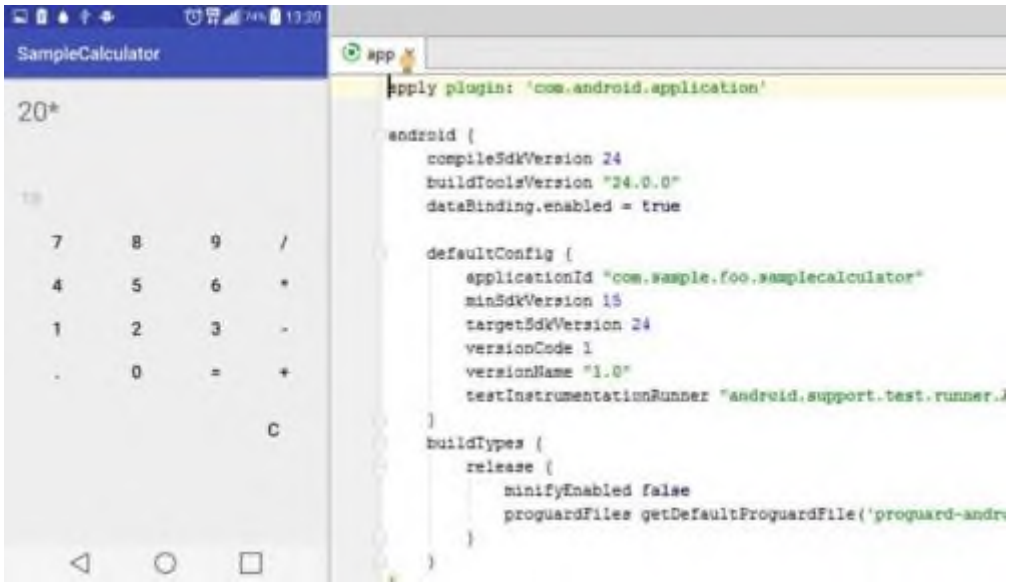
9 лабораторная работа

Тема: Создайте приложение-калькулятор и установите его на свой мобильный телефон.

Цель работы: Создание приложение- калькулятор и его компиляция.

В этом руководстве мы расскажем, как создать калькулятор на Java для Android. Если вы новичок в программировании и никогда раньше не создавали приложения, ознакомьтесь с нашим предыдущим руководством по написанию первого приложения для Android:

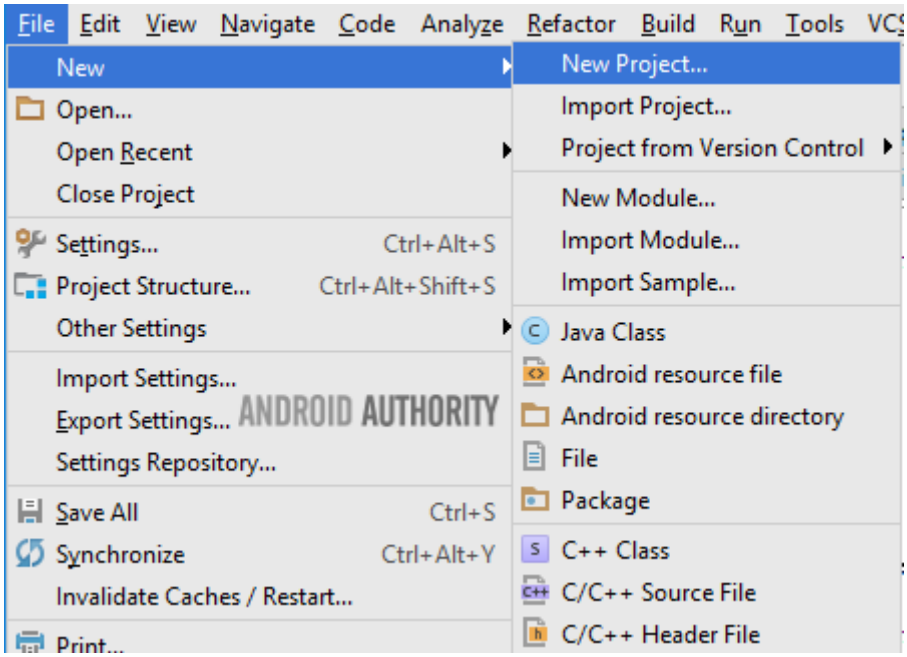


Предполагается, что у вас есть хотя бы минимальный базовый опыт создания **Android – приложений**.

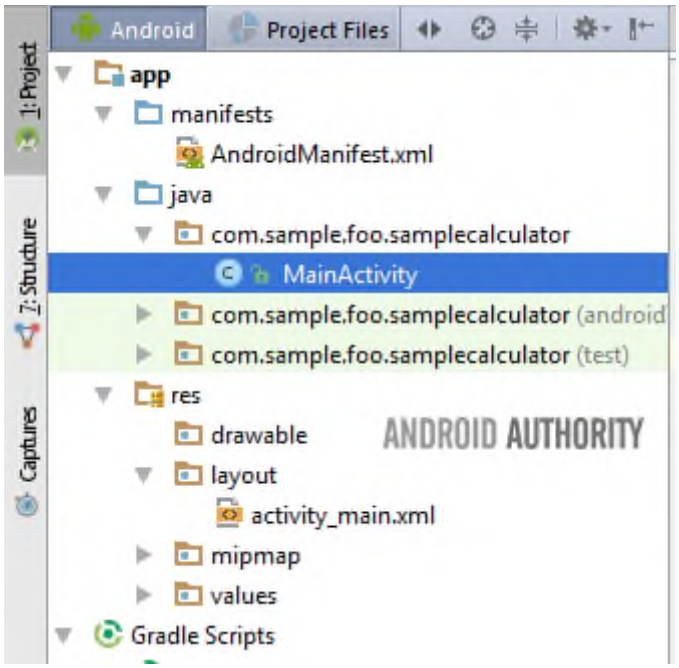
Полный исходный код калькулятора, описанного ниже, доступен для использования и изменения на github.

Создание проекта

Первое, что нужно сделать - это создать в **Android Studio** новый проект: **Start a new Android Studio project или File - New - New Project**:



Для этого руководства мы выбрали в панели «***Add an Activity to Mobile***» опцию «***EmptyActivity***», для «***MainActivity***» мы оставили имя по умолчанию – «***Activity***». На этом этапе структура должна выглядеть, как показано на рисунке ниже. У вас есть **MainActivity** внутри пакета проекта и файл **activity_main.xml** в папке **layout**:



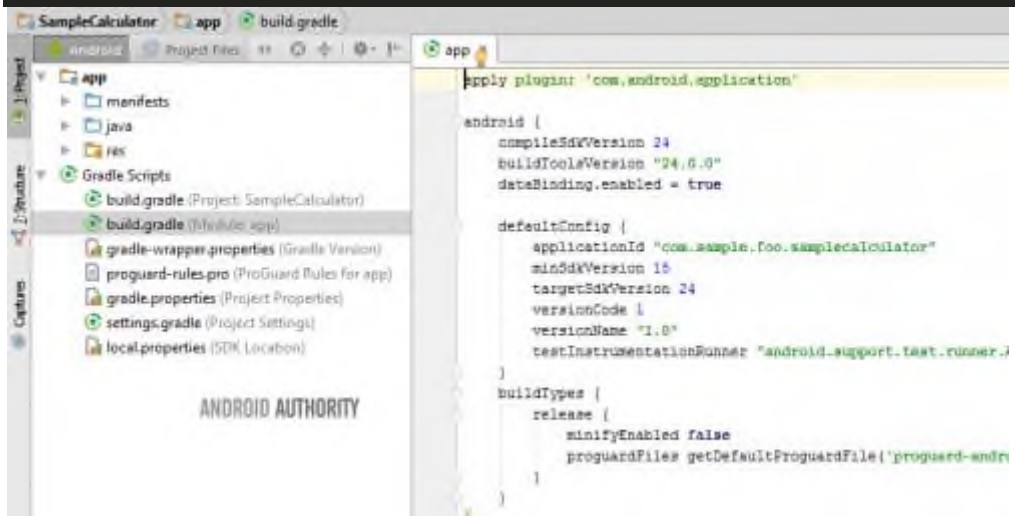
Включение привязки данных в проекте

Перед тем, как создать приложение для Андроид с нуля, нужно уяснить, что использование привязки данных помогает напрямую обращаться к виджетам (***Buttons***, ***EditText*** *и* ***TextView***), а не находить их с помощью

методов **findViewById()**. Чтобы включить привязку данных, добавить следующую строку кода в файл **build.gradle**.

JAVA

```
android {
    ...
    dataBinding.enabled = true
    ...
```



Разработка макета калькулятора

Для включения привязки данных в файле **activity_main.xml** требуется еще одно изменение. Оберните сгенерированный корневой тег (***RelativeLayout***) в **layout**, таким образом сделав его новым корневым тегом.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout>
    <RelativeLayout>
    ...
    </RelativeLayout>
</layout>
```

Как научиться создавать приложения для Андроид? Читайте наше руководство дальше.

Тег layout - это предупреждает систему построения приложения, что этот файл макета будет использовать привязку данных. Затем система генерирует для этого файла макета класс **Binding**. Поскольку целевой **XML-файл** называется **activity_main.xml**, система построения приложения создаст класс **ActivityMainBinding**, который можно использовать в приложении, как

и любой другой класс **Java**. Имя класса составляется из имени файла макета, в котором каждое слово через подчеркивание будет начинаться с заглавной буквы, а сами подчеркивания убираются, и к имени добавляется слово «***Binding***».

Теперь перейдите к файлу **MainActivity.java**. Создайте закрытый экземпляр **ActivityMainBinding** внутри вашего класса, а в методе **onCreate()** удалите строку **setContentView ()** и вместо нее добавьте **DataBindingUtil.setContentView()**, как показано ниже.

JAVA

```
public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
    }
}
```

Общие принципы создания виджетов макета

В приложении калькулятора есть четыре основных элемента:

RelativeLayout - определяет, как другие элементы будут укладываться или отображаться на экране. **RelativeLayout** используется для позиционирования дочерних элементов по отношению друг к другу или к самим себе.

TextView - элемент используется для отображения текста. Пользователи не должны взаимодействовать с этим элементом. С помощью **TextView** отображается результат вычислений.

EditText - похож на элемент TextView, с той лишь разницей, что пользователи могут взаимодействовать с ним и редактировать текст. Но поскольку калькулятор допускает только фиксированный набор вводимых данных, мы устанавливаем для него статус «*не редактируемый*». Когда пользователь нажимает на цифры, мы выводим их в **EditText**.

Button - реагирует на клики пользователя. При создании простого приложения для Андроид мы используем кнопки для цифр и операторов действий в калькуляторе.

Создание макета калькулятора

Многопоточность в Java – руководство с примерами

Код макета калькулятора объемный. Это связано с тем, что мы должны явно определять и тщательно позиционировать каждую из кнопок интерфейса. Ниже представлен фрагмент сокращенной версии файла макета **activity_main**:

```
<layout>
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.sample.foo.samplecalculator.MainActivity">

    <TextView
        android:id="@+id/infoTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="30dp"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/infoTextView"
        android:enabled="false"
        android:gravity="bottom"
        android:lines="2"
        android:maxLines="2"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonSeven"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/editText"
        android:text="7"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonEight"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/editText"
        android:layout_toRightOf="@id/buttonSeven"
        android:text="8"
        android:textSize="20sp" />

    <Button
```

```
    android:id="@+id/buttonNine"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText"
    android:layout_toRightOf="@id/buttonEight"
    android:text="9"
    android:textSize="20sp" />

...

...

<Button
    android:id="@+id/buttonDot"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/buttonOne"
    android:text="."
    android:textSize="20sp" />

<Button
    android:id="@+id/buttonZero"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/buttonEight"
    android:layout_below="@id/buttonTwo"
    android:text="0"
    android:textSize="20sp" />

<Button
    android:id="@+id/buttonEqual"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/buttonNine"
    android:layout_below="@id/buttonThree"
    android:text="="
    android:textSize="20sp" />

<Button
    android:id="@+id/buttonDivide"
    style="@style/Widget.AppCompat.Button.Borderless"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/buttonNine"
        android:layout_toRightOf="@id/buttonNine"
        android:text="/"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonMultiply"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/buttonSix"
        android:layout_toRightOf="@id/buttonSix"
        android:text="*"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonSubtract"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/buttonThree"
        android:layout_toRightOf="@id/buttonThree"
        android:text="-"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonAdd"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/buttonEqual"
        android:layout_toRightOf="@id/buttonEqual"
        android:text="+"
        android:textSize="20sp" />

    <Button
        android:id="@+id/buttonClear"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@id/buttonAdd"
        android:layout_below="@id/buttonAdd"
        android:layout_marginTop="@dimen/activity_vertical_margin"
```

```
        android:text="C"
        android:textSize="20sp" />
    </RelativeLayout>
</layout>
```

Язык программирования Java - руководство для начинающих

Внутренние компоненты калькулятора

Перед тем, как создать приложение на телефон **Android**, отметим, что **valueOne** и **valueTwo** содержат цифры, которые будут использоваться. Обе переменные имеют тип **double**, поэтому могут содержать числа с десятичными знаками и без них. Мы устанавливаем для **valueOne** специальное значение **NaN** (*не число*) - подробнее это будет пояснено ниже.

```
private double valueOne = Double.NaN;
    private double valueTwo;
```

Этот простой калькулятор сможет выполнять только операции сложения, вычитания, умножения и деления. Поэтому мы определяем четыре статических символа для представления этих операций и переменную **CURRENT_ACTION**, содержащую следующую операцию, которую мы намереваемся выполнить.

```
private static final char ADDITION = '+';
    private static final char SUBTRACTION = '-';
    private static final char MULTIPLICATION = '*';
    private static final char DIVISION = '/';
    private char CURRENT_ACTION;
```

Затем мы используем класс **DecimalFormat** для форматирования результата. Конструктор десятичного формата позволяет отображать до десяти знаков после запятой.

```
decimalFormat = new DecimalFormat("#.##########");
```

Обработка нажатий на цифры

В нашем создаваемом простом приложении для **Андроид** всякий раз, когда пользователь нажимает на цифру или точку, нам нужно добавить эту цифру в **editText**. Пример кода ниже иллюстрирует, как это делается для цифры ноль (**0**).

```
binding.buttonZero.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            binding.editText.setText(binding.editText.getText() + "0");
        }
    });
```

Обработка кликов по кнопкам операторов

Обработка нажатия кнопок операторов (*действий*) выполняется по-другому. Сначала нужно выполнить все ожидающие в очереди вычисления. Поэтому мы определяем метод **computeCalculation**. В **computeCalculation**, если **valueOne** является допустимым числом, мы считываем **valueTwo** из **editText** и выполняем текущие операции в очереди.

Если же **valueOne** является **NaN**, для **valueOne** присваивается цифра в **editText**.

JAVA

```
private void computeCalculation() {
        if(!Double.isNaN(valueOne)) {
            valueTwo = Double.parseDouble(binding.editText.getText().toString());
            binding.editText.setText(null);
            if(CURRENT_ACTION == ADDITION)
                valueOne = this.valueOne + valueTwo;
            else if(CURRENT_ACTION == SUBTRACTION)
                valueOne = this.valueOne - valueTwo;
            else if(CURRENT_ACTION == MULTIPLICATION)
                valueOne = this.valueOne * valueTwo;
            else if(CURRENT_ACTION == DIVISION)
                valueOne = this.valueOne / valueTwo;
        }
        else {
            try {
                valueOne = Double.parseDouble(binding.editText.getText().toString());
            }
            catch (Exception e){}
        }
    }
```

Продолжаем создавать копию приложения на **Андроид**. Для каждого оператора мы сначала вызываем **computeCalculation()**, а затем устанавливаем для выбранного оператора **CURRENT_ACTION**. Для оператора равно **(=)** мы вызываем **computeCalculation()**, а затем очищаем содержимое **valueOne** и **CURRENT_ACTION**.

JAVA

```
binding.buttonAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                computeCalculation();
                CURRENT_ACTION = ADDITION;
                binding.infoTextView.setText(decimalFormat.format(valueOne) + "+");
                binding.editText.setText(null);
            }
        });
        binding.buttonSubtract.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = SUBTRACTION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "-");
            binding.editText.setText(null);
        }
    });
    binding.buttonMultiply.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = MULTIPLICATION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "*");
            binding.editText.setText(null);
        }
    });
    binding.buttonDivide.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = DIVISION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "/");
            binding.editText.setText(null);
        }
    });
    binding.buttonEqual.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();

binding.infoTextView.setText(binding.infoTextView.getText().toString() +
                    decimalFormat.format(valueTwo) + " = " +
decimalFormat.format(valueOne));
            valueOne = Double.NaN;
            CURRENT_ACTION = '0';
        }
    });
```

Мы завершили создание простого калькулятора. Теперь вы сможете **создать приложение для Андроид** сами.