**Меню**

[Создание меню при помощи ресурсов](http://developer.alexanderklimov.ru/android/theory/menu.php#xml)  
[Выбор пунктов меню](http://developer.alexanderklimov.ru/android/theory/menu.php#choosemenu)  
[Программное создание меню](http://developer.alexanderklimov.ru/android/theory/menu.php#code)  
[Программное открытие или закрытие меню](http://developer.alexanderklimov.ru/android/theory/menu.php#openclose)  
[Программное удаление пункта меню](http://developer.alexanderklimov.ru/android/theory/menu.php#removemenu)  
[Определить наличие кнопки Menu](http://developer.alexanderklimov.ru/android/theory/menu.php#hardmenu)  
[Разметка для меню](http://developer.alexanderklimov.ru/android/theory/menu.php#layout)  
[Меню в фрагментах](http://developer.alexanderklimov.ru/android/theory/menu.php#fragment)

Меню выбора опций (Options Menu)

За меню отвечает класс **android.view.Menu**. Каждая активность связана с одним объектом меню. Само меню содержит пункты меню (класс **android.view.MenuItem**) и подменю (класс **android.view.SubMenu**).

При нажатии кнопки **Menu** на старых телефонах появляется набор пунктов меню, прикрепляемый к активности. Меню может содержать значки. Такое меню может содержать шесть пунктов (как правило). При наличии более шести пунктов используется расширенное меню - в этом случае вместо шестого пункта появляется пункт **Опции** (More). При нажатии данного пункта показывается расширенное меню со списком пунктов, которые не поместились в основной части меню выбора опций.

Когда меню открывается впервые, Android вызывает метод **onCreateOptionsMenu()**, передавая в качестве параметра объект **Menu**. Меню можно создавать в виде ресурсов в XML-файле или использовать метод **add()**.

В стандартном проекте при выборе обычного шаблона уже есть заготовка для меню из одного пункта **Settings** и вызов метода для меню (вы об этом уже знаете).

Создание меню при помощи ресурсов

Рассмотрим работу с меню через ресурсы. Для создания меню используются ресурсы, которые должны храниться в XML-файле. Сам файл должен находиться в папке **res/menu/** вашего проекта. Меню состоит из следующих элементов:

**<menu>**

Определяет меню, которое будет содержать пункты меню. Элемент <menu> должен быть корневым элементом в XML-структуре файла и может содержать один или несколько элементов <item> и <group>

**<item>**

Создает непосредственно пункты меню. Данный элемент может иметь вложенный элемент <menu> для создания подменю

**<group>**

При желании можете также использовать невидимый контейнер для элементов <item>. Это позволяет достичь некоторых эффектов

Предположим, мы решили использовать меню для какой-нибудь игры. Создадим новый файл *game\_menu.xml*:

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

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

<item android:id="@+id/new\_game"

android:title="@string/new\_game" />

<item android:id="@+id/help"

android:title="@string/help" />

</menu>

Мы создали меню с двумя пунктами. Каждый пункт включает в себя следующие атрибуты:

**android:id**

Идентификатор пункта меню, по которому приложение может распознать при выделении пункта меню пользователем

**android:title**

Текст, который будет выводиться в меню

Существуют и другие атрибуты для элемента **item**, например *android:icon="@drawable/home"* позволит также вывести значок для пункта меню, а *android:enabled="false"* позволяет сделать пункт меню недоступным.

Атрибут **android:titleCondensed** применяется в том случае, если обычный заголовок слишком широкий и не «помещается» в выбранном элементе меню.

Атрибут **android:orderInCategory** определяет порядок, в котором отображаются элементы меню MenuItems.

Кстати, вы можете использовать встроенные системные значки Android. Например, **android:icon="@android:drawable/ic\_menu\_help"** позволит вам вывести значок помощи, который зашит в систему. Подробнее о системных значках почитайте в статье [Системные графические ресурсы](http://developer.alexanderklimov.ru/android/system-drawables.php)

При создании меню мы указали на строковые ресурсы **@string/new\_game** и **@string/help**. Необходимо добавить новые строки в файле **strings.xml**:

<string name="new\_game">Новая игра</string>

<string name="help">Справка</string>

Теперь нужно внести изменения в классе активности, в котором будет выводиться меню. Программа должна сконвертировать созданный нами ресурс меню в программный объект. Для этой цели существует специальный метод **MenuInflater.inflate()**, который вызывается в специальном методе обратного вызова **onCreateOptionsMenu()**. Данный метод и предназначен для вывода меню при нажатии кнопки **MENU** на устройстве:

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.game\_menu, menu);

return true;

}

После вставки кода среда разработки попросит импортировать недостающие пространства имен.

import android.view.Menu;

import android.view.MenuInflater;

Метод **onCreateOptionsMenu()** метод инициирует первое появление меню на экране и принимает в качестве параметра объект Menu (для старых устройств). Вы можете сохранить ссылку на меню и использовать ее в любом месте кода, пока метод **onCreateOptionsMenu()** опять не будет вызван. Вам необходимо всегда использовать реализацию этого обработчика из родительского класса, потому как она при необходимости автоматически включает в меню дополнительные системные пункты. В новых устройствах метод вызывается при создании активности. Метод должен возвращать значение **true**, чтобы меню было видимым на экране.

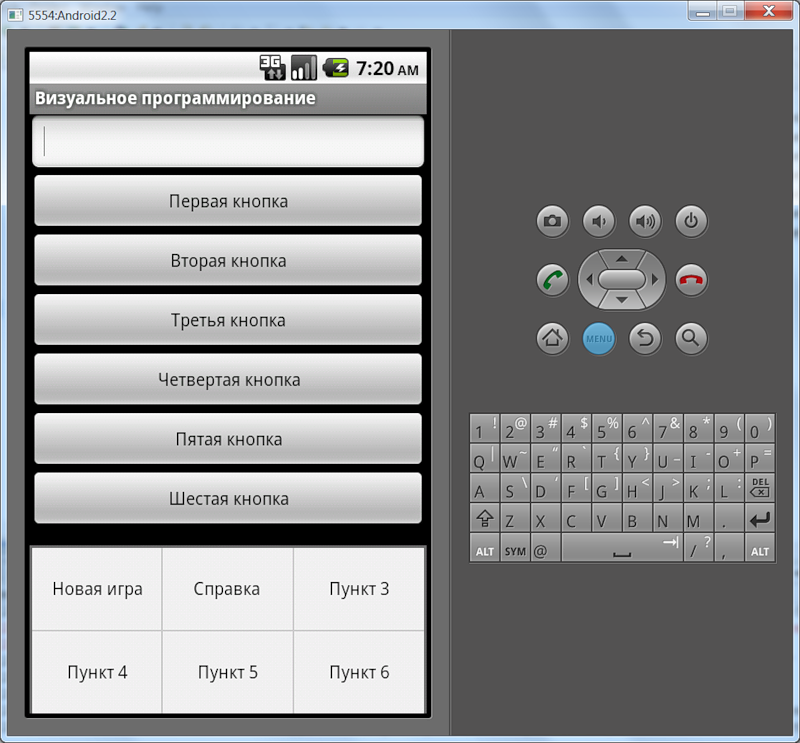
Запустив программу, нажмите кнопку **MENU** на эмуляторе, чтобы увидеть созданное меню.



Метод **getMenuInflater()** возвращает экземпляр класса **MenuInflater**, который мы используем для чтения данных меню из XML.

Как видите, меню появляется в нижней части экрана. Всего можно одновременно вывести на экран шесть пунктов меню. Если пунктов больше, то будет выведено пять пунктов плюс шестой пункт **More**, который позволит увидеть остальные пункты. Давайте проверим и добавим новые пункты меню.

Сначала добавим шесть пунктов.



Добавим еще один пункт к меню, чтобы их стало семь.



Выбор пунктов меню

Мы научились создавать меню. Но пока оно бесполезно, так как пункты меню никак не реагируют на наши нажатия. Для обработки нажатий пунктов меню служит метод **onOptionsItemSelected()**. Метод распознает пункт, выбранный пользователем, через **MenuItem**. Мы можем теперь определить выбранный пункт через вызов **getItemId()**, который возвращает идентификатор пункта меню. Далее через оператор **switch** нам остается определить нужные команды:

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

// Операции для выбранного пункта меню

switch (item.getItemId())

{

case R.id.new\_game:

newGame();

return true;

case R.id.help:

showHelp();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

public void newGame()

{

edtext.setText("Выбран пункт Новая игра");

}

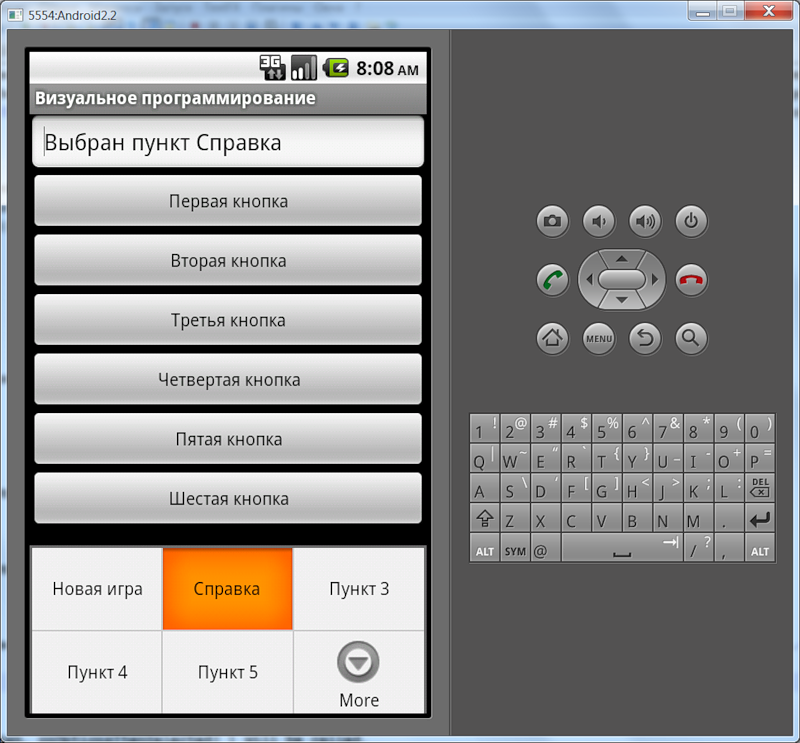
public void showHelp()

{

edtext.setText("Выбран пункт Справка");

}

Запустите приложение, вызовите меню и выберите первый или второй пункт меню. В текстовом поле должно появиться сообщение.



В приведенном примере **getItemId()** запрашивает ID для выбранного пункта меню и начинает сравнивать через оператор выбора switch с идентификаторами, которые мы задали в XML-ресурсах. При обнаружении нужного идентификатора выполняется обработчик для заданного пункта меню. Если программа ничего не обнаружит, то выполняется оператор **default**, который возвращает super class.

В Android 3.0 можно добавить атрибут **android:onClick** в ресурсах меню, и вам уже не нужно использовать **onOptionsItemSelected()**. При помощи **android:onClick** вы можете указать нужный метод при выборе пункта меню.

// у атрибута пункта меню установлено значение android:onClick="onMenuClick"

public void onMenuClick(MenuItem item){

edtext.setText("Выбран пункт Накормить кота");

}

Программное создание меню

Рассмотрим программное создание меню для полноты картины. Нам понадобится определить несколько констант для пунктов меню:

// идентификаторы для пунктов меню

private static final int IDM\_OPEN = 101;

private static final int IDM\_SAVE = 102;

public boolean onCreateOptionsMenu(Menu menu)

{

// добавляем пункты меню

menu.add(Menu.NONE, IDM\_OPEN, Menu.NONE, "Открыть");

menu.add(Menu.NONE, IDM\_SAVE, Menu.NONE, "Сохранить");

}

У метода **add()** есть четыре параметра:

* идентификатор группы - позволяет связывать пункт меню с группой других пунктов этого меню
* идентификатор пункта для обработчика события выбора пункта меню
* порядок расположения пункта в меню - позволяет определять позицию в меню. По умолчанию (Menu.NONE или 0) пункты идут в том порядке, как задано в коде
* заголовок - текст, который выводится в пункте меню. Можно использовать строковый ресурс

Метод возвращает объект **MenuItem**, который можно использовать для установки дополнительных свойств, например, для установить значок, горячую клавишу и т.д.

Если вы хотите создать меню со значками, то воспользуйтесь методом **setIcon()**

menu.add(Menu.NONE, IDM\_OPEN, Menu.NONE, "Открыть")

.setIcon(R.drawable.icon\_menu\_open);

Напомним еще раз, что значки можно добавить только к шести пунктам меню (или к пяти, если пунктов больше шести).

Метод **onCreateOptionsMenu** вызывается системой только один раз при создании меню. Если вам требуется обновить меню во время работы программы, то используйте метод обратного вызова **onPrepareOptionsMenu()**.

При выборе пункта меню вызывается метод **onOptionsItemSelected**, который передает объект **MenuItem** - пункт меню, выбранный пользователем. При помощи метода **getItemId** можно получить идентификатор выбранного пункта меню. После идентификации пункта меню можно написать код для обработки события выбора меню:

public boolean onOptionsItemSelected(MenuItem item)

{

switсh (item.getItemId())

case IDM\_OPEN:

return true;

case IDM\_SAVE:

return true;

return false;

}

Горячие клавиши

Также можно задавать горячие клавиши для быстрого доступа, используя символы клавиатуры, при помощи нескольких методов:

* setAlphabeticShortcut(char) - добавляет символ
* setNumericShortcut(int) - добавляет число
* setShortcut(char, int) - добавляет комбинацию символа и числа

Например, если задать горячую клавишу *setAlphabeticShortcut('q');, то при открытии меню (или при удерживании клавиши MENU) нажатие клавиши****Q****выберет данный пункт меню. Эта горячая клавиша (или сочетание клавиш) будет показана как подсказка, отображающая ниже имени пункта меню. В новых клавиатурах есть отдельная клавиша****Ctrl****, которая работает также, как на обычных клавиатурах.*

Горячие клавиши можно создать и через XML: **android:alphabeticShortcut="c"**.

Обрабатывать нажатия можно через метод активности **onKeyShortcut()**:

@Override

public boolean onKeyShortcut(int keyCode, KeyEvent event) {

switch (keyCode) {

case KeyEvent.KEYCODE\_R:

Toast.makeText(this, "Reply", Toast.LENGTH\_SHORT).show();

return true;

default:

return super.onKeyShortcut(keyCode, event);

}

}

Создание подменю

Подменю можно добавить в любое меню, кроме другого подменю. Подменю создается в методе обратного вызова **onCreateOptionsMenu()** с помощью метода **addSubMenu()**, который возвращает объект **SubMenu**. В объект **SubMenu** можно добавить дополнительные пункты к этому меню, используя метод **add()**. Например:

public static final int IDM\_HELP = 101;

public static final int IDM\_NEW = 201;

public static final int IDM\_OPEN = 202;

public static final int IDM\_SAVE = 203;

public static final int IDM\_CUT = 301;

public static final int IDM\_COPY = 302;

public static final int IDM\_PASTE = 303;

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

SubMenu subMenuFile = menu.addSubMenu("Файл");

subMenuFile.add(Menu.NONE, IDM\_NEW, Menu.NONE, "Новый");

subMenuFile.add(Menu.NONE, IDM\_OPEN, Menu.NONE, "Открыть");

subMenuFile.add(Menu.NONE, IDM\_SAVE, Menu.NONE, "Сохранить");

SubMenu subMenuEdit = menu.addSubMenu("Правка");

subMenuEdit.add(Menu.NONE, IDM\_CUT, Menu.NONE, "Вырезать");

subMenuEdit.add(Menu.NONE, IDM\_COPY, Menu.NONE, "Копировать");

subMenuEdit.add(Menu.NONE, IDM\_PASTE, Menu.NONE, "Вставить");

menu.add(Menu.NONE, IDM\_HELP, Menu.NONE, "Справка");

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

CharSequence message;

switch (item.getItemId()) {

case IDM\_NEW:

message = "Выбран пункт Новый";

break;

case IDM\_OPEN:

message = "Выбран пункт Открыть";

break;

case IDM\_SAVE:

message = "Выбран пункт Сохранить";

break;

case IDM\_CUT:

message = "Выбран пункт Вырезать";

break;

case IDM\_COPY:

message = "Выбран пункт Копировать";

break;

case IDM\_PASTE:

message = "Выбран пункт Вставить";

break;

case IDM\_HELP:

message = "Выбран пункт Справка";

break;

default:

return false;

}

// выводим уведомление о выбранном пункте меню

Toast toast = Toast.makeText(this, message, Toast.LENGTH\_LONG);

toast.setGravity(Gravity.CENTER, 0, 0);

toast.show();

return true;

}

Теперь при выборе пункта меню появится еще одно окно с подменю. Попробуйте сами.

Добавление флажков и переключателей

В пункты меню возможно добавление флажков или переключателей. Чтобы добавить флажок или переключатель для отдельного элемента меню, необходимо использовать метод **setCheckable()**:

MenuItem item = menu.add(0, IDM\_FORMAT\_BOLD, 0, "Bold");

item.setCheckable(true);

Если есть необходимость добавить несколько пунктов меню с флажками или переключателями, то можно объединить их в группы меню, создав отдельный идентификатор. Пункт меню добавляется в группу через метод **add()**, передав ему в качестве первого параметра идентификатор группы меню. Допустим, мы объявили идентификаторы для группы меню Цвет и элементов меню для установки цвета:

public static final int IDM\_COLOR\_GROUP = 400;

public static final int IDM\_COLOR\_RED = 401;

public static final int IDM\_COLOR\_GREEN = 402;

public static final int IDM\_COLOR\_BLUE = 403;

Теперь для создания группы меню с флажками нужно назначить идентификатор группы на каждый пункт меню и вызвать метод **setGroupCheckable()** для всей группы (этом случае нет необходимости вызывать метод **setCheckable()** для каждого пункта меню):

SubMenu subMenuColor = menu.addSubMenu("Цвет");

subMenuColor.add(IDM\_COLOR\_GROUP, IDM\_COLOR\_RED, Menu.NONE, "Красный");

subMenuColor.add(IDM\_COLOR\_GROUP, IDM\_COLOR\_GREEN, Menu.NONE,"Зеленый");

subMenuColor.add(IDM\_COLOR\_GROUP, IDM\_COLOR\_BLUE, Menu.NONE, "Синий");

subMenuColor.setGroupCheckable(IDM\_COLOR\_GROUP, true, false);

У метода **setGroupCheckable()** три параметра:

* первый параметр — идентификатор группы меню;
* второй параметр — true, если в группе разрешены переключатели или флажки;
* третий параметр — устанавливает единственный (true) или множественный (false) выбор пунктов меню. Этот параметр фактически определяет внешний вид меню — это будет меню с переключателями или флажками.

Для управления состоянием флажков и переключателей в обработчике события выбора пункта меню нужно написать следующее:

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

CharSequence message;

switch (item.getItemId()) {

...

case IDM\_COLOR\_RED:

// инвертируем состояние флажка

item.setChecked(!item.isChecked());

message = "Красный цвет";

break;

default:

return false;

}

Запустите проект, вызовите меню и выберите пункт меню **Цвет**. У вас появится подменю с тремя пунктами (Красный, Зеленый, Синий) в виде флажков. Состояние флажков и переключателей обрабатывается в коде программы и сохраняется при повторных вызовах меню.

Можно сразу назначить намерение выбранному пункту меню через метод setIntent(), которое сработает при нажатии этого пункта, если данное событие не было перехвачено обработчиками onMenuItemClickListener (устар.) или onOptionsItemSelected. Сработав, намерение передается в метод startActivity.

menuItem.setIntent(new Intent(this, MyOtherActivity.class));

Программное открытие или закрытие меню

Если вам по каким-то причинам нужно программно открыть меню (например, в демонстрационных целях), то используйте метод **openOptionsMenu()**:

openOptionsMenu();

Для программного закрытия меню используйте метод **closeOptionsMenu()**, впрочем у меня повторный вызов метода **openOptionsMenu()**также закрывает меню.

Программное удаление пункта меню

Допустим, мы определили пункт меню в xml-файле:

<item

android:id="@+id/action\_dog"

android:orderInCategory="100"

android:showAsAction="never"

android:title="Пёсик"/>

Чтобы удалить явно лишний пункт меню из нашей программы о котах, нужно получить доступ к пункту меню через метод **findItem()** и сделать его невидимым. Ссылку на объект Menu нужно передать в метод **onCreateOptionsMenu**, чтобы программа узнала об изменении состава меню.

// переменная класса

Menu menu;

@Override

public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

// передаём ссылку на наш объект

this.menu = menu;

getMenuInflater().inflate(R.menu.test, menu);

return true;

}

// щелчок кнопки

public void onClick(View v) {

if (menu != null) {

// находим нужный элемент

MenuItem item\_dog = menu.findItem(R.id.action\_dog);

// делаем его невидимым

item\_dog.setVisible(false);

}

}

Но у данного решения есть недостаток, если мы повернём экран, то активность пересоздатся и удалённое меню снова появится. Как же нам избавиться от сранного пёсика?

Надо запомнить состояние пункта меню и сохранить его в объекте типа Bundle в методе **onSaveInstanceState**, а в методе **onCreate()** извлечь сохранённое состояние и передать методу **onPrepareOptionsMenu**, который вызывается перед показом меню на экране:

package ru.alexanderklimov.test;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

public class TestActivity extends Activity {

Menu menu;

Boolean savedMenuDogIsVisible;

final static String KEY\_MENU\_DOG = "KEY\_MENU\_DOG";

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

// извлекаем данные о видимости пункта меню

if (savedInstanceState != null) {

savedMenuDogIsVisible = savedInstanceState.getBoolean(KEY\_MENU\_DOG,

true);

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

this.menu = menu;

getMenuInflater().inflate(R.menu.test, menu);

return true;

}

public void onClick(View v) {

if (menu != null) {

MenuItem item\_dog = menu.findItem(R.id.action\_dog);

// прячем пункт меню

item\_dog.setVisible(false);

}

}

@Override

protected void onSaveInstanceState(Bundle outState) {

// TODO Auto-generated method stub

super.onSaveInstanceState(outState);

if (menu != null) {

MenuItem item\_dog = menu.findItem(R.id.action\_dog);

// сохраняем текущее состояние пункта меню - true или false

outState.putBoolean(KEY\_MENU\_DOG, item\_dog.isVisible());

}

}

@Override

public boolean onPrepareOptionsMenu(Menu menu) {

if (savedMenuDogIsVisible != null) {

MenuItem item\_dog = menu.findItem(R.id.action\_dog);

// перед выводом на экран узнаём нужное состоятние пункта меню

item\_dog.setVisible(savedMenuDogIsVisible);

}

return super.onPrepareOptionsMenu(menu);

}

}

Определить наличие кнопки Menu

На старых устройствах использовалась реальная кнопка Menu. В новых версиях Android меню убрали в ActionBar и её наличие в виде отдельной кнопки стало необязательным. Но многие производители по-прежнему выпускают телефоны с кнопкой для меню. Чтобы определить, есть ли такая кнопка, в Android 14 добавили новый метод, который позволит определить наличие этой кнопки.

if (Build.VERSION.SDK\_INT <= 10

|| (Build.VERSION.SDK\_INT >= 14 && ViewConfiguration.get(this)

.hasPermanentMenuKey())) {

// menu key is present

Toast.makeText(this, "Кнопка Menu есть", Toast.LENGTH\_LONG).show();

} else {

// No menu key

Toast.makeText(this, "Кнопки Menu нет", Toast.LENGTH\_LONG).show();

}

Разметка для меню

В современных устройствах меню является частью **ActionBar**. И вы можете настроить разметку меню через XML.

Допустим, вы выбрали такой вариант:

<item

android:id="@+id/action\_new"

android:actionLayout="@layout/action\_layout"

android:orderInCategory="100"

android:showAsAction="always|withText"

android:title="Новый"/>

В атрибуте **showAsAction** не используйте значение **never**, иначе разметку не увидите. Сама разметка задана через атрибут **actionLayout**. Код для разметки:

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

<LinearLayout 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:orientation="horizontal" >

<ImageButton

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:contentDescription="Custom button!"

android:scaleType="fitCenter"

android:src="@drawable/blue" />

</LinearLayout>

Меню в фрагментах

Меню может быть не только частью активности, но и частью фрагмента. Принцип работы практически не отличается. У фрагмента есть соответствующий метод.

@Override

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

super.onCreateOptionsMenu(menu, inflater);

inflater.inflate(R.menu.fragment\_crime\_list, menu);

}

**FragmentManager** отвечает за вызов **onCreateOptionsMenu()** при получении активностью обратного вызова **onCreateOptionsMenu()** от системы. Вы должны явно сообщить менеджеру **FragmentManager**, что фрагмент должен получить вызов **onCreateOptionsMenu()**. Для этого вызывается метод **setHasOptionsMenu()**:

// В коде фрагмента

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setHasOptionsMenu(true);

getActivity().setTitle(R.string.cat);

...

}