечивается при генерации идентификаторов ресурсов.

```
//Java
@Override
    private void onOptionsItemSelected(MenuItem item) {
    this.appendMenuItemText(item);
    if    (item.getItemId()    == R.id.menu_clear){//this.emptyText();
    }
    else if    (item.getItemId()    == R.id.menu_dial) {// какая-то обработка
    }
    else if    (item.getItemId()    == R. id.menu_testPick) {// какая-то обработка
    }
    else if    (item.getItemId()    == R.id.menu_testGetContent) {// какая-то обработка
    }
    else if    (item.getItemId()    == R.id.menu_show_browser) {
    // какая-то обработка
    }
}
```

```
//Kotlin
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    this.appendMenuItemText(item)
    if (item.getItemId() == R.id.menu_clear) {//this.emptyText()
    } else if (item.getItemId() == R.id.menu_dial) {
        // какая-то обработка
    } else if (item.getItemId() == R.id.menu_testPick) {
        // какая-то обработка
    } else if (item.getItemId() == R.id.menu_testGetContent) {
        // какая-то обработка
    } else if (item.getItemId() == R.id.menu_show_browser) {
        // какая-то обработка
    }
}
```

Обратите внимание, что у имен пунктов меню из ресурсного XML-файла имеются автоматически сгенерированные идентификаторы пунктов в пространстве R.id.

Уведомления

Приложения могут отображать два типа уведомлений: краткие всплывающие сообщения (**Toast Notification**) и постоянные напоминания (**Status Bar Notification**). **Первые** отображаются на экране мобильного устройства какое-то время и не требуют внимания пользователя. Как правило, это не критические информационные уведомления. **Вторые** постоянно отображаются в строке состояния и требуют реакции пользователя.

Например, приложение требует подключения к вашему серверу. Если соединение успешно установлено, можно отобразить краткое уведомление, а вот если подключиться не получилось, тогда отображается постоянное уведомление, чтобы пользователь сразу мог понять, почему приложение не работает.

Чтобы отобразить всплывающее сообщение, используйте класс **Toast** и его методы **makeText** (создает текст уведомления) и **Show** (отображает уведомление):

```
//Java
Context context = getApplicationContext();
Toast toast = Toast.makeText(context, "This is notification",
Toast.LENGTH_SHORT);

//Kotlin
val context: Context = applicationContext
val toast = Toast.makeText(context, "This is notification", Toast.LENGTH_SHORT)
```

Первый параметр метода makeText — это контекст приложения, который можно получить с помощью вызова getApplicationContext().
Второй параметр — текст уведомления.
Третий — задает продолжительность отображения уведомления:
LENGTH_SHORT — небольшая продолжительность (1-2 секунды) отображения текстового уведомления;
LENGTH_LONG — показывает уведомление в течение более длительного периода времени (примерно 4 секунды).

Анимация

Рассмотрим следующие трансформации с обычными View-компонентами:

- менять прозрачность
- менять размер
- перемещать
- поворачивать

Трансформации конфигурируются в XML файлах, затем в коде программы считываются и присваиваются View-элементам.

В нашем проекте есть папка res. Надо в ней создать папку anim. Сделать это можно, например, так: правой кнопкой на res и в меню выбираем New -> Folder. В папке anim надо создать файлы. Делается это аналогично: правой кнопкой на anim и в меню выбираем New -> File. В этих файлах будем конфигурировать анимацию.

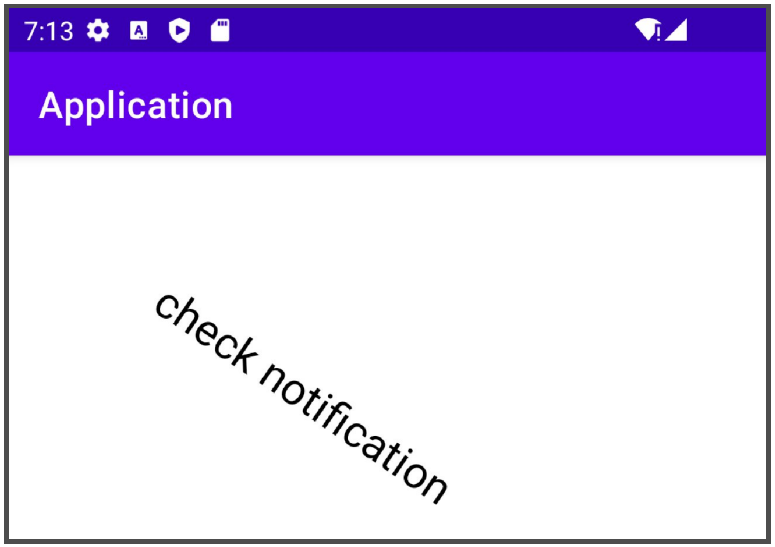
```
<?xml version="1.0" encoding="utf-8"?>
<alpha
xmlns:android="http://schemas.android.com/apk/res/android"
android:fromAlpha="0.0"
android:toAlpha="1.0"
android:duration="3000"
>
</alpha>
```

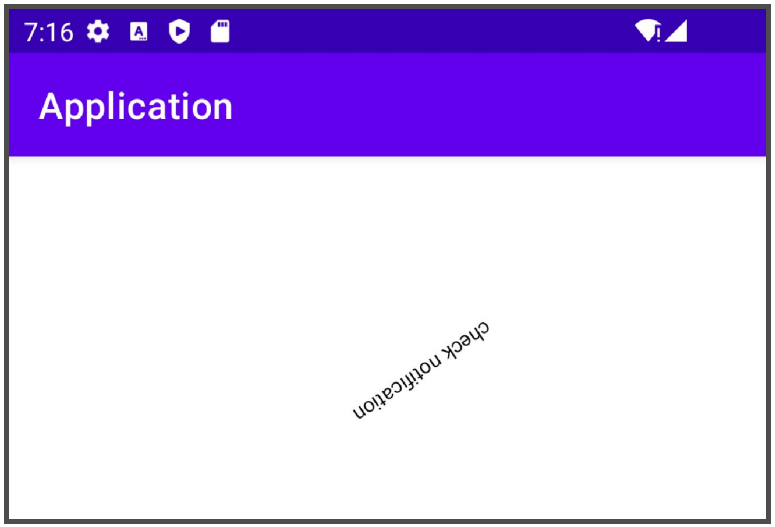
меняется прозрачность с 0 до 1 в течение трех секунд.

```
<?xml version="1.0" encoding="utf-8"?>
<rotate
xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="3000"
    android:fromDegrees="0"
    android:toDegrees="360" >

</rotate>
```

Поворот относительно левого верхнего угла (т.к. не указаны pivotX, pivotY) на 360 градусов в течение трех секунд

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

    <rotate
        android:duration="3000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="360" >
    </rotate>

    <scale
        android:duration="3000"
        android:fromXScale="0.1"
        android:fromYScale="0.1"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="1.0"
        android:toYScale="1.0" >
    </scale>

</set>
```





```
//Java
public class MainActivity extends Activity {
     final int MENU_ALPHA_ID = 1;
     final int MENU_SCALE_ID = 2;
     final int MENU_TRANSLATE_ID = 3;
     final int MENU_ROTATE_ID = 4;
     final int MENU_COMBO_ID = 5;
     TextView tv;
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
          tv = (TextView) findViewById(R.id.tv);
          // регистрируем контекстное меню для компонента tv
          registerForContextMenu(tv);
     }
     @Override
     public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo) {
          // TODO Auto-generated method stub
          switch (v.getId()) {
          case R.id.tv:
          // добавляем пункты
          menu.add(0, MENU_ALPHA_ID, 0, "alpha");
          menu.add(0, MENU_SCALE_ID, 0, "scale");
          menu.add(0, MENU_TRANSLATE_ID, 0, "translate");
          menu.add(0, MENU_ROTATE_ID, 0, "rotate");
          menu.add(0, MENU_COMBO_ID, 0, "combo");
          break;
          }
          super.onCreateContextMenu(menu, v, menuInfo);
     }
}
```

```kotlin
//Kotlin
class MainActivity : AppCompatActivity() {
    val MENU_ALPHA_ID = 1
    val MENU_SCALE_ID = 2
    val MENU_TRANSLATE_ID = 3
    val MENU_ROTATE_ID = 4
    val MENU_COMBO_ID = 5
    var tv: TextView? = null
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        tv = findViewById<View>(R.id.tv) as TextView
        // регистрируем контекстное меню для компонента tv
        registerForContextMenu(tv)
    }
    override fun onCreateContextMenu(menu: ContextMenu, v: View, menuInfo:
ContextMenu.ContextMenuInfo?) {
        //TODO Auto-generated method stub
        when (v.id) {
            R.id.tv -> {
                // добавляем пункты
                menu.add(0, MENU_ALPHA_ID, 0, "alpha")
                menu.add(0, MENU_SCALE_ID, 0, "scale")
                menu.add(0, MENU_TRANSLATE_ID, 0, "translate")
                menu.add(0, MENU_ROTATE_ID, 0, "rotate")
                menu.add(0, MENU_COMBO_ID, 0, "combo")
            }
        }
        super.onCreateContextMenu(menu, v, menuInfo)
    }
}
```

```
//Java
@Override
public boolean onContextItemSelected(MenuItem item) {
     Animation anim = null;
     // определяем какой пункт был нажат
     switch (item.getItemId()) {
          case MENU_ALPHA_ID:
          // создаем объект анимации из файла anim/myalpha
          anim = AnimationUtils.loadAnimation(this, R.anim.myalpha);
          break;
          case MENU_SCALE_ID:
          anim = AnimationUtils.loadAnimation(this, R.anim.myscale);
          break;
          case MENU_TRANSLATE_ID:
          anim = AnimationUtils.loadAnimation(this, R.anim.mytrans);
          break;
          case MENU_ROTATE_ID:
          anim = AnimationUtils.loadAnimation(this, R.anim.myrotate);
          break;
          case MENU_COMBO_ID:
          anim = AnimationUtils.loadAnimation(this, R.anim.mycombo);
          break;
     }
     // запускаем анимацию для компонента tv
     tv.startAnimation(anim);
     return super.onContextItemSelected(item);
}
```

```kotlin
//Kotlin
override fun onContextItemSelected(item: MenuItem): Boolean {
    var anim: Animation? = null
    when (item.getItemId()) {
        MENU_ALPHA_ID ->
            // создаем объект анимации из файла anim/myalpha
            anim = AnimationUtils.loadAnimation(this, R.anim.myalpha)
        MENU_SCALE_ID -> anim = AnimationUtils.loadAnimation(this,
R.anim.myscale)
        MENU_TRANSLATE_ID -> anim = AnimationUtils.loadAnimation(this,
R.anim.mytrans)
        MENU_ROTATE_ID -> anim = AnimationUtils.loadAnimation(this,
R.anim.myrotate)
        MENU_COMBO_ID -> anim = AnimationUtils.loadAnimation(this,
R.anim.mycombo)
    }    // запускаем анимацию для компонента tv

    tv!!.startAnimation(anim)
    return super.onContextItemSelected(item)
}
```

Разработка пользовательского интерфейса в Android

С Android поставляется простая и понятная среда с ограниченным **набором готовых графических элементов**. Android берет на себя также многие задачи, обычно сопровождающие конструирование и создание качественных пользовательских интерфейсов. Наряду с тем, что пользователь обычно хочет выполнять лишь одну конкретную операцию, это позволяет создавать эффективные пользовательские интерфейсы и удобные среды работы пользователей.

В составе Android SDK поставляется **множество элементов управления**, которые позволяют **создавать пользовательские интерфейсы** для приложений. Подобно другим SDK, в Android SDK имеются текстовые поля, кнопки, списки, таблицы и т.д. Кроме того, Android предоставляет коллекцию элементов управления, приспособленных для мобильных устройств.

Создание пользовательского интерфейса полностью в XML

Рассмотрим тот же пользовательский интерфейс в XML. По умолчанию в папке res/layout будет создан компоновочный XML-файл activity_main.xml. Дважды щелкните на этом файле, чтобы открыть его содержимое в визуальном редакторе Возможно, в начале представления будет находиться строка с текстом вроде "Hello World, MainActivity!" Перейдите на вкладку activity_main.xml (внизу), чтобы просмотреть XML код из файла activity_main.xml Здесь находятся элементы LinearLayout и TextView.

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

    <LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Name:" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="John Doe" />
    </LinearLayout>
```

```
    <LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Address:" />

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="911 Hollywood Blvd." />
    </LinearLayout>

</LinearLayout>
```

Создание пользовательского интерфейса в XML с кодом

Лучше всего проектировать пользовательские интерфейсы в XML, а затем обращаться к элементам управления в коде. Такой подход позволит связывать динамические данные с элементами, определенными на этапе проектирования. Более того, это и есть рекомендованный подход. Можно относительно легко создавать компоновки в XML, а затем с помощью кода заполнять их динамическими данными.

В листинге приведен тот же пользовательский интерфейс с несколько другой XML-разметкой: в нем элементам TextView назначены идентификаторы, с помощью которых к элементам можно обращаться в Java-коде.

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

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

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

        <TextView
            android:id="@+id/nameValue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
```

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

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/addr_text" />

        <TextView
            android:id="@+id/addrValue"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>

</LinearLayout>
```

Кроме идентификаторов для элементов TextView, которые будут заполняться в коде, имеются еще элементы TextView для меток, которые будут заполняться из файла строковых ресурсов. Это элементы TextView без идентификаторов и с атрибутами android:text. Строки для этих TextView берутся из файла strings.xml, который находится в папке /res/values. Пример файла strings.xml приведен в листинге.

```
<resources>

    <string name="app_name">hello99</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="addr_text">Address</string>
    <string name="name_text">Name</string>

</resources>
```

В листинге показано, как можно получить ссылки на элементы, определенные в XML, для задания их свойств. Такой код можно поместить в метод onCreate() соответствующего действия.

```
//Java
setContentView(R.layout.activity_main);
TextView nameValue = (TextView)findViewById(R.id.nameValue);
nameValue.setText("Mr. Hello World");
TextView addrValue = (TextView)findViewById(R.id.addrValue);
addrValue.setText("Moscow, Red Square");

//Kotlin
setContentView(R.layout.activity_main)
val nameValue = findViewById(R.id.nameValue) as TextView
nameValue.text = "Mr. Hello World"
val addrValue = findViewById(R.id.addrValue) as TextView
addrValue.text = "Moscow, Red Square"
```

Обратите внимание, что ресурс загружается с помощью вызова setContentView перед вызовом findViewById() т.к. невозможно получить ссылки на представления до их загрузки.

```
//Java
package com.example.hello99;

import android.os.Bundle;
import android.app.Activity;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends Activity {

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);
TextView nameValue = (TextView) findViewById(R.id.nameValue);
nameValue.setText("Mr. Hello World");
TextView addrValue = (TextView) findViewById(R.id.addrValue);
addrValue.setText("Moscow, Red Square");
}
}
```

```
//Kotlin
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    public override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val nameValue = findViewById<View>(R.id.nameValue) as
TextView
        nameValue.text = "Mr. Hello World"
        val addrValue = findViewById<View>(R.id.addrValue) as
TextView
        addrValue.text = "Moscow, Red Square"
    }
}
```

Текстовые поля

В Android вы можете использовать два текстовых поля: TextView и EditText. Первое используется для отображения текста без возможности его редактирования, а второе — это классическое текстовое поле с возможностью ввода и редактирования текста.

Несмотря на свою простоту, виджет TextView довольно часто используется в Android-программах для вывода инструкций использования программы и другого текста.

В файле разметки значение TextView можно установить так:

```
android:text="Text";
```

В Java-коде значение виджета устанавливается так:

```
//Java
TextView text = (TextView)findViewById(R.id.text1);
text.setText("Sample text");
//Kotlin
val text = findViewById<View>(R.id.text1) as TextView
text.text = "Sample text"
```

Иначе обстоит дело, если вы планируете создать приложения с многоязыковой поддержкой пользовательского интерфейса. Тогда непосредственно в файле разметки значение (текстовую строку) указывать не нужно. Вместо этого создается ссылка на текстовый XML-ресурс:

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

Здесь **str_value** — это имя строкового ресурса, описанного в файле strings.xml. В Java-коде установить имя ресурса можно тем же методом setText():

```
//Java
TextView text = (TextView)findViewById(R.id.text1);
text.setText(R.string.str_value);

//Kotlin
val text = findViewById<View>(R.id.text1) as TextView
text.setText(R.string.str_value)
```

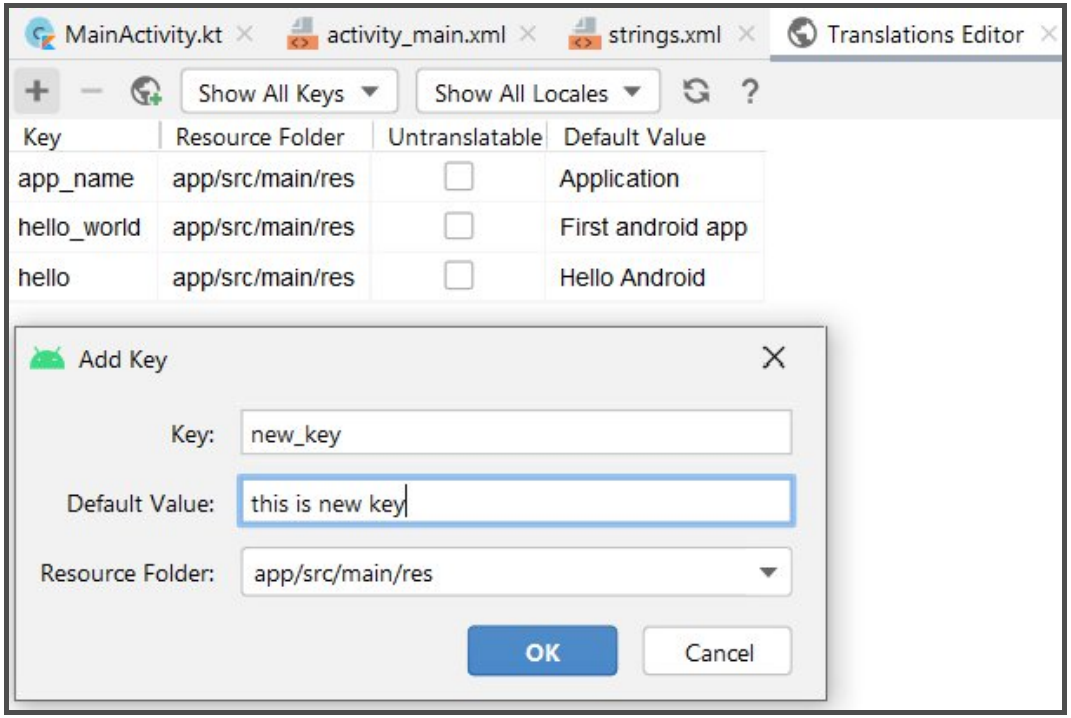
У элемента TextView есть много методов и свойств. Рассмотрим только основные свойства, относящиеся к отображению текста.
Размер шрифта можно задать свойством **android:textSize**. **Размер задается** в пикселях (px), независимых от плотности пикселях (dp), независимых от масштабирования пикселях (sp), пунктах (pt), дюймах (in) и миллиметрах (mm):

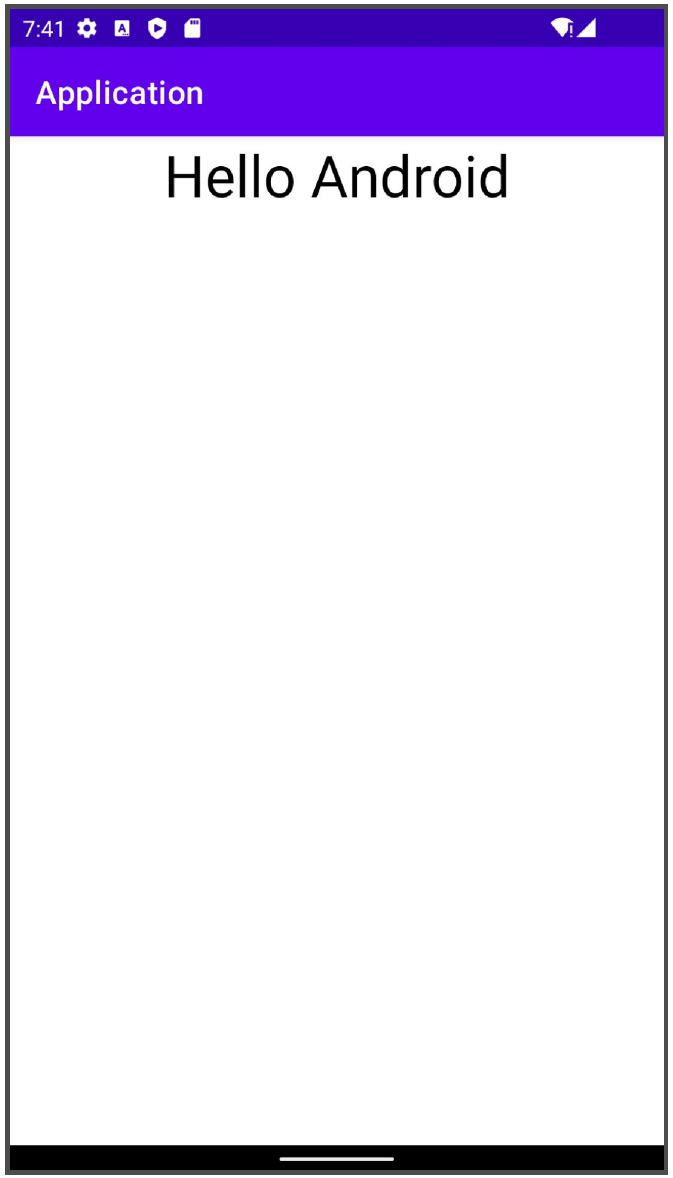
android:textSize="14pt"
Стиль текста задается свойством android:textStyle:
normal — обычное начертание символов;
bold — полужирное начертание символов;
italic — курсив.

Пример:
android:textStyle="bold"

Цвет шрифта задается свойством **android:textColor**. Цвет указывается в шестнадцатеричной кодировке в формате **#RGB** или **#ARGB**. Во втором случае значение A — это прозрачность. Если A = 0, то прозрачность 100% (элемент не будет виден).

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:id="@+id/txt1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="20pt"
android:textStyle="bold"
android:text="@string/hello"
/>
</LinearLayout>
```





Класс EditText

Класс EditText является подклассом TextView. Как понятно из имени, элементы EditText **позволяют редактировать текст**. Одним из наиболее важных свойств класса EditText является inputType. Для него можно задать значение **textAutoCorrect**, и элемент будет исправлять распространенные грамматические ошибки. А значение **textCapWords** приведет к тому, что текст будет набираться заглавными буквами. Есть и другие параметры, применимые **только для телефонных номеров или паролей**.

Существуют старые, но не рекомендуемые сейчас, способы задать набор заглавными буквами, многострочный текст и другие возможности. Если их указать без свойства inputType, то они учитываются, но в присутствии inputType игнорируются.

Можно ограничить ввод пользователя только одной строкой, задав в свойстве **singleLine значение true**. В этом случае пользователь будет продолжать печатать в той же строке. Если не указать параметр textMultiLine, по умолчанию будет выполняться ввод в одну строку. Так что если вам нужен многострочный ввод, нужно просто задать в inputType значение **textMultiLine**.

Одной из удобных характеристик элемента EditText является возможность задания **текста подсказки**. Такой текст выводится бледными символами и исчезает, если пользователь начинает вводить текст. Подсказки предназначены для того, чтобы пользователь знал, что нужно вводить в данном поле, без необходимости выделять и стирать текст подсказки. В XML такой атрибут имеет вид вроде **`android:hint="Текст_подсказки"`** или **`android:hint="@string/текст_подсказки"`**, где текст_подсказки — имя ресурсной строки из файла /res/values/strings . xml. В коде можно вызвать метод setHint () с указанием последовательности символов или идентификатора ресурса.

Текстовое поле EditText позволяет вводить и редактировать текст. Основной метод этого поля — метод **`getText()`**, позволяющий получить введенный пользователем текст. Значение, возвращаемое методом getText(), имеет тип Editable. По сути, Editable — это надстройка над String, но в отличие от него значение типа Editable может быть изменено в процессе выполнения программы (String является неизменяемым типом, при изменении типа String попросту создается новый экземпляр String, содержащий новое значение).

Кроме метода getText() вам может понадобиться метод **`selectAll()`**, выделяющий весь текст в окне EditText. Если весь текст выделять не нужно, можно использовать метод **`setSelection()`**:

```
setSelection(int start, int stop)
```

Данный метод выделяет участок текста, начиная с позиции start до позиции stop.

Установить тип начертания шрифта можно с помощью метода setTypeface, например:

txt1.setTypeface(null, Typeface.NORMAL);

Вместо NORMAL можно указать BOLD и ITALIC.
Для **добавления поля EditText** в разметку окна нужно добавить следующий код в файл разметки:

```
<EditText
android:id="@+id/entry1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Simple text" />
```

Кнопки

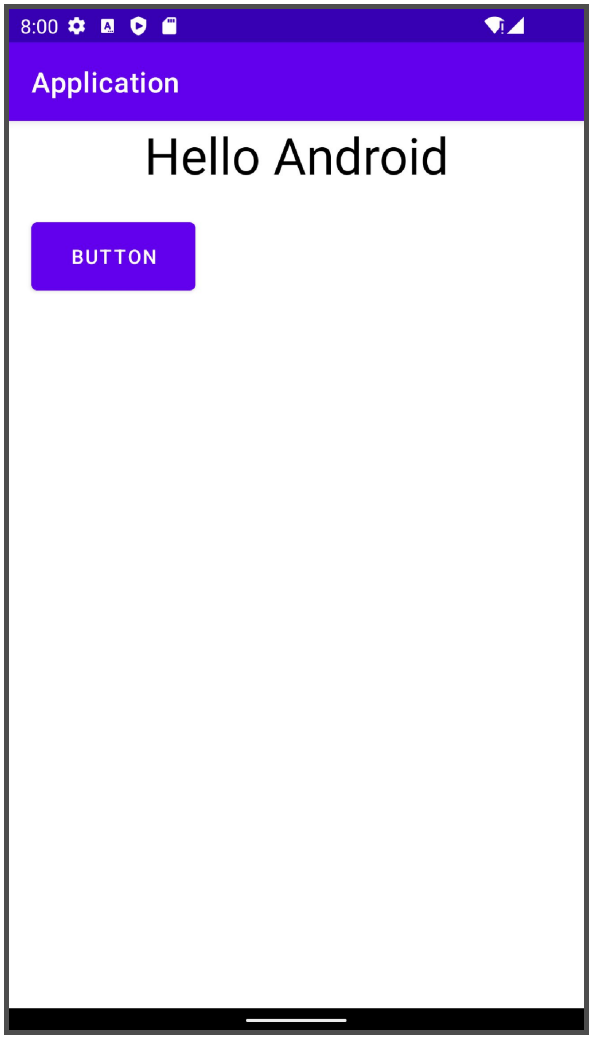
Кнопки — это очень важный элемент пользовательского интерфейса. К кнопкам относятся не только непосредственно сами кнопки, но и переключатели (независимые и зависимые). Всего к кнопкам можно отнести пять классов:

Button;
CheckButton;
ToggleButton;
RadioButton;
ImageButton.

Классы **CheckButton, ToggleButton и RadioButton** являются потомками класса CompoundButton, который, в свою очередь, является потомком класса Button. А вот класс **ImageButton** является потомком класса ImageView, поэтому ImageButton является больше изображением, нежели кнопкой в прямом понимании этого слова.

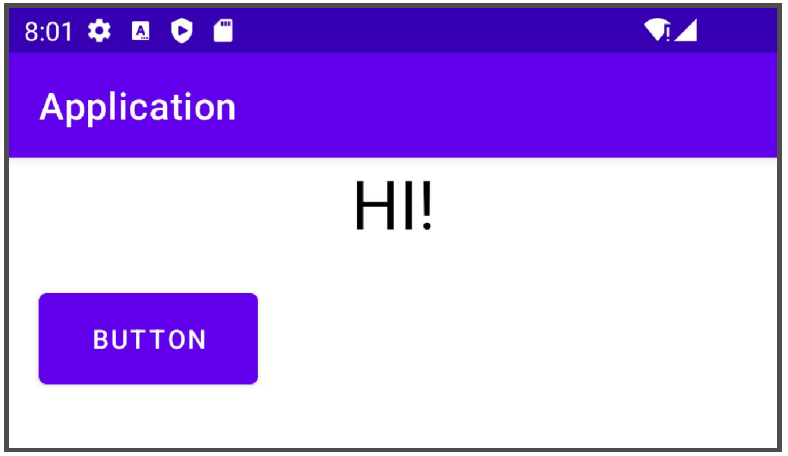
Button — обычная кнопка

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/r
es/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:id="@+id/txt1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="20pt"
android:textStyle="bold"
android:text="@string/hello"
/>
<Button
android:text="Button"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
```



Обратите внимание на идентификаторы текстового поля и кнопки. Текстовое поле называется txt1, а кнопка — button1. Эти идентификаторы будут использоваться в Java-коде.

```java
//Java
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;
import android.widget.Button;
import android.view.View;

public class MainActivity extends Activity {
private TextView txt1;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         txt1 = (TextView)findViewById(R.id.txt1);
        /* находим кнопку */
        final Button button1 = (Button)findViewById(R.id.button1);
        button1.setOnClickListener(new View.OnClickListener () {
        public void onClick(View v) {
        // TODO Auto-generated method stub
        txt1.setText("Hi!");
        }});
    }
}
```



```kotlin
//Kotlin
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private var txt1: TextView? = null
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        txt1 = findViewById<View>(R.id.txt1) as TextView
        /* находим кнопку */
        val button1: Button = findViewById<View>(R.id.button1) as
Button
        button1.setOnClickListener(View.OnClickListener {
            // TODO Auto-generated method stub
            txt1!!.text = "Hi!"
        })
    }
}
```

RadioButton — зависимые переключатели

Зависимые переключатели (виджеты класса RadioButton) используются **внутри контейнера RadioGroup** — группы зависимых переключателей. Зависимый переключатель позволяет выбрать только одну из опций в одной из групп переключателей. В одной из групп может быть активным только один переключатель.

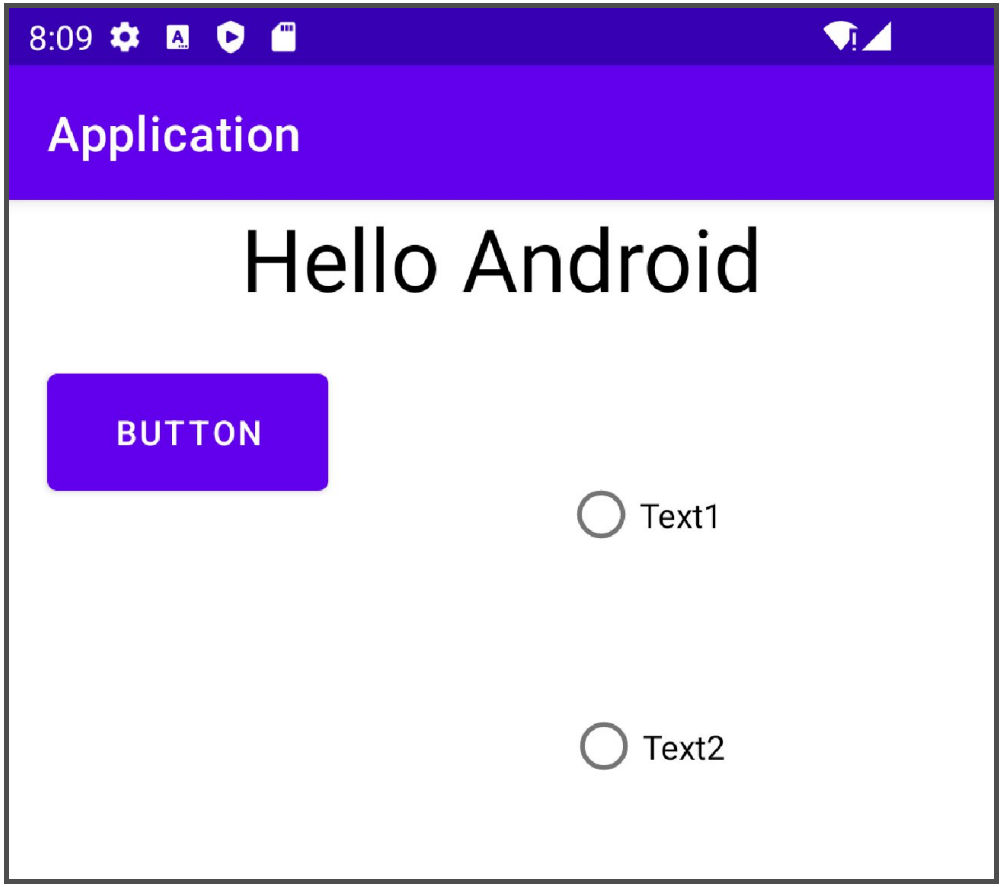
В основном используется **три метода**:

toggle() — инвертирует состояние зависимого переключателя;

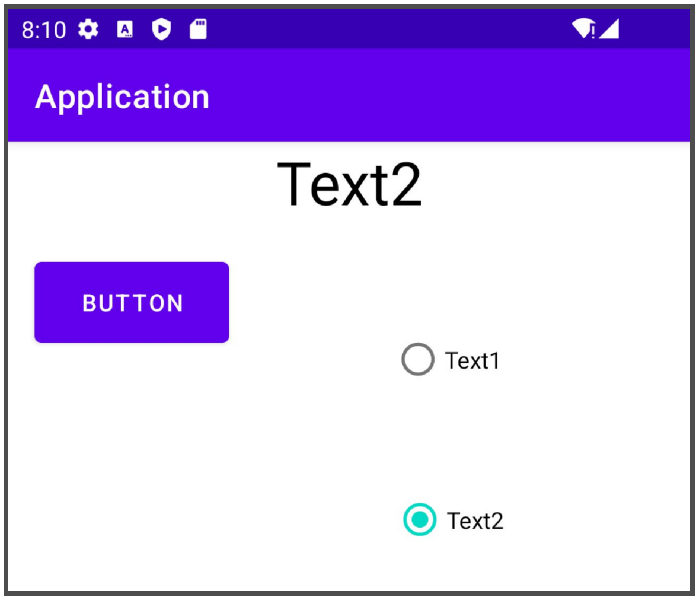
isChecked() — возвращает значение зависимого переключателя (true — активен);

setChecked() — изменяет значение переключателя в зависимости от переданного параметра.

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<RadioButton
  android:text="Value 1"
android:id="@+id/r1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
    />
<RadioButton
    android:text="Value 2"
android:id="@+id/r2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
    />

<TextView
android:id="@+id/txt1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="20pt"
android:textStyle="bold"
android:text="@string/hello"
/>
<Button
android:text="Button"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</RadioGroup>
```



```java
//Java
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;
import android.widget.Button;
import android.view.View;
import android.widget.RadioButton;

public class MainActivity extends Activity {
private TextView txt1;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         txt1 = (TextView)findViewById(R.id.txt1);
        /* находим кнопку */
        final Button button1 = (Button)findViewById(R.id.button1);
        final RadioButton r1 = (RadioButton)findViewById(R.id.r1);
        final RadioButton r2 = (RadioButton)findViewById(R.id.r2);
        button1.setOnClickListener(new View.OnClickListener () {
        public void onClick(View v) {
        // TODO Auto-generated method stub
        if (r1.isChecked()) txt1.setText("Value 1");
        if (r2.isChecked()) txt1.setText("Value 2");

        }
        });
    }
}
```



```kotlin
//Kotlin
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.RadioButton
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private var txt1: TextView? = null
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        txt1 = findViewById<View>(R.id.txt1) as TextView
        /* находим кнопку */
        val button1 = findViewById<View>(R.id.button1) as Button
        val r1 = findViewById<View>(R.id.r1) as RadioButton
        val r2 = findViewById<View>(R.id.r2) as RadioButton
        button1.setOnClickListener {
            // TODO Auto-generated method stub
            if (r1.isChecked) txt1!!.text = "Value 1"
            if (r2.isChecked) txt1!!.text = "Value 2"
        }
    }
}
```

CheckBox — независимые переключатели

Переключатели CheckBox не привязываются к какому-нибудь контейнеру (вроде RadioGroup) и значение независимого переключателя не зависит от состояния других переключателей, поэтому данные переключатели и называются независимыми.

Для работы с CheckBox вы можете использовать те же методы (isChecked(), toggle(), setChecked()), что и в случае с RadioButton. В файле разметки независимые переключатели выглядят так:

```
<CheckBox
android:id="@+id/checkbox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="I agree" />
```

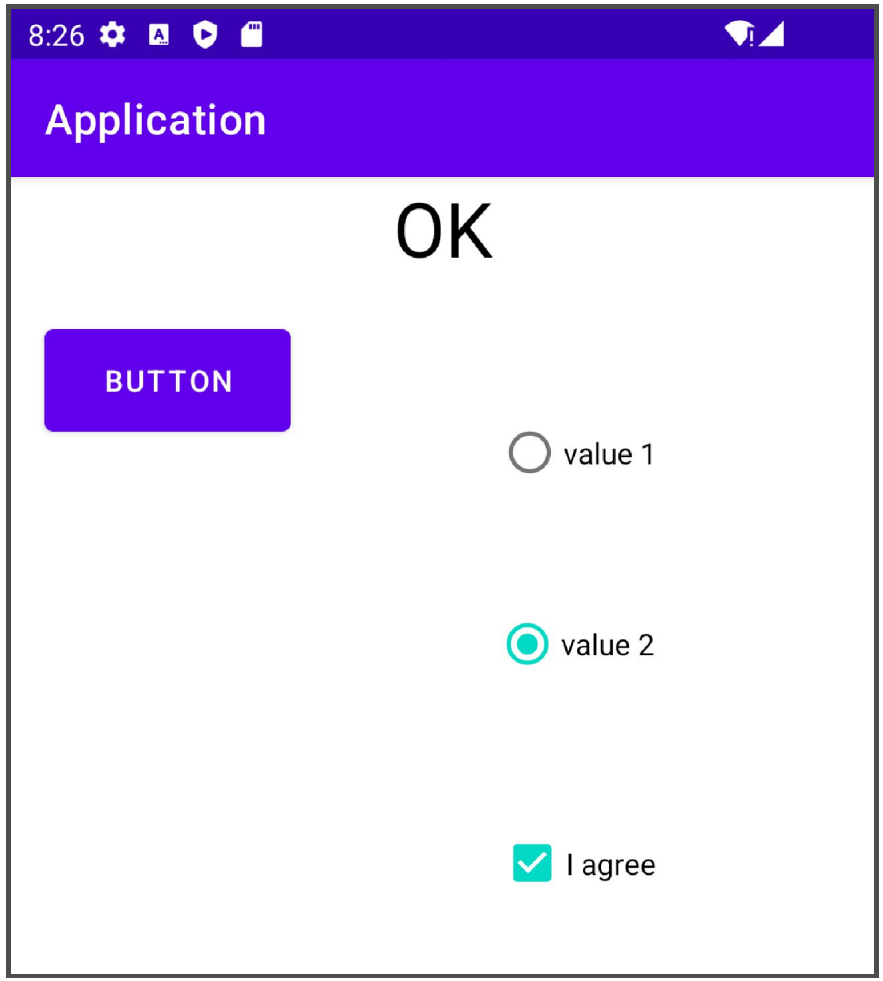
Найти CheckBox в Java-коде можно так:

```
//Java
final CheckBox check = (CheckBox)findViewById(R.id.checkbox);
//Kotlin
val check = findViewById(R.id.checkbox) as CheckBox
```

Далее можно проверить, включен ли переключатель:

```
if (check.isCheked()) txt1.setText("OK");
```

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

    ....................................
    />
    <TextView
android:id="@+id/txt1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="20pt"
android:textStyle="bold"
android:text="@string/hello"
/>
<Button
android:text="Button"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</RadioGroup>
<CheckBox
android:id="@+id/checkbox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="I agree" />
</LinearLayout>
```



```
//Java
package com.example.helloworld;
import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;
import android.widget.Button;
import android.view.View;
import android.widget.RadioButton;
import android.widget.CheckBox;

public class MainActivity extends Activity {
     private TextView txt1;
          @Override
          public void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.activity_main);
                      txt1 = (TextView)findViewById(R.id.txt1);
                    /* находим кнопку */
                    final Button button1 = (Button)findViewById(R.id.button1);
                    final RadioButton r1 = (RadioButton)findViewById(R.id.r1);
                    final RadioButton r2 = (RadioButton)findViewById(R.id.r2);
                    final CheckBox check = (CheckBox)findViewById(R.id.checkbox);
                    button1.setOnClickListener(new View.OnClickListener () {
                         public void onClick(View v) {
                              // TODO Auto-generated method stub
                              if (r1.isChecked()) txt1.setText("Value 1");
                              if (r2.isChecked()) txt1.setText("Value 2");
                              if (check.isChecked()) txt1.setText("OK");
                         }
                    });
          }
}
```

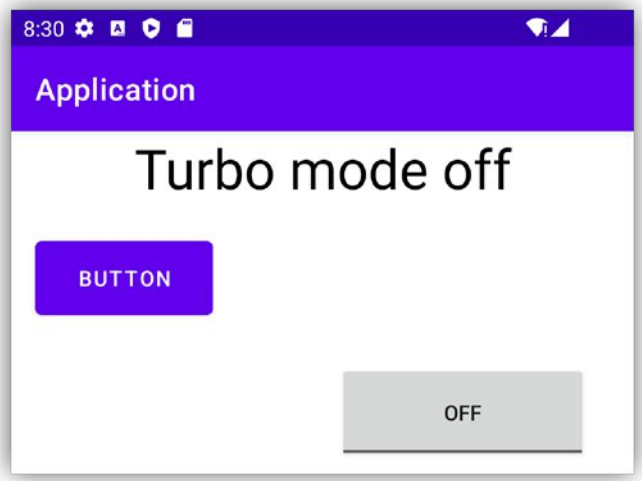
```kotlin
//Kotlin
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.CheckBox
import android.widget.RadioButton
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
    private var txt1: TextView? = null
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        txt1 = findViewById<View>(R.id.txt1) as TextView
        /* находим кнопку */
        val button1 = findViewById<View>(R.id.button1) as Button
        val r1 = findViewById<View>(R.id.r1) as RadioButton
        val r2 = findViewById<View>(R.id.r2) as RadioButton
        val check = findViewById<View>(R.id.checkbox) as CheckBox
        button1.setOnClickListener {
            // TODO Auto-generated method stub
            if (r1.isChecked) txt1!!.text = "Value 1"
            if (r2.isChecked) txt1!!.text = "Value 2"
            if (check.isChecked) txt1!!.text = "OK"
        }
    }
}
```

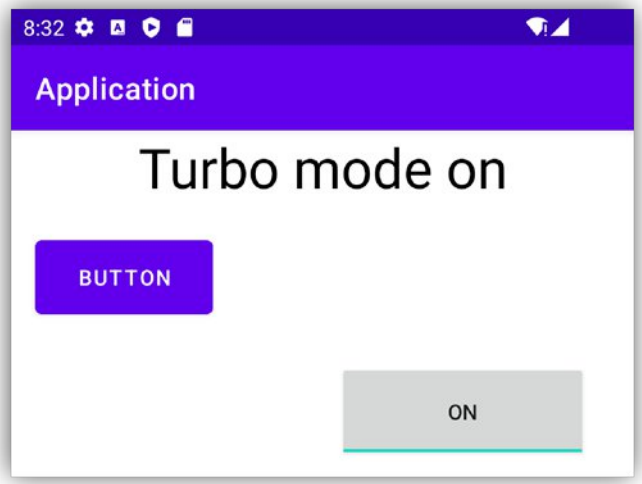
ToggleButton — кнопка включено/выключено

Кнопка **ToggleButton** может находиться в одном из положений — включено или выключено. Когда кнопка нажата, горит зеленый индикатор, как бы «встроенный» в кнопку, также видно визуально, что кнопка немного вдавлена.

У кнопки есть **два основных свойства** — **`android:textOff`** и **`android:textOn`**. Первое устанавливает текст кнопки, когда она выключена, а второе — когда включена. В программном коде этим свойствам соответствуют методы **`setTextOff()`** и **`setTextOn()`**.

С помощью метода **`setChecked(boolean checked)`** вы можете программно изменить состояние кнопки. При изменении состояния генерируется событие **`onCheckedChanged()`**.





ImageButton

Виджет ImageButton — что-то среднее между изображением и кнопкой. Вместо текста у этой кнопки будет изображение, что позволяет создавать более привлекательные кнопки.

Установить изображение кнопки можно или атрибутом **android:src** элемента <ImageButton>, или методом **setImageResource(int).**

В файле разметки элемент <ImageButton> описывается так:

```
<ImageButton
        android:id="@+id/b23"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="22dp"
        android:src="@drawable/ic_action_search" />
```

Работа с такой кнопкой осуществляется так же, как и с обычной кнопкой. Первым делом нужно подключить необходимый пакет, объявить переменную и найти сам элемент в файле разметки:

```
import android.widget.ImageButton;
...
//Java
ImageButton button;
...
button = (ImageButton)findViewById(R.id.button);
button.setImageResource(R.drawable.play);
final ImageButton button23= (ImageButton)findViewById(R.id.b23);
button23.setImageResource(R.drawable.ic_launcher);

//Kotlin
var button: ImageButton? = null
…
button = findViewById<ImageButton>(R.id.button)
button?.setImageResource(R.drawable.play)
val button23 = findViewById<ImageButton>(R.id.b23)
button23.setImageResource(R.drawable.ic_launcher)
```

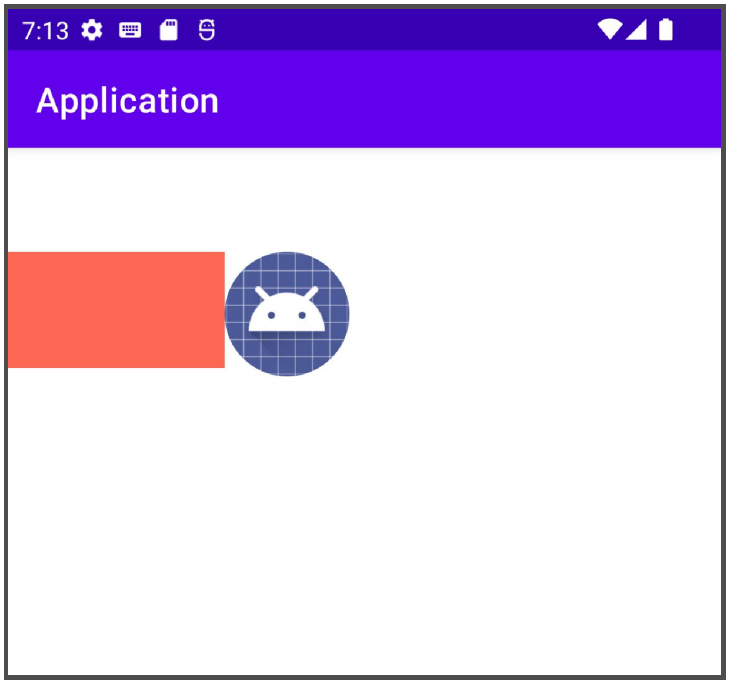
Последние два оператора находят кнопку ImageButton и устанавливают для нее изображение с именем *ic_launcher*, которое находится в папке res/drawable или res/drawable-* (имя завист от разрешения экрана — для более новых платформ Android). Лучше всего использовать файлы в формате PNG. В нашем случае (ресурс R.drawable.*ic_launcher*) в каталоге res/drawable должен быть файл *ic_launcher*.png.

Обработчик нажатия кнопки устанавливается так же, как и для обычной кнопки Button.

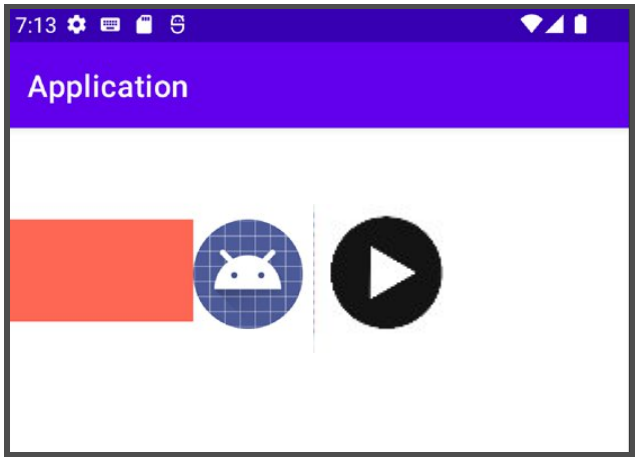
Элемент ImageView

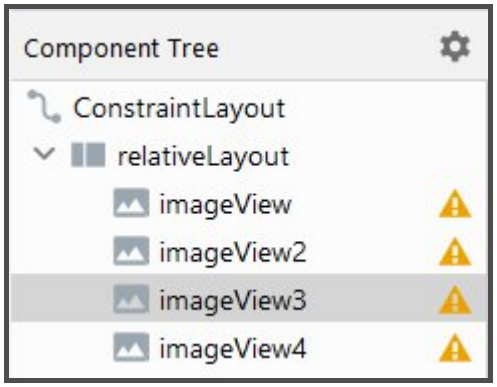
Рассмотрим элемент ImageView. Он применяется для вывода изображений из файла, поставщика контента или графического ресурса. Можно даже просто указать цвет, и ImageView выведет этот цвет.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
<ImageView
    android:id="@+id/image1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />
<ImageView
    android:id="@+id/image2"
    android:layout_width="202dp"
    android:layout_height="61dp"
    android:src="@drawable/ic_launcher" />
</LinearLayout>
```



```
<ImageView
    android:id="@+id/image3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<ImageView
    android:id="@+id/image4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:maxHeight="50dip"
    android:maxWidth="35dip"
    android:scaleType="centerInside"
    android:src="@drawable/Clipboard01" />
```





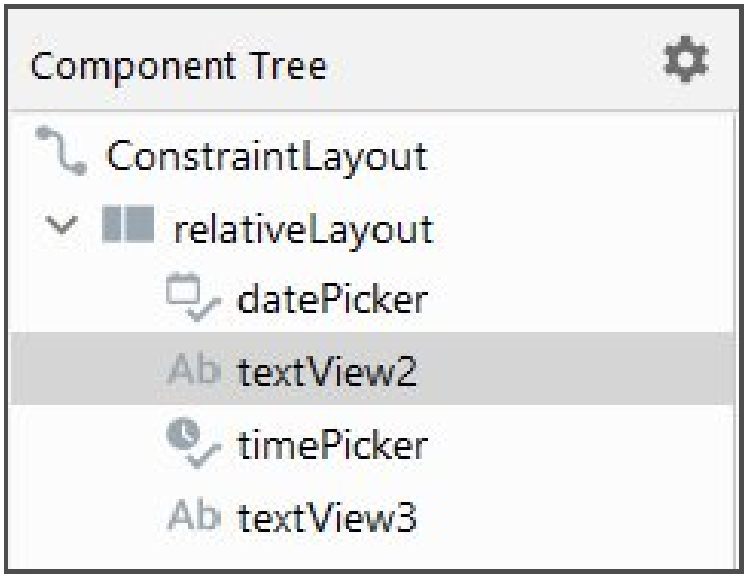

```
//Java
ImageView imgView =(ImageView)findViewByld(R.id.image3);
imgView.setImageResource( R.drawable.icon );
imgView.setImageBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.manateel4);
imgView.setImageDrawable(Drawable.createFromPath("/mnt/sdcard/dave2.jpg"));
imgView.setImageURI(Uri.parse("file://mnt/sdcard/dave2.jpg"));

//Kotlin
var imgView = findViewById<ImageView>(R.id.image3)
imgView.setImageResource(R.drawable.icon)
imgView.setImageBitmap(BitmapFactory.decodeResource(this.getResources(), R.drawable.manateel4))
imgView.setImageDrawable(Drawable.createFromPath("/mnt/sdcard/dave2.jpg"))
imgView.setImageURI(Uri.parse("file://mnt/sdcard/dave2.jpg"))
```

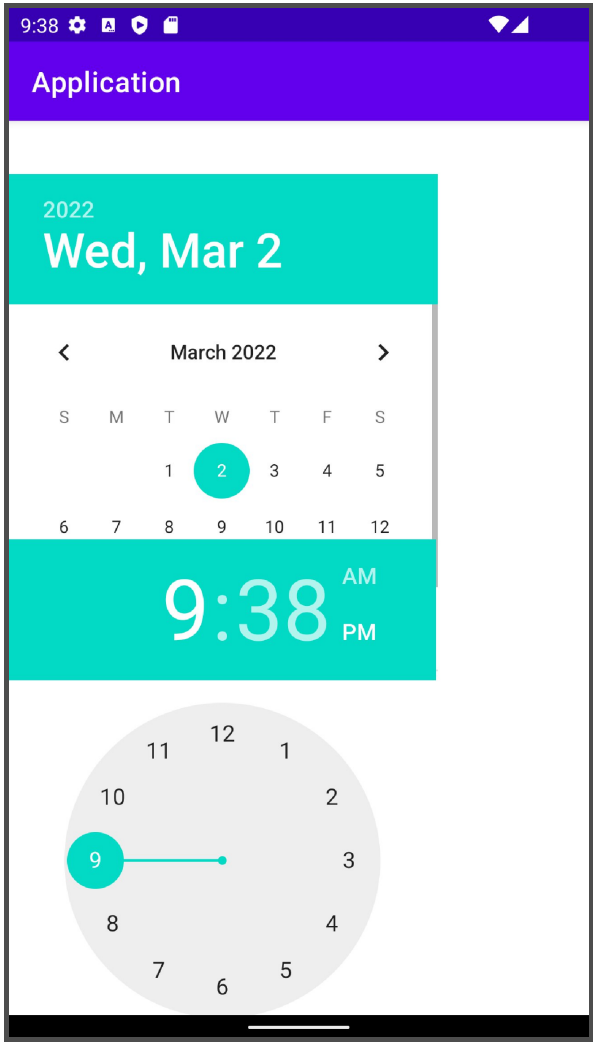
Элементы указания даты и времени

Элементы указания даты и времени часто встречаются во многих наборах виджетов. В Android имеется несколько таких элементов, и некоторые из них мы рассмотрим ниже. Это элементы **DatePicker, TimePicker, DigitalClock и AnalogClock.**

Элементы DatePicker и TimePicker

Как понятно из названий, элемент DatePicker предназначен для выбора даты, а элемент TimePicker — для выбора времени.





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

    <TextView
        android:id="@+id/dateDefault"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <DatePicker
        android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/timeDefault"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TimePicker
        android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Рассмотрим XML-код компоновки: определить такие элементы довольно легко. Как и в случае других элементов из инструментального набора Android, к этим элементам можно обращаться программным образом, чтобы инициализировать их либо извлекать из них данные. Пример инициализации приведен в листинге.

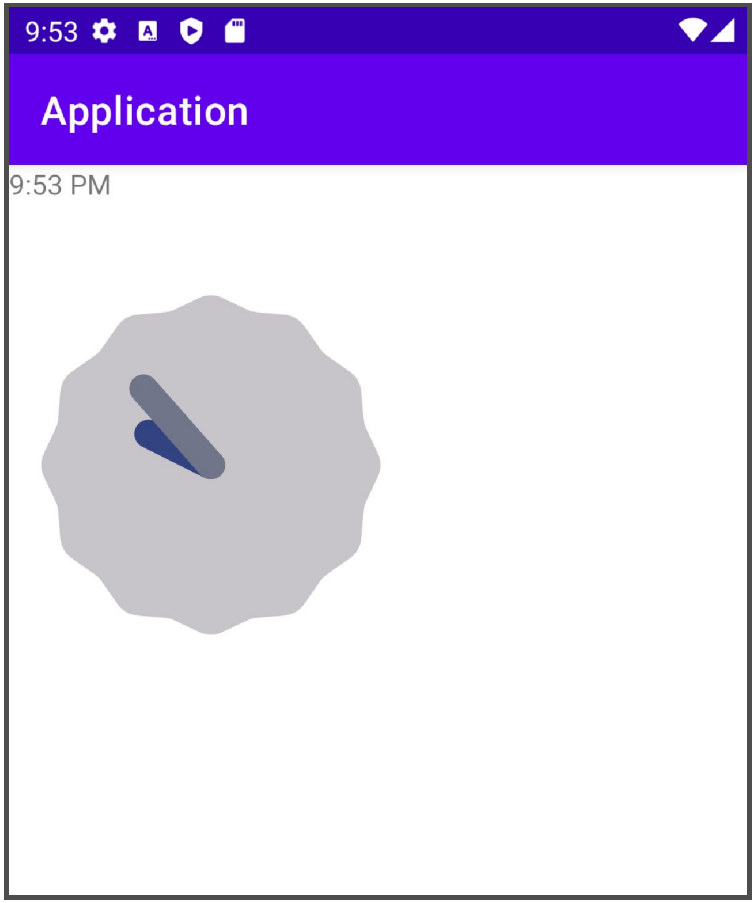
```
//Java
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView dateDefault = (TextView)findViewById(R.id.dateDefault);
        TextView timeDefault = (TextView)findViewById(R.id.timeDefault);
        DatePicker dp =   (DatePicker)this.findViewById(R.id.datePicker);
        // Месяцы  (и только месяцы)  нумеруются с нуля.  Для вывода нужно
        добавить 1.
        dateDefault.setText("Date defaulted to  "  +   (dp.getMonth()   + 1)
        + "/"  + dp.getDayOfMonth()   + "/" + dp.getYear());
        // А здесь из номера декабря   (12)   вычитается 1,  чтобы получить
        внутренний номер
        dp.init(2008,   11,   10,  null);
        TimePicker tp = (TimePicker)this.findViewById(R.id.timePicker);
        Formatter timeF = new Formatter();
        timeF.format("Time defaulted to %d:%02d", tp.getCurrentHour (),
        tp.getCurrentMinute()); timeDefault.setText(timeF.toString());
        tp.setIs24HourView(true); tp.setCurrentHour(new Integer (10));
        tp.setCurrentMinute(new Integer(10));
    }
}
```

```kotlin
//Kotlin
class MainActivity : AppCompatActivity() {
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val dateDefault = findViewById<View>(R.id.dateDefault) as TextView
        val timeDefault = findViewById<View>(R.id.timeDefault) as TextView
        val dp = findViewById<View>(R.id.datePicker) as DatePicker
        // Месяцы (и только месяцы) нумеруются с нуля. Для вывода нужно
добавить 1.
        dateDefault.text = "Date defaulted to  " + (dp.month + 1) + "/" +
dp.dayOfMonth + "/" + dp.year
        // А здесь из номера декабря (12) вычитается 1, чтобы получить
внутренний номер
        dp.init(2008, 11, 10, null)
        val tp = findViewById<View>(R.id.timePicker) as TimePicker
        val timeF = Formatter()
        timeF.format("Time defaulted to %d:%02d", tp.currentHour,
tp.currentMinute)
        timeDefault.setText(timeF.toString())
        tp.setIs24HourView(true)
        tp.currentHour = 10
        tp.currentMinute = 10
    }
}
```

Элементы DlgitalClock и AnalogClock

В Android имеются элементы DigitalClock и AnalogClock.

Как видно на рисунке, цифровые часы показывают не только часы и минуты, но и секунды. У аналоговых часов в Android только две стрелки — одна для часов и другая для минут. Чтобы добавить их в компоновку, понадобится XML-код, подобный представленному в листинге.




```
 <DigitalClock
android:layout_width="wrap_content"
android:layout_height="wrap_content"
<AnalogClock
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

Оба эти элемента предназначены только для вывода текущего времени, но не для изменения даты и времени. То есть если нужно изменять дату или время, понадобятся элементы DatePicker/TimePicker или DatePickerDialog/TimePickerDialog.