**ОГЛАВЛЕНИЕ**

## ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является приобретение практических навыков создания графических интерфейсов на основе различных виджетов и меню.

Основными задачами выполнения лабораторной работы являются:

1. Научиться использовать графический и текстовый режимы для создания интерфейса приложения
2. Изучить особенности реализации обработчиков событий
3. Научиться реализовывать логику работы приложения с учетом специфики платформы Android
4. Понять логику работы приложения с меню с учетом специфики платформы Android
5. Разработать и подключить разные типы меню с учетом аппаратных ограничений мобильных устройств
6. Понять особенности реализации обработчиков пунктов меню

Результатами работы являются:

- Приложение, иллюстрирующее навыки разработки графического интерфейса для системы Android согласно варианту задания.
- Подготовленный отчет.

# ФОРМИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

## Компоновка элементов управления

Компоновка – это архитектура расположения элементов интерфейса пользователя для конкретного окна, представляющего Activity. Компоновка определяет структуру расположения элементов в окне и содержит все элементы, которые предоставляются пользователю программы.

Эта важная тема, поскольку проектирование пользовательского интерфейса для мобильных телефонов сложнее, чем для настольных систем или для Web-страниц. Экраны мобильных телефонов имеют гораздо меньшее разрешение, чем обычные мониторы. Кроме того, существует много разновидностей дисплеев для мобильных телефонов, отличающихся размерами, разрешением и плотностью пикселов.

Необходимо также учесть, что большинство экранов для мобильных телефонов сенсорные, причем могут быть разного типа. Например, емкостный экран реагирует на касание пальцем, а для взаимодействия с резистивным экраном используется стилус. Поэтому важно правильно задавать компоновку и размеры элементов управления, чтобы пользователю было удобно управлять вашим приложением независимо от типа экрана.

## Иерархия элементов компоновки

В Android-приложении графический интерфейс пользователя формируется с использованием объектов View и ViewGroup. Класс View является базовым классом для ViewGroup и состоит из коллекции объектов View (рисунок 1). Есть множество типов View и ViewGroup, каждый из которых является потомком класса View.

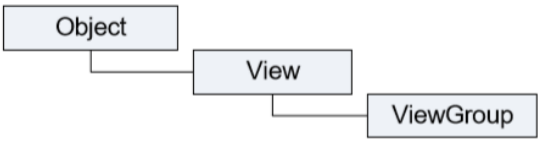


**Рис. 1.** Иерархия классов View и ViewGroup

Объекты View — это основные модули для создания графического интерфейса пользователя на платформе Android. Класс View служит базовым для классов элементов управления, называемых виджетами, — текстовых полей, кнопок и т. д.

Объект View — структура, свойства которой сохраняют параметры компоновки и содержание для определенной прямоугольной области экрана. Как объект в интерфейсе пользователя объект View является точкой взаимодействия пользователя и программы.

Класс ViewGroup представляет собой контейнер, который служит ядром для подклассов, называемых компоновками (layouts). Эти классы формируют расположение элементов пользовательского интерфейса на форме и содержат дочерние элементы View или ViewGroup.

На платформе Android необходимо определить пользовательский интерфейс для каждого Activity, используя иерархии узлов View и ViewGroup, как показано на рисунке 2. Это дерево иерархии может быть и простым, и сложным — в зависимости от требований к графическому интерфейсу приложения.

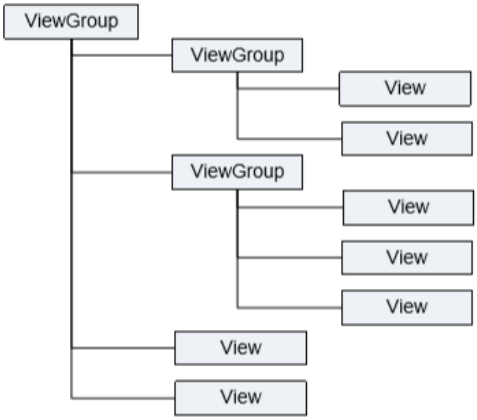


**Рис. 2.** Пример дерева узлов View и ViewGroup для Activity

При запуске программы система Android получает ссылку на корневой узел дерева и использует ее для прорисовки графического интерфейса на экране мобильного устройства. Система также анализирует элементы дерева от вершины дерева иерархии,

прорисовывая дочерние объекты View и ViewGroup и добавляя их родительским элементам. Для этого в методе onCreate() необходимо вызвать метод setContentView(), передав ему в качестве параметра ссылку на ресурс компоновки в следующем виде:

```
R.layout.layout_file_name
```

Например, если компоновка находится в файле main.xml, ее загрузка в методе onCreate() происходит так:

```
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)
  }
```

Для создания окон существует несколько типов компоновок, которые вы можете использовать в создаваемых приложениях:

— **LinearLayout** — отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical)

— **TableLayout** — отображает элементы в виде таблицы, по строкам и столбцам

— **RelativeLayout** — для каждого элемента настраивается его положение относительно других элементов

— **AbsoluteLayout** — для каждого элемента указывается явная позиция на экране в системе координат (x,y)

Прорисовка начинается с корневого узла дерева компоновки. Затем последовательно прорисовываются дочерние объекты дерева компоновки. Это означает, что родители будут прорисовываться раньше, чем их дочерние объекты, — т. е. по окончании процесса прорисовки родители будут находиться на заднем плане по отношению к дочерним узлам.

**Создание компоновки**

Компоновку можно объявлять несколькими способами:

1. Объявить элементы пользовательского интерфейса в XML-файле. Android обеспечивает прямой XML-словарь, который соответствует классам View и ViewGroup;
2. Создать разметку для окна в коде программы во время выполнения — инициализировать объекты Layout и дочерние объекты View, ViewGroup и управлять их свойствами программно.

При создании пользовательского интерфейса можно использовать каждый из этих методов в отдельности или оба сразу для объявления и управления пользовательским интерфейсом в приложении. Например, можно объявить заданную по умолчанию компоновку окна вашего приложения в XML-файле, включая экранные элементы, которые появятся в них, и их свойства, а затем добавить код в приложение, который во время выполнения изменит состояние объектов на экране, включая объявленные в XML-файле.

Android Studio включает в себя специальный редактор для создания разметки двумя способами. Редактор имеет две вкладки: одна позволяет увидеть, как будут отображаться элементы управления, а вторая – создавать XML-разметку вручную.

Создавая пользовательский интерфейс в XML-файле, вы можете отделить представление приложения от программного кода. Вы можете изменять пользовательский интерфейс в файле разметки без необходимости изменения вашего программного кода. Например, вы можете создавать XML-разметки для различных ориентаций экрана мобильного устройства (portrait, landscape), размеров экрана и языков интерфейса.

Создание интерфейса в коде java не позволяет достичь разграничение интерфейса и логики приложения, что усложняет разработку и модификацию кода. Наиболее предпочтительным является способ определения интерфейса в файле xml.

**XML-файл компоновки**

XML предлагает удобную структуру для компоновки, похожую на HTML-компоновку Web-страницы. О преимуществе такого подхода

уже было упомянуто выше - объявление пользовательского интерфейса в XML-файле дает возможность отделить дизайн приложения от программного кода, который управляет поведением приложения.

Используя XML-словарь Android, можно быстро проектировать пользовательский интерфейс компоновки и экранные элементы, которые он содержит, тем же самым способом, которым вы создаете Web-страницы в HTML — с рядом вложенных элементов.

Каждый файл компоновки должен содержать только один корневой элемент, который должен быть объектом View или ViewGroup. Как только вы определили корневой элемент, вы можете добавить дополнительные объекты компоновки или виджеты как дочерние элементы, чтобы постепенно формировать иерархию элементов, которую определяет создаваемая компоновка.

Файлы разметки графического интерфейса располагаются в проекте в каталоге res/layout. По умолчанию при создании проекта уже есть один файл ресурсов разметки activity_main.xml, который может выглядеть примерно так:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"

tools:context="com.example.eugene.viewsapplication.MainActivity">

 <TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

```
    app:layout_constraintTop_toTopOf="parent" />
 </android.support.constraint.ConstraintLayout>
```

В файле определяются все графические элементы и их атрибуты, которые составляют интерфейс. При создании разметки в XML следует соблюдать некоторые правила: каждый файл разметки должен содержать один корневой элемент, который должен представлять объект View или ViewGroup. В данном случае корневым элементом является элемент ConstraintLayout, который содержит элемент TextView.

При компиляции каждый XML-файл разметки компилируется в ресурс View. Загрузка ресурса разметки осуществляется в методе Activity.onCreate. Чтобы установить разметку для текущего объекта activity, надо в метод setContentView в качестве параметра передать ссылку на ресурс разметки.

Название ресурса layout будет совпадать с именем файла, поэтому чтобы использовать файл activity_main.xml в качестве источника визуального интерфейса, нам надо изменить код MainActivity следующим образом:

```
class MainActivity : AppCompatActivity() {

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)
        }
    }
```

**Графические возможности в Android Studio**

Android Studio имеет довольно продвинутый инструментарий, который облегчает разработку графического интерфейса. Мы можем открыть файл activity_main.xml и внизу с помощью кнопки Design переключиться в режим дизайнера к графическому представлению интерфейсу в виде эскиза смартфона. (рисунок 3).

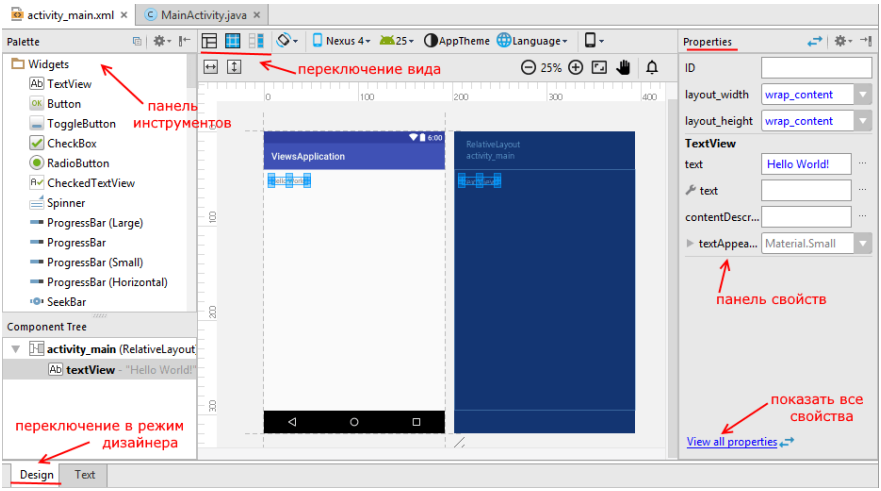


**Рис. 3.** Редактор в режиме дизайнера

Слева будет находиться панель инструментов, с которой мы можем переносить нужный элемент мышкой на эскиз смартфона. И все перенесенные элементы будут автоматически добавляться в файл activity_main.xml. С помощью мыши мы можем изменять позиционирование уже добавленных элементов, перенося их в другое место на смартфоне.

Справа будет окно Properties - панель свойств выделенного элемента. Здесь мы можем изменить значения свойств элемента. И опять же после изменения свойств изменится и содержимое файла activity_main.xml. То есть при любых изменениях в режиме дизайнера будет происходить синхронизация с файлом activity_main.xml. Все равно, что мы вручную изменяли бы код непосредственно в файле activity_main.xml.

Но даже если мы предпочитаем работать с разметкой интерфейса в текстовом виде, то даже здесь мы можем включить предварительный просмотр для файла activity_main.xml. Для этого после переключения в текстовый режим необходимо нажать на вкладку Preview справа в Android Studio (рисунок 4).

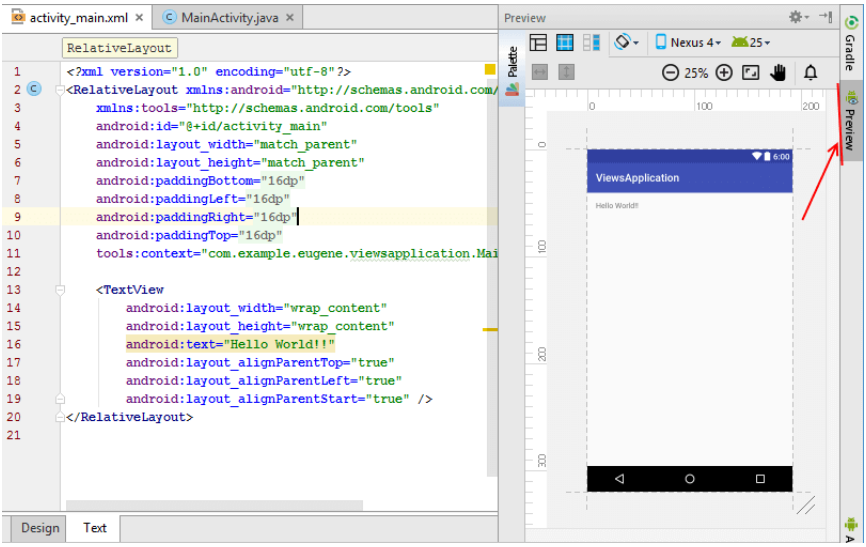


**Рис. 4.** Предварительный просмотр интерфейса в текстовом режиме

Это очень удобно, так как сразу позволяет просмотреть, как будет выглядеть приложение. А при любых изменениях область предварительного просмотра будет автоматически синхронизироваться с содержанием файла activity_main.xml.

**Виджеты**

Виджет — это объект View, который служит интерфейсом для взаимодействия с пользователем. Говоря простым языком, виджеты — это элементы управления. Android обеспечивает набор готовых виджетов, таких как кнопки, переключатели и текстовые поля, с помощью которых можно быстро сформировать пользовательский интерфейс приложения.

Стандартные элементы имеют привычные свойства: ширина, высота, цвет и тому подобные. Еще два важных свойства, которые могут влиять на размер и положение дочерних элементов - важность (weight) и выравнивание (gravity). Weight используется для присвоения элементу показателя важности, отличающего его от других элементов, находящихся в контейнере. Предположим, в контейнере находится три

элемента управления: первый имеет важность 1 (максимальное возможное значение), а два других имеют значение 0. В этом случае элемент управления, который имеет значение важности 1, займет в контейнере все свободное пространство. Gravity – это ориентация в контейнере. Например, необходимо выровнять текст надписи по правому краю, тогда свойство gravity будет иметь значение right. Набор значений для gravity ограничен: left, center, right, top, bottom, center_vertical и еще некоторые.

**TextView**

Виджет TextView предназначен для отображения текста без возможности редактирования его пользователем. TextView один из самых используемых виджетов. С его помощью пользователю удобнее ориентироваться в программе. По сути, TextView служит для представления пользователю описательного текста.

Кроме того, элемент TextView используется как элемент для отображения текстовых данных в контейнерных виджетах для отображения списков. От класса TextView наследуется множество других виджетов: кнопки, флажки и переключатели — элементы управления, на которых может быть отображен текст.

**EditText**

Элемент EditText — это текстовое поле для пользовательского ввода. EditText представляет собой тонкую оболочку над классом TextView, которая сконфигурирована для редактирования вводимого текста.

Основной метод класса EditText — getText(), который возвращает текст, содержащийся в окне элемента EditText. Возвращаемое значение имеет тип Editable. Этот тип представляет собой интерфейс для текста, информационное наполнение которого может изменяться.

**Button**

Кнопка – один из самых распространенных элементов управления в программировании. Наследуется от Textview и является базовым классом для класса CompoundButton. От класса CompoundButton, в

свою очередь, наследуются такие элементы как CheckBox, ToggleButton и RadioButton. На кнопке располагается текст и на кнопку нужно нажать, чтобы получить результат.

**Другие типы виджетов**

— **CheckBox** является флажком, с помощью которого пользователь может отметить (поставить галочку) определенную опцию. Очень часто флажки используются в настройках, когда нужно выборочно выбрать определенные пункты, необходимые для комфортной работы пользователю.

— **RadioButton**. Главная особенность элемента RadioButton состоит в том, что он не используется в одиночестве. Всегда должно быть два и более переключателя и только один из них может быть выбранным.

— **Switch** еще один вид переключателей, представляет собой полоску с двумя состояниями, переключиться между которыми можно сдвиганием ползунка.

— **Spinner** похож на выпадающий список. В закрытом состоянии элемент показывает одну строчку, при раскрытии выводит список в виде диалогового окна с переключателями.

— **ProgressBar** (индикатор прогресса) применяется в тех случаях, когда пользователю нужно показать, что программа не зависла, а выполняет продолжительную работу.

Это далеко не полный список виджетов предлагаемый android.

# ОБРАБОТКА СОБЫТИЙ

После добавления виджетов в пользовательский интерфейс, вам потребуется организовать взаимодействие виджетов с пользователем. Для этого необходимо определить обработчик события и зарегистрировать его для данного элемента.

Класс View содержит коллекцию вложенных интерфейсов, которые называются OnListener(), в каждом из которых объявлен единственный абстрактный метод. Этот метод необходимо переопределить в вашем классе. Его будет вызывать система Android, когда с экземпляром View, к которому был подсоединен слушатель события, будет взаимодействовать с пользователем.

Всего класс View содержит шесть вложенных интерфейсов:

— OnClickListener
— OnLongClickListener
— OnFocusChangeListener
— OnKeyListener
— OnTouchListener
— OnCreateContextMenuListener

Например, если требуется, чтобы кнопка получила уведомление о нажатии ее пользователем, необходимо в классе окна реализовать интерфейс OnClickListener и определить его метод обратного вызова onClick(), куда будет помещен код обработки нажатия кнопки, и зарегистрировать слушатель события с помощью метода setOnClickListener():

```
button1.setOnClickListener(object : View.OnClickListener() {
        fun onClick(v: View) {
            mText.setText("Click on First Button")
        }
    })
```

Существует несколько способов подключения событий. Рассмотрим эти способы на примере обработки события нажатия на кнопку.

**Первый способ - атрибут onClick**

Относительно новый способ, специально разработанный для Android - использовать атрибут onClick.

```
android:onClick="onMyButtonClick"
```

Имя для события можно выбрать произвольное. Далее нужно прописать в классе активности придуманное вами имя метода, который будет обрабатывать нажатие. Метод должен быть открытым (public) и с одним параметром, использующим объект View.

```
fun onMyButtonClick(View view)
   {
      // выводим сообщение
      Toast.makeText(this, "Нажатие на кнопку",
                  Toast.LENGTH_SHORT).show();
   }
```

Когда пользователь нажимает на кнопку, то вызывается метод onMyButtonClick(), который в свою очередь генерирует всплывающее сообщение (рисунок 5).


**Рис. 5.** Обработка события нажатия на кнопку

Данный способ применим не только к кнопке, но и к другим элементам и позволяет сократить количество строк кода.

**Второй способ - метод setOnClickListener()**

Более традиционный способ в Java - через метод setOnClickListener(), который прослушивает нажатия на кнопку.

Предположим, у вас на экране уже есть кнопка button. В коде вы объявляете её обычным способом:

```
val button = findViewById<View>(R.id.button) as Button
```

Следующий шаг - написание метода для нажатия:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    fun onClick(View v) {
    }
});
```

Теперь у вас есть рабочая заготовка и сразу внутри фигурных скобок метода onClick() вы можете писать свой код.

**Третий способ - интерфейс OnClickListener**

Третий способ является родственным второму способу и также является традиционным для Java. Кнопка присваивает себе обработчика с помощью метода setOnClickListener (View.OnClickListener l), т.е. подойдет любой объект с интерфейсом View.OnClickListener. Мы можем указать, что наш класс Activity будет использовать интерфейс View.OnClickListener.

После слов extends Activity дописываем слова implements OnClickListener

```
class ButtonActivity : Activity(), View.OnClickListener {

}
```

После этого, заготовка для нажатия кнопки будет сгенерирована автоматически.

```
fun onClick(v: View) {

    }
```

Метод будет реализован не в отдельном объекте-обработчике, а в Activity, который и будет выступать обработчиком. В методе onCreate() присвоим обработчик кнопке. Это будет объект this, т.е. текущий объект нашей активности.

```
button.setOnClickListener(this)
```

Первый способ является самым простым и понятным. Однако использование второго и третьего способа дадут вам представление, как писать обработчики для других событий, так как кнопка может иметь и другие события. Например, кроме обычного нажатия существует долгое нажатие на кнопку (long click).

# МЕНЮ

Меню — важная часть любого приложения. Система Android предлагает простой интерфейс программирования для создания стандартизированных прикладных меню для приложений с разнообразной функциональностью.

В ранних версиях Android меню появлялось при нажатии клавиши <MENU> на мобильном устройстве. При этом меню выбора опций (Options Menu) имело максимум 6 пунктов. При наличии более 6 пунктов появлялось расширенное меню (Expanded Menu). В этом случае вместо шестого пункта появлялся пункт Опции (More). Начиная с Android 3.0 (API level 11) устройства на базе Android больше не обязаны предоставлять специально выделенную кнопку для меню. С таким изменением приложения Android должны уйти от зависимости к традиционной панели меню с 6 элементами, и вместо этого предоставить строку меню (action bar), чтобы в ней указывать общие возможные действия для пользователя.

Хотя дизайн (касаемо опыта пользователя) для некоторых элементов меню изменился, семантика для определения набора действий все еще основывается на функциях Menu API. Существует 3 разновидности типа меню, которые мы рассмотрим далее.

**Меню выбора опций**

Меню выбора опций — наиболее распространенный тип меню в приложениях, является главной коллекцией элементов меню для экземпляра класса Activity (этот класс олицетворяет окно программы, его вид и поведение). Меню опций - именно то место, где располагаются действия, влияющие на приложение в целом, такие как "Search" (поиск), "Compose email" (составить письмо) и "Settings" (настройки).

Если Вы разрабатываете приложение для Android 2.3 или более старой версии, пользователи могут отобразить панель меню путем нажатия кнопки Menu.

На Android 3.0 и более новых версиях меню опций (options menu) доступно в полосе action bar в виде предложения действий, перекрывающих на экране основное окно программы.

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).

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.

**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.

Вы можете использовать встроенные системные значки Android. Например, android:icon="@android:drawable/ic_menu_help" позволит вам вывести значок помощи, который зашит в систему.

При создании меню мы указали на строковые ресурсы @string/new_game и @string/help. Для того, чтобы студия понимала о чем идет речь, 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().

```
fun onCreateOptionsMenu(menu: Menu): Boolean {
        val inflater = menuInflater
        inflater.inflate(R.menu.game_menu, menu)
        return true
    }
```

При этом необходимо импортировать недостающие пространства имен.

```
import android.view.Menu;
import android.view.MenuInflater;
```

Если ваше приложение предназначено для версии Android 2.3.x или более ранней, система вызывает метод onCreateOptionsMenu() для создания меню параметров, когда пользователь открывает это меню впервые при нажатии клавиши <MENU>. Если приложение предназначено для версии Android 3.0 и или более поздней, система вызывает метод onCreateOptionsMenu() при запуске операции, чтобы отобразить пункты в action bar (рисунок 6).

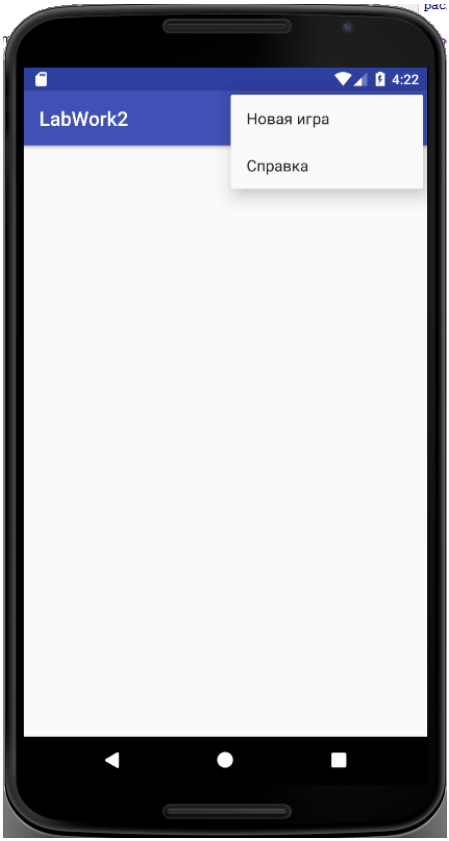


**Рис. 6.** Меню выбора опций

**Обработка нажатий пунктов меню**

Созданное меню 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 определить выбранный пункт через вызов getItemId(), который возвращает идентификатор пункта меню. Далее через оператор switch необходимо определить нужные команды. Для вывода информации воспользуемся текстовой меткой. Добавьте на экран активности компонент TextView. Можете использовать имеющийся TextView с надписью "Hello World!".

```
fun onOptionsItemSelected(item: MenuItem): Boolean {
    val id = item.getItemId()
    val infoTextView = findViewById(R.id.textView) as TextView
    // Операции для выбранного пункта меню
    when (id) {
        R.id.new_game -> {
            infoTextView.text = "Выбран пункт Новая игра"
            return true
        }
        R.id.help -> {
            infoTextView.text = "Выбран пункт Справка"
            return true
        }
        else -> return super.onOptionsItemSelected(item)
    }
}
```

Запустите приложение, вызовите меню и выберите первый или второй пункт меню. В текстовом поле должно появиться соответствующее сообщение (рисунок 7).


**Рис. 7.** Обработка нажатия пунктов меню

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 и уже в данном методе производить обработку нажатия.

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

fun onMenuClick(item: MenuItem) {
    infoTextView.setText("Выбран пункт меню")
}
```

Сохраняем, запускаем и проверяем. Теперь при выборе первого пункта меню выводится соответствующий текст в текстовом поле (рисунок 8).


**Рис. 8.** Обработка нажатия пунктов меню при помощи атрибута

**Контекстное меню**

Контекстное меню в Android напоминает контекстное меню в настольных системах, появляющееся при нажатии правой кнопки мыши. Меню вызывается при нажатии на объект в течение двух секунд (событие long-press).

В отличие от обычного меню, в контекстном меню не поддерживаются значки и быстрые клавиши. Второе важно отличие - контекстное меню применимо к виду, а меню к активности. Поэтому в

приложении может быть одно меню и несколько контекстных меню, например, у каждого элемента TextView.

Существует два способа предоставления возможности контекстных действий:

**В плавающем контекстном меню**. Меню отображается в виде плавающего списка пунктов меню (наподобие диалогового окна), когда пользователь длительно нажимает на экран (нажимает и удерживает нажатым) в представлении, которое объявляет поддержку контекстного меню. Пользователи могут каждый раз выполнять контекстное действие только с одним элементом.

**В режиме контекстных действий**. Этот режим является системной реализацией ActionMode, которая отображает строку контекстных действий вверху экрана с пунктами действий, которые затрагивают выбранные элементы. Когда этот режим активен, пользователи могут одновременно выполнять действие с несколькими элементами (если это допускается приложением). Режим контекстных действий поддерживается в версии Android 3.0 (уровень API 11) и последующих версиях.

Рассмотрим создание плавающего контекстного меню.

Откроем main.xml и нарисуем там два TextView:

```
<?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:layout_height="wrap_content"
      android:textSize="26sp"
      android:layout_width="wrap_content"
      android:id="@+id/tvColor"
      android:layout_marginBottom="50dp"
      android:layout_marginTop="50dp"
      android:text="Text color">
```

```
    </TextView>
    <TextView
        android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        android:textSize="22sp"
        android:id="@+id/tvSize"
        android:text="Text size">
    </TextView>
</LinearLayout>
```

Для первого TextView мы сделаем контекстное меню, с помощью которого будем менять цвет текста. Для второго – будем менять размер текста.

Принцип создания контекстного меню похож на создание обычного меню. Но есть и отличия.

Метод создания onCreateContextMenu вызывается каждый раз перед показом меню. На вход ему передается:

— **ContextMenu**, в который мы будем добавлять пункты

— **View** – элемент экрана, для которого вызвано контекстное меню

— **ContextMenu.ContextMenuInfo** – содержит дополнительную информацию, когда контекстное меню вызвано для элемента списка.

Метод обработки onContextItemSelected аналогичный методу onOptionsItemSelected для обычного меню. На вход передается MenuItem – пункт меню, который был нажат.

Также необходим третий метод registerForContextMenu. На вход ему передается View и это означает, что для этой View необходимо создавать контекстное меню. Если не выполнить этот метод, контекстное меню для View создаваться не будет.

В MainActivity.java. опишем и найдем TextView и укажем, что необходимо создавать для них контекстное меню.

```
var tvColor: TextView
    var tvSize:TextView

    public override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        tvColor = findViewById(R.id.tvColor) as TextView
        tvSize = findViewById(R.id.tvSize) as TextView

        // для tvColor и tvSize необходимо создавать контекстное меню
        registerForContextMenu(tvColor)
        registerForContextMenu(tvSize)
    }
```

Теперь опишем создание контекстных меню. Используем константы для хранения ID пунктов меню.

```
    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
```

И создаем

```
    override fun onCreateContextMenu(
        menu: ContextMenu, v: View,
        menuInfo: ContextMenuInfo
    ) {
        when (v.id) {
            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")
        }
    }
}
```

Здесь по ID определяем View, для которого вызвано контекстное меню и в зависимости от этого создаем определенное меню. Т.е. если контекстное меню вызвано для tvColor, то мы создаем меню с перечислением цветов, а если для tvSize – с размерами шрифта.

Обработка при нажатии:

```
fun onContextItemSelected(item: MenuItem): Boolean {
    // TODO Auto-generated method stub
    when (item.getItemId()) {
        // пункты меню для tvColor
        MENU_COLOR_RED -> {
            tvColor.setTextColor(Color.RED)
            tvColor.text = "Text color = red"
        }
        MENU_COLOR_GREEN -> {
            tvColor.setTextColor(Color.GREEN)
            tvColor.text = "Text color = green"
        }
        MENU_COLOR_BLUE -> {
            tvColor.setTextColor(Color.BLUE)
            tvColor.text = "Text color = blue"
        }
        // пункты меню для tvSize
        MENU_SIZE_22 -> {
            tvSize.setTextSize(22)
            tvSize.text = "Text size = 22"
        }
        MENU_SIZE_26 -> {
```

```
            tvSize.setTextSize(26)
            tvSize.text = "Text size = 26"
        }
        MENU_SIZE_30 -> {
            tvSize.setTextSize(30)
            tvSize.text = "Text size = 30"
        }
    }
    return super.onContextItemSelected(item)
}
```

В этом методе мы определяем по ID, какой пункт меню был нажат. И выполняем соответствующие действия: меняем цвет текста для tvColor или размер шрифта для tvSize. Сохраняем, запускаем и проверяем, что контекстные меню теперь реагируют на нажатия и делают то, что от них требуется (рисунок 9).

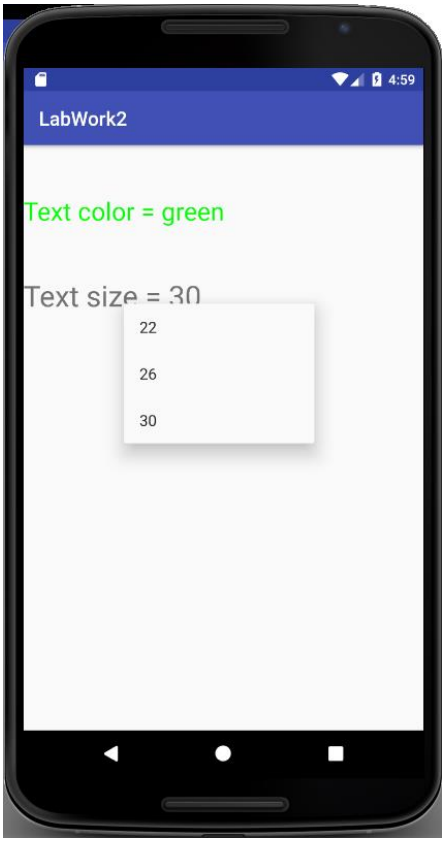


**Рис. 9.** Контекстное меню

**Всплывающее меню (Popup menu)**

Начиная с Android 3.0, в системе появилась возможность создавать всплывающее меню, привязанное к элементу View. Меню реализовано в виде модального окна, которое отображается снизу от родителя меню или в другом месте, если места снизу недостаточно. PopupMenu не нужно путать с контекстным меню. У них разные задачи, хотя поведение весьма схоже. В новых версиях Android использование всплывающих меню предпочтительнее контекстных, которые можно считать устаревшим интерфейсом.

В Android 4.0 добавили новую функциональность, чтобы работать было проще. В частности, всплывающее меню можно получить из

XML-файла, используя метод inflate(int), которому следует передать идентификатор ресурса меню. А до этого приходилось использовать отдельный класс MenuInflator с избыточным кодом.

Также появился слушатель PopupMenu.OnDismissListener для работы с закрытием меню. Он срабатывает либо, когда пользователь щёлкает на пункте меню и меню закрывается, либо пользователь щёлкает в другом месте экрана, и меню также закрывается.

Создать всплывающее меню очень просто. По сути мы повторяем шаги по созданию обычного меню. Сначала в ресурсах меню создадим нужный файл res/menu/popupmenu.xml:

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

  <group android:id="@+id/menugroup1" >
    <item
      android:id="@+id/menu1"
      android:title="Popup menu item 1"/>
    <item
      android:id="@+id/menu2"
      android:title="Popup menu item 2"/>
    <item
      android:id="@+id/menu3"
      android:title="Popup menu item 3">
      <menu>
        <item
          android:id="@+id/submenu"
          android:title="Подменю"/>
      </menu>
    </item>
  </group>
  <group android:id="@+id/menugroup2" >
    <item
      android:id="@+id/menu4"
      android:checkable="true"
```

```
            android:checked="true"
            android:title="Popup menu item 4"/>
        <item
            android:id="@+id/menu5"
            android:title="Popup menu item 5"
            android:enabled="false"/>
        <item
            android:id="@+id/menu6"
            android:title="Popup menu item 6"/>
    </group>
</menu>
```

Обратите внимание как в примере создано подменю и использованы атрибуты доступности.

Далее добавим на экран активности кнопку. При щелчке на которую мы будем выводить всплывающее меню:

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

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Показать меню" />
</LinearLayout>
```

Осталось написать код. Обратите внимание на закомментированный код, который работал в Android 3.0. В Android 4.0 этот код можно не использовать.

```
class MainActivity : ActionBarActivity() {

    internal var viewClickListener: View.OnClickListener = object :
View.OnClickListener() {
        fun onClick(v: View) {
            showPopupMenu(v)
        }
    }

    protected fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val button = findViewById(R.id.button) as Button
        button.setOnClickListener(viewClickListener)
    }

    private fun showPopupMenu(v: View) {
        val popupMenu = PopupMenu(this, v)
        popupMenu.inflate(R.menu.popupmenu) // Для Android 4.0
        // для версии Android 3.0 нужно использовать длинный вариант
        // popupMenu.getMenuInflater().inflate(R.menu.popupmenu,
        // popupMenu.getMenu());
        popupMenu
            .setOnMenuItemClickListener(object                    :
PopupMenu.OnMenuItemClickListener() {

                fun onMenuItemClick(item: MenuItem): Boolean {
                    // Toast.makeText(PopupMenuDemoActivity.this,
                    // item.toString(), Toast.LENGTH_LONG).show();
                    // return true;
                    when (item.itemId) {
                        R.id.menu1 -> {
                            Toast.makeText(
```

```
                    getApplicationContext(),
                    "Вы выбрали PopupMenu 1",
                    Toast.LENGTH_SHORT
                ).show()
                return true
            }
            R.id.menu2 -> {
                Toast.makeText(
                    getApplicationContext(),
                    "Вы выбрали PopupMenu 2",
                    Toast.LENGTH_SHORT
                ).show()
                return true
            }
            R.id.menu3 -> {
                Toast.makeText(
                    getApplicationContext(),
                    "Вы выбрали PopupMenu 3",
                    Toast.LENGTH_SHORT
                ).show()
                return true
            }
            else -> return false
        }
    }
}

    popupMenu.setOnDismissListener(object :
PopupMenu.OnDismissListener() {

        fun onDismiss(menu: PopupMenu) {
            Toast.makeText(
                getApplicationContext(), "onDismiss",
                Toast.LENGTH_SHORT
            ).show()
```

```
            }
        })
        popupMenu.show()
    }
}
```

Необходимо создать новый экземпляр PopupMenu, указав контекст активности и компонент, к которому будет привязано это меню. Далее необходимо загрузить меню из ресурсов и добавить методы для обработки щелчков. Для отображения на экране вызывается метод show(). Запустив проект, и щелкнув по кнопке, вы можете увидеть всплывающее меню (рисунок 10), а после выбора пункта меню появится соответствующее всплывающее сообщение (рисунок 11).

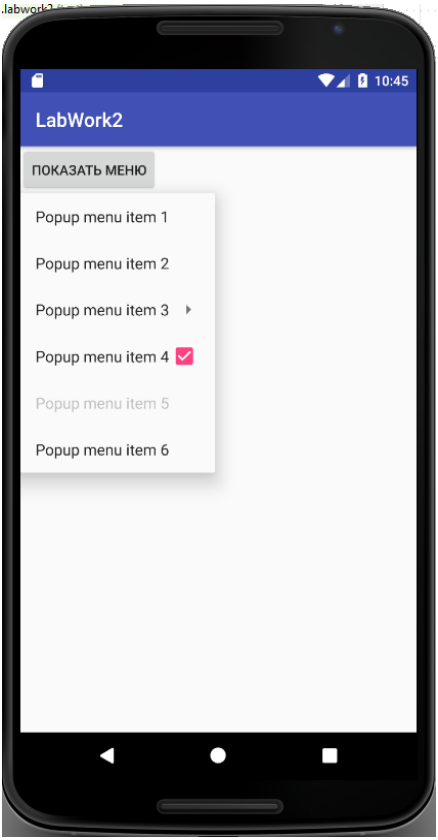


**Рис. 10.** Всплывающее меню


**Рис. 11.** Сообщение выбора

## ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Для всех вариантов изучить стандартные типы компоновок. Разобраться с физическими характеристиками экранов и единицами измерения для задания параметров View элементов. Рассмотреть имеющиеся свойства базовых виджетов. Создать приложение согласно заданию указанному в варианте.

## ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

В качестве результата работы приложения необходимо создать не менее шести логов. Также необходимо добавить не менее четырех уведомлений. В меню, при имеющейся возможности, необходимо добавить иконки.

## ВАРИАНТЫ ЗАДАНИЙ

**Вариант 1**

Используя меню Options (подменю) разработать приложение для выбора цвета (4 цвета) и фигуры (3 фигуры). В меню предусмотреть пункт очистки Activity т.е. заливка layout белым цветом. Изначально экран пустой с белым цветом фона. Если пользователь выбирает цвет, то он запоминается, но фигура не отображается. Если пользователь выбирает фигуру, то она отображается цветом по умолчанию. В дальнейшем цвет и фигура меняются в зависимости от выбора пунктов меню.

**Вариант 2**

Используя меню Options (подменю) разработать приложение для выбора цвета (4 цвета), фигуры (3 фигуры) и выбора области Activity для вывода фигуры (по центру, слева, справа, сверху и внизу). В меню предусмотреть пункт очистки Activity т.е. заливка layout белым цветом. Изначально экран пустой с белым цветом фона. Если пользователь выбирает цвет и область вывода, то они запоминаются, но фигура не отображается. Если пользователь выбирает фигуру, то она отображается цветом по умолчанию и в области по умолчанию,

например, в центре. В дальнейшем цвет, область вывода и фигура меняются в зависимости от выбора пунктов меню.

**Вариант 3**

Используя меню Options (подменю) разработать приложение для выбора цвета (4 цвета), фигуры (3 фигуры) и выбора области Activity для вывода фигуры (по центру, слева, справа, сверху и внизу). В меню предусмотреть пункт очистки Activity т.е. заливка layout белым цветом. При выборе цвета, фигуры и области вывода в меню включить пункт для задания случайных значений данных параметров, при этом устанавливая один из параметров случайным образом, все другие параметры также должны быть установлены случайно. Изначально экран пустой с белым цветом фона. Если пользователь выбирает цвет и область вывода, то они запоминаются, но фигура не отображается. Если пользователь выбирает фигуру, то она отображается цветом по умолчанию и в области по умолчанию, например, в центре. В дальнейшем цвет, область вывода и фигура меняются в зависимости от выбора пунктов меню.

**Вариант 4**

Разработать приложение, в котором пользователь указывает в компонентах Edit Text цвета в модели RGB. При нажатии на кнопку Button в трех элементах ImageView отображаются заданные цвета и размер элемента соответствует введенной компоненте. В четвертом элементе ImageView отображается заданный цвет.

**Вариант 5**

На компоненте Layout находится три компонента TextView. Для первого компонента TextView создать контекстное меню для выбора 5 фиксированных цветов текста. Для второго компонента TextView создать контекстное меню для выбора 5 фиксированных размеров текста. Для третьего компонента TextView создать контекстное меню для выбора 3 положений текста в TextView. Предусмотреть OptionsMenu для сброса всех назначенных параметров по умолчанию

и задания 4-х фиксированных цветов Layout в том числе случайным образом.

**Вариант 6**

Имеются 10 компонентов imageButton, расположенные в отдельном Layout, для которых определены случайные цвета при запуске приложения. При выборе imageButton менять цвет другого Layout, на котором по центру расположен элемент TextView с произвольным цветом. Для элемента TextView указать контекстное меню для выбора размера текста (6 значений). Предусмотреть основное меню, в котором происходит сброс всех назначенных ранее параметров.

**Вариант 7**

Имеются компоненты TextView и ImageView. Соответственно в них определены текст и изображение. С каждым компонентом связать контекстное меню, в котором определены по 4 разные вида анимации (2 отдельных и 2 смешанных). Предусмотреть OptionsMenu, в котором можно выбрать случайную анимацию для каждого из компонентов.

**Вариант 8**

Имеется компонент TextView с произвольным текстом по центру экрана. Используя OptionsMenu задать размер текста, цвет текста и цвет фона текста. Для данного TextView связать контекстное меню с выбором трех вариантов анимации.

**Вариант 9**

Имеется компонент imageView с заданным по умолчанию изображением. Для данного imageView связать контекстное меню с выбором трех дополнительных изображений. Используя OptionsMenu задать 3 варианта анимации для данного компонента, при этом один вариант анимации должен быть смешанным.

**Вариант 10**

По центру Activity расположен элемент DigitalClock. Связать с элементом контекстное меню, в котором задаются 4 разные вида

анимации в том числе и смешанная. Разработать OptionsMenu, для которого предусмотреть подменю выбора цвета, размера и положения элемента DigitalClock на Layout.

**Вариант 11**

На layout расположены 3 компонента EditText. С использованием меню отображать и скрывать 3 компонента EditText. Также имеется возможность из OptionsMenu задавать случайные значения от 0 до 255 для данных EditText. Один из пунктов меню предназначен для присвоения полученного в EditText цвета в модели RGB для Layout.

**Вариант 12**

Activity имеет 3 Layout и 3 EditText для задания цвета. С использованием OptionsMenu меню для 1 Layout задать фиксированный цвет. С использованием контекстного меню для второго Layout задать фиксированный цвет. Для третьего Layout считать значения компонент RGB из EditText и с использованием OptionsMenu применить данный цвет к третьему layout.

**Вариант 13**

На Layout расположены 2 компонента TextView. Для первого компонента TextView задать цвет текста, размер текста, цвет фона и сброс параметров с помощью контекстного меню. Для второго TextView, используя контекстное меню, изменить положение компонента TextView на Layout. Выбрать шесть различных областей.

**Вариант 14**

На Layout расположены три компонента TextView и три компонента RadioButton. В зависимости от выбора RadioButton менять посредством подменю размер текста, цвет текста и цвет фона ассоциированного с RadioButton компонента TextView.

**Вариант 15**

На Layout расположены три компонента TextView и три компонента CheckButton. В зависимости от выбора элементов CheckButton с

использованием OptionsMenu и подменю менять цвет текста, начертание и размер. При отсутствии выбранного CheckButton при выборе любого пункта меню для всех TextView применять параметры по умолчанию.

**Вариант 16**

На Layout расположены два компонента TextView и два компонента ToggleButton. Для первого компонента TextView назначить контекстное меню для выбора вариантов выравнивания текста (3 типа выравнивания). Для второго TextView с использованием контекстного меню задать начертание текста. С использованием OptionsMenu в зависимости от выбора ToggleButton изменить цвет и размер текста, а также предусмотреть сброс всех параметров на значения по умолчанию.

**Вариант 17**

На layout расположены два компонента TextView и два компонента RadioButton. В зависимости от выбора RadioButton с использованием OptionsMenu применить три вида анимации к выбранному компоненту TextView. Для второго компонента TextView создать контекстное меню для выбора размера и цвета текста.

**Вариант 18**

На layout расположены компоненты ImageView, EditText, TextView и 2 компонента Button. По нажатии на кнопки (+,-) параметр alpha компонента ImageView меняется на значение, установленное в EditText, а в компоненте TextView отображается текущее значение параметра alpha. С использованием OptionsMenu задать параметр alpha для ImageView следующими значениями: 10, 40, 80, 100 и случайно.

## КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Раскройте понятие «компоновка». Опишите иерархию элементов компоновки.
2. Перечислите существующие типы компоновок и дайте их краткое описание.
3. Перечислите существующие способы создания компоновки.
4. Изложите принцип создания компоновки в xml файле. В чем преимущество такого подхода.
5. Опишите графические возможности создания интерфейса в Android Studio.
6. Раскройте понятие «виджет». Перечислите базовые типы виджетов и дайте их краткое описание.
7. Перечислите способы задания обработчиков событий.
8. Перечислите существующие типы меню.
9. Дайте характеристику меню выбора опций. В чем заключается способ создания меню при помощи ресурсов.
10. Опишите процесс обработки событий нажатия пунктов меню.
11. Дайте характеристику контекстного меню.
12. Дайте характеристику всплывающего меню.

## ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), листинг (xml код разметки экрана, графическое представление, соответствующее разметке экрана, код программы и при необходимости наличие кода дополнительных классов), результаты выполнения работы (графические изображения примеров работы приложения), выводы.

## ОСНОВНАЯ ЛИТЕРАТУРА


1. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android / А. Семакова. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 103 с. : ил. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429181
2. Введение в разработку приложений для ОС Android / Ю.В. Березовская, О.А. Юфрякова, В.Г. Вологдина и др. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 434 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428937
3. Разработка приложений для смартфонов на ОС Android / Е.А. Латухина, О.А. Юфрякова, Ю.В. Березовская, К.А. Носов. - 2-е изд., исправ. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 252 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428807


## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА


4. Дэвид, Х. Разработка приложений Java EE 6 в NetBeans 7. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2013. — 330 с. — Режим доступа: http://e.lanbook.com/book/58693 — Загл. с экрана.
5. Сильвен, Р. Android NDK. Разработка приложений под Android на C/C++. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2012. — 496 с. — Режим доступа: http://e.lanbook.com/book/9126 — Загл. с экрана.
6. Ретабоуил, С. Android NDK: руководство для начинающих. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2016. — 518 с. — Режим доступа: http://e.lanbook.com/book/82810 — Загл. с экрана.

7. Соколова, В.В. Разработка мобильных приложений: учебное пособие / В.В. Соколова ; Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский государственный университет», Министерство образования и науки Российской Федерации. - Томск: Издательство Томского политехнического университета, 2015. - 176 с.: ил., табл., схем. - Библиогр. в кн.. - ISBN 978-5-4387-0369-3; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=442808 (15.06.2017).
8. Баженова, И.Ю. Язык программирования Java / И.Ю. Баженова. - М.: Диалог-МИФИ, 2008. - 254 с. : табл., ил. - ISBN 5-86404-091-6 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=54745 (15.06.2017).
9. Джошуа Блох Java. Эффективное программирование [Электронный ресурс]/ Джошуа Блох— Электрон. текстовые данные. — Саратов: Профобразование, 2017.— 310 с.— Режим доступа: http://www.iprbookshop.ru/64057.html .— ЭБС «IPRbooks»

**Электронные ресурсы:**

10. Научная электронная библиотека http://eLIBRARY.RU
11. Электронно-библиотечная система http://e.lanbook.com
12. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru
13. Электронно-библиотечная система IPRBook http://www.iprbookshop.ru
14. Базовый сайт о системе Android - https://www.android.com/intl/ru_ru
15. Разработка приложения на базе Android - https://developer.android.com/index.html