Методические материалы

Практическая работа 5+

Основные элементы управления

TextView

Для простого вывода текста на экран предназначен элемент TextView. Он просто отображает текст без возможности его редактирования. Некоторые его основные атрибуты:

- android:text: устанавливает текст элемента
- android:textSize: устанавливает высоту текста, в качестве единиц измерения для указания высоты используются sp
- android:background: задает фоновый цвет элемента в виде цвета в шестнадцатиричной записи или в виде цветового ресурса
- android:textColor: задает цвет текста
- android:textAllCaps: при значении true делает все символы в тексте заглавными
- android:textDirection: устанавливает направление текста. По умолчанию используется направление слева направо, но с помощью значения rtl можно установить направление справо налево
- android:textAlignment: задает выравнивание текста. Может принимать следующие значения:
- center: выравнивание по центру
- textStart: по левому краю
- textEnd: по правому краю
- viewStart: при направлении текста слева направо выравнивание по левому краю, при направлении справа налево - по правому
- viewEnd: при направлении текста слева направо выравнивание по правому краю, при направлении справа налево - по левому
- android:fontFamily: устанавливает тип шрифта. Может принимать следующие значения:
  - ✓ monospace
  - ✓ serif
  - ✓ serif-monospace
  - ✓ sans-serif

- ✓ sans-serif-condensed
- ✓ sans-serif-smallcaps
- ✓ sans-serif-light
- ✓ casual
- ✓ cursive

Например, определим три текстовых поля:

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

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_margin="10dp"

        android:text="Hello Android "
        android:fontFamily="sans-serif"
        android:textSize="26sp"
        android:background="#ffebee"
        android:textColor="#f44336"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="0dp"
```

```
        android:layout_margin="10dp"

        android:text="Hello Java"
        android:textAllCaps="true"
        android:textSize="26sp"
        android:background="#ede7f6"
        android:textColor="#7e57c2"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_margin="10dp"

        android:text="Hello World"
        android:textAlignment="textEnd"
        android:textSize="26sp"
        android:background="#e8eaf6"
        android:textColor="#5c6bc0"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Атрибуты TextView в Android Studio

Установка элемента в коде тоже не отличается сложностью.

Например, создадим элемент и выведем его на экран:

```
package com.example.viewapp;
 import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.graphics.Typeface;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
        ConstraintLayout constraintLayout = new ConstraintLayout(this);
       TextView textView = new TextView(this);
       // установка фонового цвета
```

```
        textView.setBackgroundColor(0xffe8eaf6);
        // установка цвета текста
        textView.setTextColor(0xff5c6bc0);
        // делаем все буквы заглавными
        textView.setAllCaps(true);
        // устанавливаем вравнивание текста по центру
        textView.setTextAlignment(TextView.TEXT_ALIGNMENT_CENTER);
        // устанавливаем текста
        textView.setText("Hello Android!");
        // установка шрифта
        textView.setTypeface(Typeface.create("casual", Typeface.NORMAL));
        // устанавливаем высоту текста
        textView.setTextSize(26);

        ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
        textView.setLayoutParams(layoutParams);

        constraintLayout.addView(textView);
        setContentView(constraintLayout);
    }
}
```

Создание TextView в Java в Android Studio

Иногда необходимо вывести на экран какую-нибудь ссылку, либо телефон, по нажатию на которые производилось бы определенное действие. Для этого в TextView определен атрибут **android:autoLink**:

```
<TextView
    android:text="Посетите сайт https://yandex.ru"
    android:textSize="21sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="web | email"

    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
```

autolink in Android

android:autoLink может принимать несколько значений:

- none: отключает все ссылки
- web: включает все веб-ссылки
- email: включает ссылки на электронные адреса
- phone: включает ссылки на номера телефонов
- map: включает ссылки на карту
- all: включает все вышеперечисленные ссылки

То есть при настройке android:autoLink="web" если в тексте есть упоминание адреса url, то этот адрес будет выделяться, а при нажатии на него будет осуществлен переход к веб-браузеру, который откроет страницу по этому адресу. С помощью прямой черты мы можем объединять условия, как в данном случае: android:autoLink="web | email"

EditText

Элемент EditText является подклассом класса TextView. Он также представляет текстовое поле, но теперь уже с возможностью ввода и редактирования текста. Таким образом, в EditText мы можем использовать все те же возможности, что и в TextView.

Из тех атрибутов, что не рассматривались в теме про TextView, следует отметить атрибут **android:hint**. Он позволяет задать текст, который будет отображаться в качестве подсказки, если элемент EditText пуст. Кроме того, мы можем использовать атрибут **android:inputType**, который позволяет задать клавиатуру для ввода. В частности, среди его значений можно выделить следующие:

- text: обычная клавиатура для ввода однострочного текста
- textMultiLine: многострочное текстовое поле
- textEmailAddress: обычная клавиатура, на которой присутствует символ @, ориентирована на ввод email
- textUri: обычная клавиатура, на которой присутствует символ /, ориентирована на ввод интернет-адресов
- textPassword: клавиатура для ввода пароля
- textCapWords: при вводе первый введенный символ слова представляет заглавную букву, остальные - строчные

- number: числовая клавиатура
- phone: клавиатура в стиле обычного телефона
- date: клавиатура для ввода даты
- time: клавиатура для ввода времени
- datetime: клавиатура для ввода даты и времени

Используем EditText:

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

    <EditText
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Введите имя"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <EditText
```

```
    android:id="@+id/message"
    android:layout_marginTop="16dp"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:hint="Введите сообщение"
    android:inputType="textMultiLine"
    android:gravity="top"
    app:layout_constraintTop_toBottomOf="@+id/name"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Первое поле здесь обычное однострочное, а второе - многострочное. Чтобы во втором поле текст выравнивался по верху, дополнительно устанавливается атрибут android:gravity="top".

EditText в Java и Android

Одной из возможностей элемента EditText также является возможность обработать введенные символы по мере ввода пользователя. Для этого определим в файле **activity_main.xml** следующую разметку:

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
```

```
        android:layout_height="wrap_content"
        android:textSize="34sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Введите имя"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Предполагается, что введенные в EditText символы тут же будут отображаться в элементе TextView. И для этого также изменим код MainActivity:

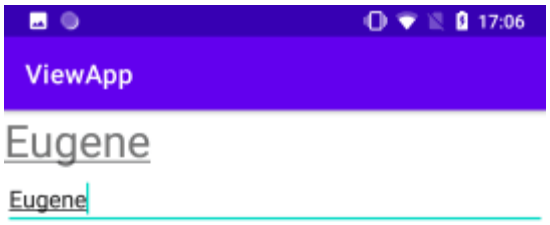
```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.widget.EditText;

import android.widget.TextView;


public class MainActivity extends AppCompatActivity {
```

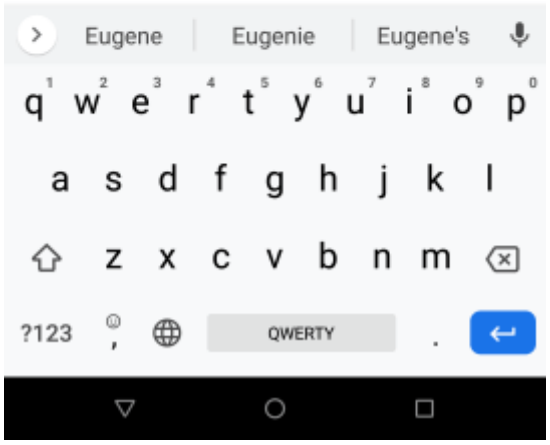
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText editText = findViewById(R.id.editText);

        editText.addTextChangedListener(new TextWatcher() {

            public void afterTextChanged(Editable s) {}

            public void beforeTextChanged(CharSequence s, int start,
                                          int count, int after) {
            }

            public void onTextChanged(CharSequence s, int start, int before, int count)
{
                TextView textView = findViewById(R.id.textView);
                textView.setText(s);
            }
        });
    }
}
```

С помощью метода addTextChangedListener() здесь к элементу EditText добавляется слушатель ввода текста - объект TextWatcher. Для его использования нам надо реализовать три метода, но в реальности нам хватит реализации метода onTextChanged, который вызывается при изменении текста. Введенный текст передается в этот метод в качестве параметра

CharSequence. В самом методе просто передаем этот текст в элемент TextView.

В итоге при вводе в EditText все символы также будут отображаться в TextView:





TextChangedListener и EditText в Android

Button

Одним из часто используемых элементов являются кнопки, которые представлены классом android.widget.Button. Ключевой особенностью кнопок является возможность взаимодействия с пользователем через нажатия.

Некоторые ключевые атрибуты, которые можно задать у кнопок:

text: задает текст на кнопке

textColor: задает цвет текста на кнопке

background: задает фоновый цвет кнопки

textAllCaps: при значении true устанавливает текст в верхнем регистре. По умолчанию как раз и применяется значение true

onClick: задает обработчик нажатия кнопки

Итак, изменим код в activity_main.xml следующим образом:

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="34sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
    <EditText
```

```
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Введите имя"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ввод"
        android:onClick="sendMessage"
        app:layout_constraintTop_toBottomOf="@+id/editText"
        app:layout_constraintLeft_toLeftOf="parent" />



</androidx.constraintlayout.widget.ConstraintLayout>
```

При помощью атрибута android:onClick можно задать метод в коде java, который будет обрабатывать нажатия кнопки. Так, в вышеприведенном примере это метод sendMessage. Теперь перейдем к коду MainActivity и пропишем в нем такой метод:

```
package com.example.viewapp;


import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    // Обработка нажатия кнопки
    public void sendMessage(View view) {
        TextView textView = findViewById(R.id.textView);
        EditText editText = findViewById(R.id.editText);
        textView.setText("Добро пожаловать, " + editText.getText());
    }
}
```

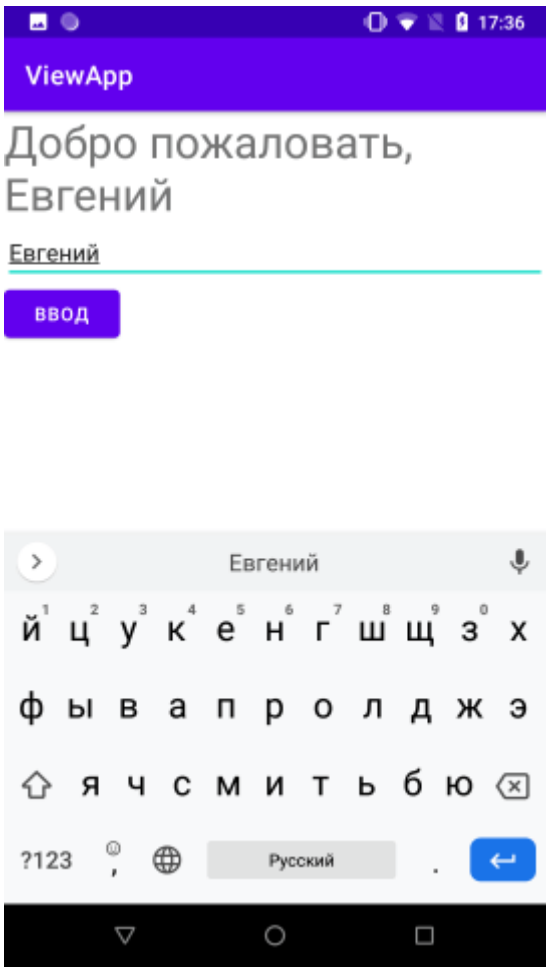
При создании метода обработки нажатия следует учитывать следующие моменты:

Метод должен объявляться с модификатором public

Должен возвращать значение void

В качестве параметра принимать объект View. Этот объект View и представляет собой нажатую кнопку

В данном случае после нажатия на кнопку в TextView выводится текст из EditText.



onClick и Button нажатие кнопки в Java и Android

Аналогичный пример полностью в коде MainActivity:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
```

```
    EditText editText;
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);

        ConstraintLayout constraintLayout = new ConstraintLayout(this);
        textView = new TextView(this);
        textView.setId(View.generateViewId());
        ConstraintLayout.LayoutParams textViewLayout =  new
ConstraintLayout.LayoutParams(
                ConstraintLayout.LayoutParams.MATCH_CONSTRAINT,
ConstraintLayout.LayoutParams.WRAP_CONTENT
        );
        textViewLayout.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
        textViewLayout.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        textViewLayout.rightToRight =
ConstraintLayout.LayoutParams.PARENT_ID;
        textView.setLayoutParams(textViewLayout);
        constraintLayout.addView(textView);

        editText = new EditText(this);
        editText.setId(View.generateViewId());
        editText.setHint("Введите имя");
        ConstraintLayout.LayoutParams editTextLayout =  new
ConstraintLayout.LayoutParams(
                ConstraintLayout.LayoutParams.MATCH_CONSTRAINT,
ConstraintLayout.LayoutParams.WRAP_CONTENT
        );
```

```
        editTextLayout.topToBottom = textView.getId();
        editTextLayout.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        editTextLayout.rightToRight =
ConstraintLayout.LayoutParams.PARENT_ID;
        editText.setLayoutParams(editTextLayout);
        constraintLayout.addView(editText);


        Button button = new Button(this);
        button.setText("Ввод");
        ConstraintLayout.LayoutParams buttonLayout =  new
ConstraintLayout.LayoutParams(
                ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT
        );
        buttonLayout.topToBottom = editText.getId();
        buttonLayout.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        button.setLayoutParams(buttonLayout);
        constraintLayout.addView(button);


        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Обработка нажатия
                textView.setText("Добро пожаловать, " + editText.getText());
            }
        });


        setContentView(constraintLayout);
    }
}
```

При программном создании кнопки мы можем определить у нее слушатель нажатия View.OnClickListener и с помощью его метода onClick также обработать нажатие:

```
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Обработка нажатия
    }
});
```

Всплывающие окна. Toast

Для создания простых уведомлений в Android используется класс Toast. Фактически Toast представляет всплывающее окно с некоторым текстом, которое отображается в течение некоторого времени.

Объект Toast нельзя создать в коде разметки xml, например, в файле activity_main.xml. Toast можно использовать только в коде java.

Так, определим в файле разметки activity_main.xml кнопку:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
```

```
        android:onClick="onClick"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

У кнопки установлен обработчик нажатия - метод onClick. Определим его в коде MainActivity:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        setContentView(R.layout.activity_main);
    }

    public  void onClick(View view){
        Toast toast = Toast.makeText(this, "Hello
Android!",Toast.LENGTH_LONG);
        toast.show();
    }
}
```

В обработчике отображается всплывающее окно. Для его создания применяется метод **Toast.makeText()**, в который передается три параметра: текущий контекст (текущий объект activity), отображаемый текст и время отобажения окна.

В качестве времени показа окна мы можем использовать целочисленное значение - количесво миллисекунд или встроенные константы **Toast.LENGTH_LONG** (3500 миллисекунд) и **Toast.LENGTH_SHORT** (2000 миллисекунд).

Для самого отображения окна вызывается метод show():

Toast в Android и Java

По умолчанию окно отображается внизу интерфейса с центрированием по центру. Но мы можем кастомизировать позиционирование окна с помощью методов **setGravity()** и **setMargin()**. Так, изменим метод onClick:

```
public  void onClick(View view){
```

```
    Toast toast = Toast.makeText(this, "Hello Android!", Toast.LENGTH_LONG);
    toast.setGravity(Gravity.TOP, 0,160);   // import android.view.Gravity;
    toast.show();
}
```

Первый параметр метода setGravity указывает, в какой части контейнера надо позиционировать Toast, второй и третий параметр устанавливают отступы от этой позиции по горизонтали и вертикали сооветственно:

Позиционирование toast в Android и Java

Метод **setMargin()** принимает два параметра: отступ от левой границы контейнера в процентах от шиирины контейнера и отступ от верхней границы в процентах от длины контейнера.

Snackbar

Элемент Snackbar в некотором роде похож на Toast: он также позволяет выводить всплывающие сообщения, но теперь сообщения растягиваются по ширине экрана.

Для применения Snackbar добавим в файл activity_main.xml определение кнопки, по нажатию на которую будет появляться Snackbar:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onClick"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определена кнопка, по нажатию на которую будет оображаться сообщение.

И также изменим класс MainActivity:

```
package com.example.viewapp;
```

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

import com.google.android.material.snackbar.Snackbar;

public class MainActivity extends AppCompatActivity {

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

    public  void onClick(View view){
        Snackbar.make(view, "Hello Android", Snackbar.LENGTH_LONG)
                .show();
    }
}
```

Snackbar создается с помощью метода **make()**, в который передаются три параметра: объект View, к которому прикрепляется всплывающее оообщение, само сообщение в виде строки и параметр, который указывает, сколько будет отображаться сообщение. Последний параметр может принимать числовое значение - количество миллисекунд, либо одну из трех констант: **Snackbar.LENGTH_INDEFINITE** (отображение в течение неопределенного периода времени), **Snackbar.LENGTH_LONG** (долгое отображение) или **Snackbar.LENGTH_SHORT** (недолгое отображение).

После создания Snackbar отображается с помощью метода show:

Snackbar в Android и Java

При этом в отличие от Toast мы не можем повлиять на позицию сообщения, оно отображается внизу экрана и занимает всю нижнюю часть.

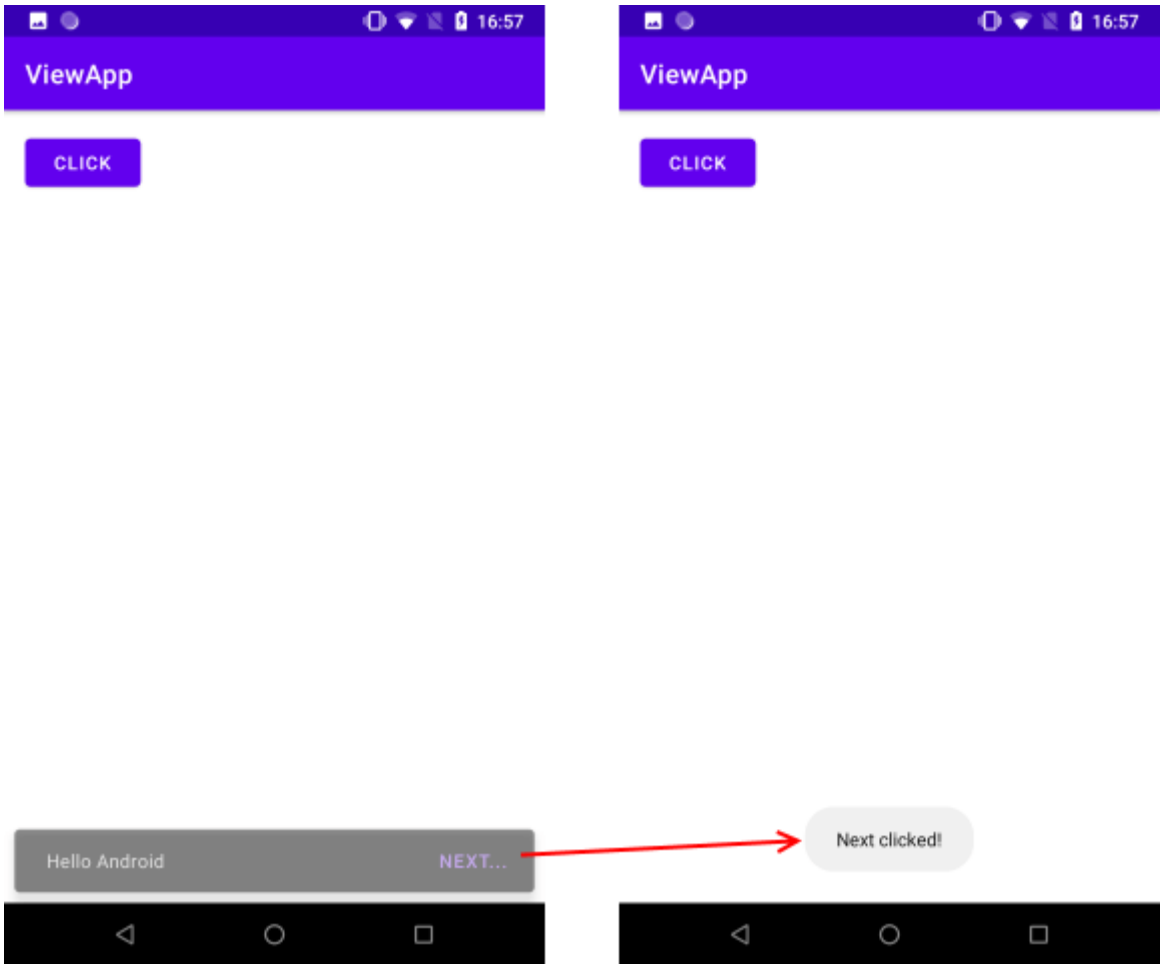
Прикрепление обработчика события

Snackbar позволяет добавить виджету действие, чтобы пользователь мог как-то прореагировать на сообщение. Например, изменим код MainActivity следующим образом:

```
package com.example.viewapp;


import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.google.android.material.snackbar.Snackbar;

public class MainActivity extends AppCompatActivity {

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

    public  void onClick(View view){
        Snackbar snackbar = Snackbar.make(view, "Hello Android",
Snackbar.LENGTH_LONG);

        snackbar.setAction("Next...", new View.OnClickListener (){
            @Override
            public void onClick(View v) {
                Toast toast = Toast.makeText(getApplicationContext(), "Next
clicked!",Toast.LENGTH_LONG);
                toast.show();
            }
        });
        snackbar.show();
    }
}
```

Для добавления действия у Snackbar применяется метод **setAction()**. Первый параметр представляет текст кнопки в сообщении, на которую может нажать пользователь - в данном случае это "Next...". Второй параметр представляет реализацию интерфейса **View.OnClickListener** (тот же самый, который используется для обработки нажатия кнопки). В методе **onClick()** сообственно выполняем действия, которые вызываются при нажатии на кнопку в сообщении. В данном случае для простоты просто отображаем всплывающее сообщение в виде объекта Toast



Обработка нажатия click в Snackbar в Android и Java

Настройка визуального вида

Ряд методов Snackbar позволяет настроить внешний вид:

- setTextColor(): настраивает цвет текста
- setBackgroundTint(): настраивает цвет фона
- setActionTextColor(): настраивает цвет текста кнопки в всплывающем сообщении

snackbar.setTextColor(0XFF81C784);

snackbar.setBackgroundTint(0XFF555555);

snackbar.setActionTextColor(0XFF0277BD);

Настройка визуального вида Snackbar в Android и Java

Checkbox

Элементы Checkbox представляют собой флажки, которые могут находиться в отмеченном и неотмеченном состоянии. Флажки позволяют производить множественный выбор из нескольких значений. Итак, определим в файле разметки activity_main.xml элемент CheckBox:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <CheckBox android:id="@+id/enabled"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Включить"
        android:textSize="26sp"

        android:onClick="onCheckboxClicked"

        app:layout_constraintLeft_toLeftOf="parent"
```

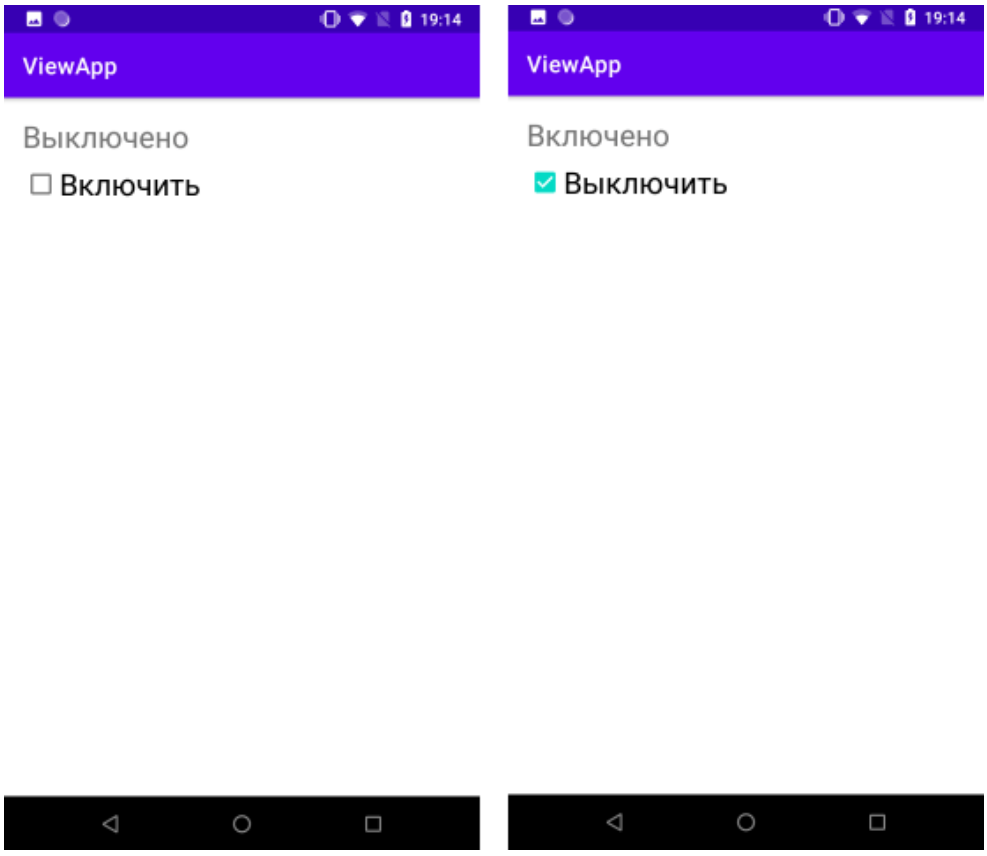
```
        app:layout_constraintTop_toBottomOf="@+id/selection"/>


</androidx.constraintlayout.widget.ConstraintLayout>
```

Атрибут android:onClick, как и в случае с простыми кнопками, позволяет задать обработчик нажатия на флажок. Определим обработчик нажатия в коде MainActivity:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

    public void onCheckboxClicked(View view) {
        // Получаем флажок
        CheckBox checkBox = (CheckBox) view;
        TextView selection = findViewById(R.id.selection);
```

```
        // Получаем, отмечен ли данный флажок
        if(checkBox.isChecked()) {
            selection.setText("Включено");
            checkBox.setText("Выключить");
        }
        else {
            selection.setText("Выключено");
            checkBox.setText("Включить");
        }
    }
}
```

В качестве параметра в обработчик нажатия onCheckboxClicked передается нажатый флажок. Обработчик срабатывает при каждом нажатии на checkBox. То есть и когда мы устанавливаем флажок, и когда мы снимем отметку. С помощью метода isChecked() можно узнать, выделен ли флажок - в этом случае метод возвращает true.



Элемент Checkbox в Android и Java

Подобным образом можно использовать несколько флажков:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <CheckBox android:id="@+id/java"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Java"
        android:textSize="26sp"

        android:onClick="onCheckboxClicked"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection"/>

    <CheckBox android:id="@+id/kotlin"
```

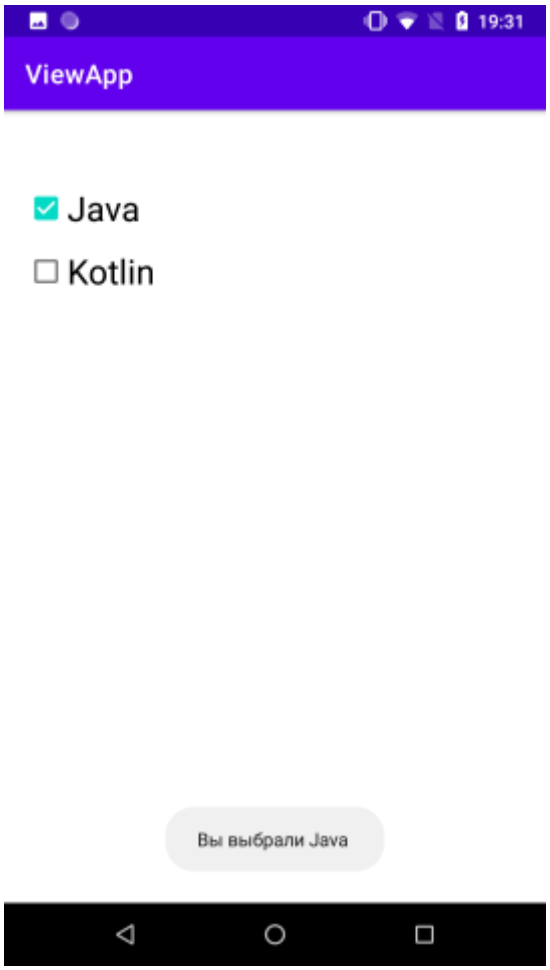
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Kotlin"
        android:textSize="26sp"

        android:onClick="onCheckboxClicked"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/java"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

На каждый флажок можно повесить свой обработчик нажатия. А можно сделать один, как в данном случае. В этом случае мы можем обработать несколько флажков в коде java с помощью конструкции switch...case

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onCheckboxClicked(View view) {
        // Получаем флажок
        CheckBox checkBox = (CheckBox) view;
        // Получаем, отмечен ли данный флажок
        boolean checked = checkBox.isChecked();

        TextView selection = findViewById(R.id.selection);

         // Смотрим, какой именно из флажков отмечен
        switch(view.getId()) {
            case R.id.java:
                if (checked)
                    Toast.makeText(this, "Вы выбрали Java
",Toast.LENGTH_LONG).show();
                break;
            case R.id.kotlin:
                if (checked)
                    Toast.makeText(this, "Вы выбрали
Kotlin",Toast.LENGTH_LONG).show();
                break;
            default:
                selection.setText("");
        }
    }
}
```

С помощью конструкции switch...case можно получить id нажатого флажка и выполнить соответствующие действия.



Элемент Checkbox в Android

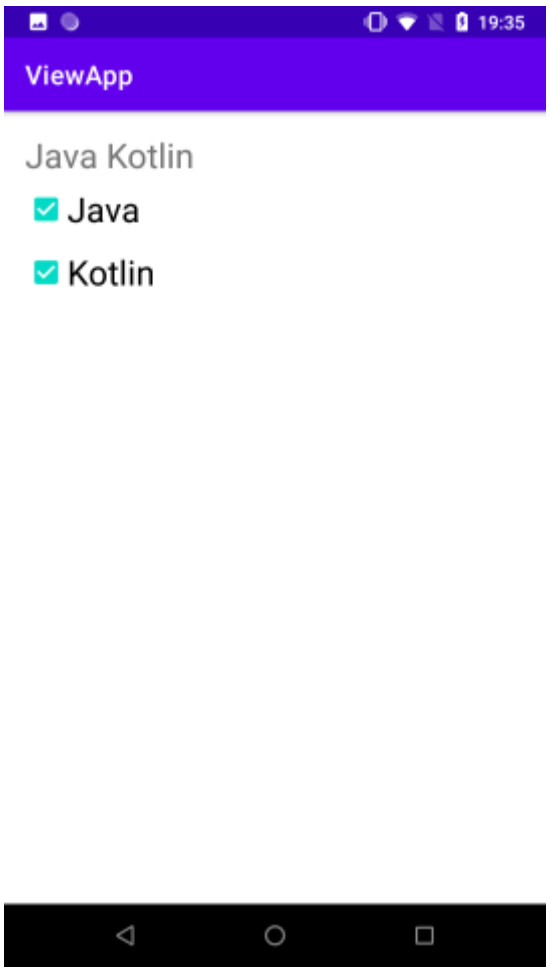
Правда, если нам просто надо взять текст из выбранного флажка, то необязательно в данном случае использовать конструкцию switch, так как мы можем сократить весь код следующим образом:

```
public void onCheckboxClicked(View view) {
    // Получаем флажок
    CheckBox language = (CheckBox) view;
    // Получаем, отмечен ли данный флажок
    TextView selection = findViewById(R.id.selection);
    if(language.isChecked())
        selection.setText(language.getText());
}
```

Однако в данном случае остается проблема: в текстовом поле отображается только один выделенный элемент. Изменим код MainActivity, чтобы отображать оба выделенных элемента:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

    public void onCheckboxClicked(View view) {

        // Получаем флажки
        CheckBox java = findViewById(R.id.java);
        CheckBox kotlin = findViewById(R.id.kotlin);
        String selectedItems = "";
        if(java.isChecked())
            selectedItems +=java.getText() + " ";
```

```
        if(kotlin.isChecked())
            selectedItems +=kotlin.getText();

        TextView selection = findViewById(R.id.selection);
        selection.setText(selectedItems);
    }
}
```



Выбор CheckBox в Android

OnCheckedChangeListener

Применение слушателя **OnCheckedChangeListener** представляет альтернативный способ отслеживания изменения флажка. Этот слушатель

срабатывает, когда мы устанавливаем или убираем отметку на флажке. Например, определим следующий checkbox:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <CheckBox android:id="@+id/enabled"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Включить"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

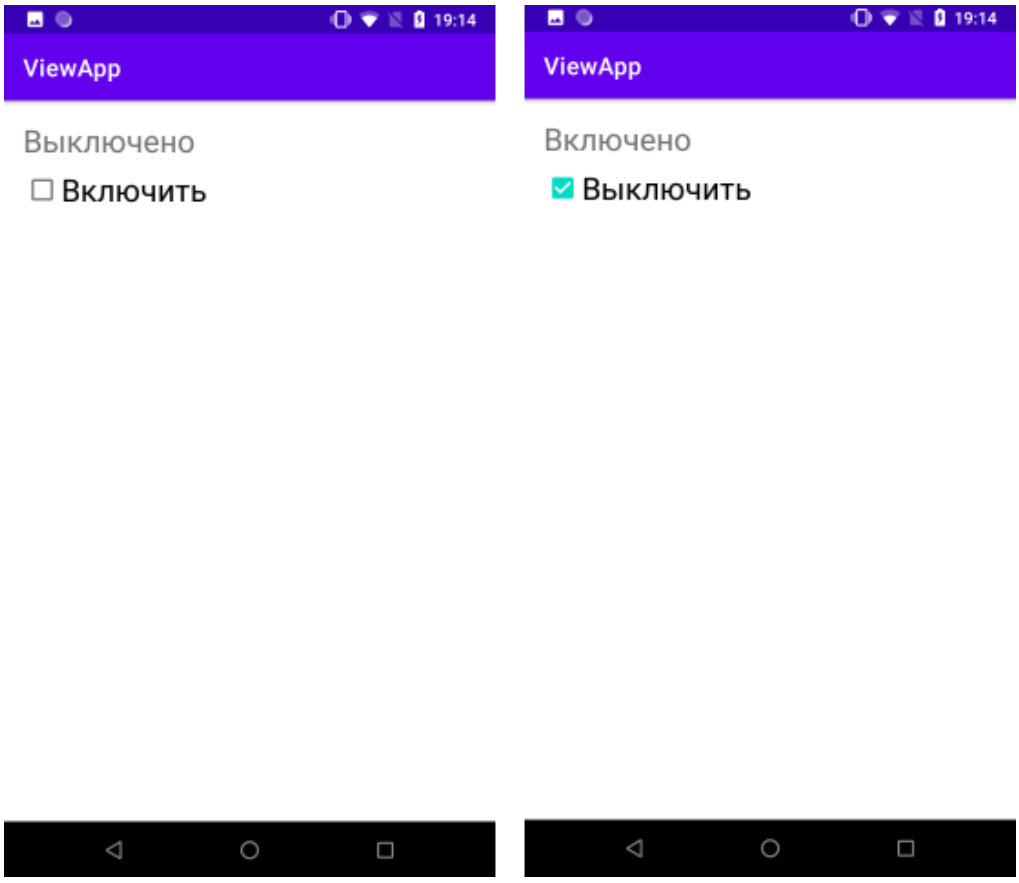
В коде MainActivity подключим обработчик изменения состояния:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        TextView selection = findViewById(R.id.selection);
        CheckBox enableBox = findViewById(R.id.enabled);

        enableBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {

                if(isChecked) {
                    selection.setText("Включено");
                    buttonView.setText("Выключить");
                }
```

```
            else {
                selection.setText("Выключено");
                buttonView.setText("Включить");
            }
        }
    });
  }
}
```

Слушатель OnCheckedChangeListener определен в базовом классе CompoundButton и определяет один метод - onCheckedChanged. Первый параметр этого метода buttonView - сам измененный флажок CheckBox. А второй параметр isChecked указывает, отмечен ли флажок.

При изменении состояния флажка будет выводиться во всплывающем окне соответствующее уведомление:



CheckBox и OnCheckedChangeListener в Android

ToggleButton

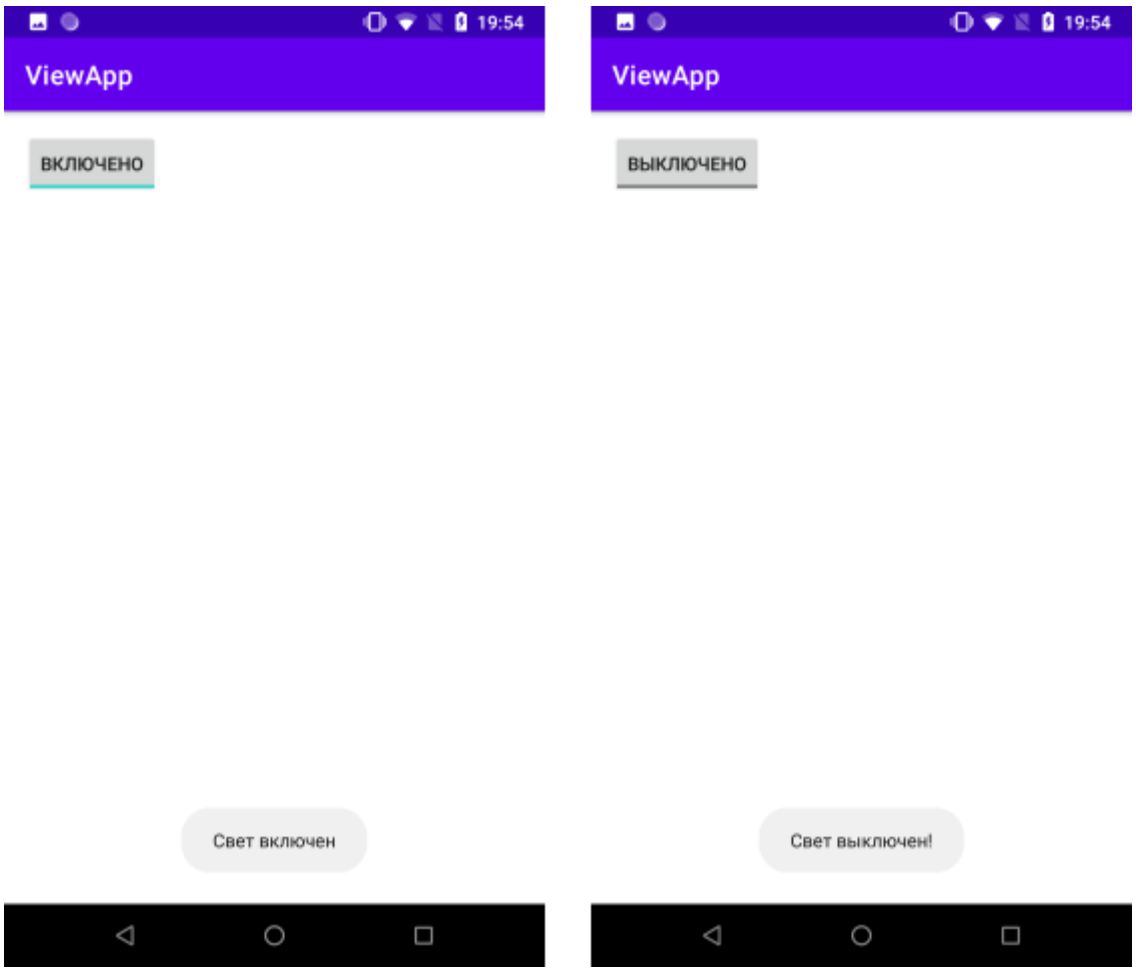
ToggleButton подобно элементу CheckBox может пребывать в двух состояниях: отмеченном и неотмеченном, причем для каждого состояния мы можем отдельно установить свой текст. Например, определим следующий элемент ToggleButton:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <ToggleButton
        android:id="@+id/toggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOn="Включено"
        android:textOff="Выключено"
        android:onClick="onToggleClicked"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
 </androidx.constraintlayout.widget.ConstraintLayout>
```

Атрибуты android:textOn и android:textOff задают текст кнопки в отмеченном и неотмеченном состоянии соответственно. И также, как и для других кнопок, мы можем обработать нажатие на элемент с помощью события onClick. В этом случае определим в классе Activity обработчик события:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onToggleClicked(View view) {

        // включена ли кнопка
        boolean on = ((ToggleButton) view).isChecked();
        if (on) {
            // действия если включена
            Toast.makeText(this, "Свет включен", Toast.LENGTH_LONG).show();
        } else {
            // действия, если выключена
            Toast.makeText(this, "Свет выключен!",
Toast.LENGTH_LONG).show();
        }
    }
}
```



ToggleButton в Android в отмеченном состоянии

Создание элемента ToggleButton в коде java:

```
package com.example.viewapp;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        ConstraintLayout layout = new ConstraintLayout(this);
        ConstraintLayout.LayoutParams layoutParams = new
ConstraintLayout.LayoutParams
                (ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        ToggleButton toggleButton = new ToggleButton(this);
        toggleButton.setTextOff("Выключено");
        toggleButton.setTextOn("Включено");
        toggleButton.setText("Выключено");
        toggleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                boolean on = ((ToggleButton) view).isChecked();


                if (on) {
                    Toast.makeText(getApplicationContext(), "Свет включен",
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(getApplicationContext(), "Свет выключен!",
Toast.LENGTH_LONG).show();
                }
            }
        });
        layoutParams.leftToLeft = ConstraintLayout.LayoutParams.PARENT_ID;
        layoutParams.topToTop = ConstraintLayout.LayoutParams.PARENT_ID;
        layout.addView(toggleButton);
```

```
        setContentView(layout);
    }
}
```

RadioButton

Схожую с флажками функциональность предоставляют переключатели, которые представлены классом RadioButton. Но в отличие от флажков единовременно в группе переключателей мы можем выбрать только один переключатель.

Чтобы создать список переключателей для выбора, вначале надо создать объект RadioGroup, который будет включать в себя все переключатели:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

   <TextView android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <RadioGroup
        android:id="@+id/radios"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

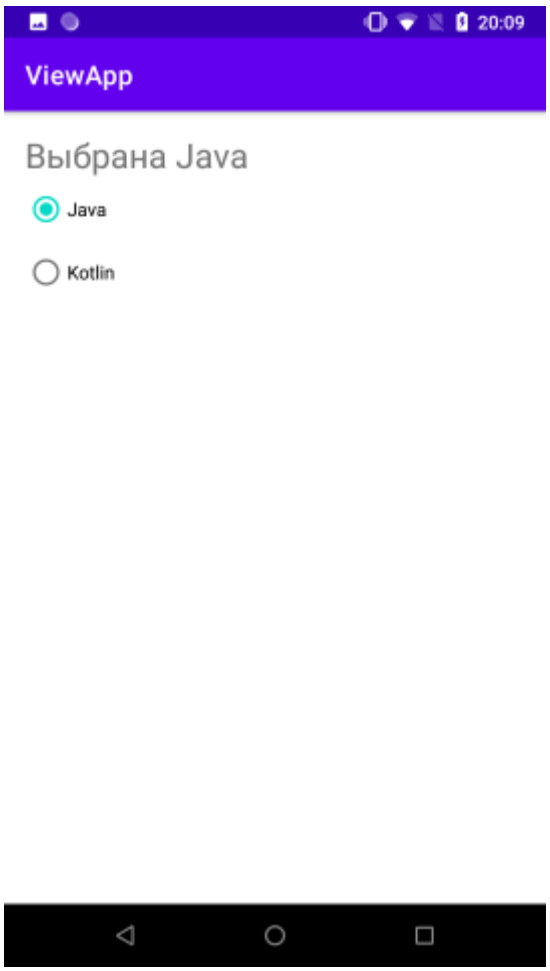
```
        android:orientation="vertical"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection"
        >

        <RadioButton android:id="@+id/java"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Java"
            android:onClick="onRadioButtonClicked"/>
        <RadioButton android:id="@+id/kotlin"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Kotlin"
            android:onClick="onRadioButtonClicked"/>
    </RadioGroup>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Поскольку класс RadioGroup является производным от LinearLayout, то мы также можем задать вертикальную или горизонтальную ориентацию списка, при том включив в него не только собственно переключатели, но и другие объекты, например, кнопку или TextView.

В классе MainActivity определим обработку выбора переключателей:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
```

```
import android.widget.RadioButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onRadioButtonClicked(View view) {
        // если переключатель отмечен
        boolean checked = ((RadioButton) view).isChecked();
        TextView selection = findViewById(R.id.selection);
        // Получаем нажатый переключатель
        switch(view.getId()) {
            case R.id.java:
                if (checked){
                    selection.setText("Выбрана Java");
                }
                break;
            case R.id.kotlin:
                if (checked){
                    selection.setText("Выбран Kotlin");
                }
                break;
        }
    }
}
```



Элемент RadioButton в Android 7

OnCheckedChangeListener

Кроме обработки нажатия на каждый отдельный переколючатель мы можем в целом повесить на весь RadioGroup с его переключателями слушатель **OnCheckedChangeListener** и обрабатывать в нем нажатия. Для этого уберем из разметки у переключателей атрибуты android:onClick, а у элемента RadioGroup определим id:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
```

```
  <TextView android:id="@+id/selection"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="26sp"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>

   <RadioGroup
      android:id="@+id/radios"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/selection">
      <RadioButton android:id="@+id/java"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Java" />
      <RadioButton android:id="@+id/kotlin"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Kotlin" />
   </RadioGroup>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Далее в коде MainActivity повесим на объект RadioGroup слушатель **OnCheckedChangeListener:**

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.RadioGroup;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // получаем объект RadioGroup
        RadioGroup radGrp = (RadioGroup)findViewById(R.id.radios);
        // обработка переключения состояния переключателя
        radGrp.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup arg0, int id) {
                TextView selection = findViewById(R.id.selection);
                switch(id) {
                    case R.id.java:
                        selection.setText("Выбрана Java");
                        break;
                    case R.id.kotlin:
```

```
                selection.setText("Выбран Kotlin");
                break;
            default:
                break;
        }
    });
  }
}
```

Слушатель RadioGroup.OnCheckedChangeListener определяет метод

onCheckedChanged(), в который передается объект RadioGroup и id выделенного переключателя. Далее также мы можем проверить id и выполнить определенную обработку.

DatePicker

DatePicker представляет элемент для выбора даты. Среди его атрибутов можно отметить следующие:

- android:calendarTextColor: цвет текста календаря
- android:calendarViewShown: указывает, будет ли отображаться вид календаря
- android:datePickerMode: устанавливает режим выбора даты
- android:dayOfWeekBackground: устанавливает фоновый цвет панели выбора дня недели
- android:endYear: устанавливает последний отображаемый год
- android:firstDayOfWeek: устанавливает первый день недели
- android:headerBackground: устанавливает фоновый цвет для панели выбранной даты
- android:maxDate: устанавливает максимальную отображаемую дату в формате mm/dd/yyyy

- android:minDate: устанавливает минимальную отображаемую дату в формате mm/dd/yyyy

- android:spinnersShown: указывает, будет ли отображаться спиннер в виджете
- android:startYear: устанавливает начальный отображаемый год
- android:yearListSelectorColor: устанавливает цвет для поля выбора года

Среди методов DatePicker можно отметить следующие:

- int getDayOfMonth(): возвращает номер выбранного дня
- int getMonth(): возвращает номер выбранного месяца (от 0 до 11)
- int getYear()(): возвращает номер выбранного года
- void init(int year, int monthOfYear, int dayOfMonth, DatePicker.OnDateChangedListener onDateChangedListener): устанавливает начальную дату. Последний параметр устанавливает слушатель изменения выбранной даты
- void setOnDateChangedListener(DatePicker.OnDateChangedListener onDateChangedListener): устанавливает слушатель изменения выбранной даты
- void setFirstDayOfWeek(int firstDayOfWeek): устанавливает первый день недели
- void updateDate(int year, int month, int dayOfMonth): программно обновляет выбранную дату

Пусть в **activity_main.xml** определен элемент DatePicker:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/dateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Применим некоторые методы DatePicker для управления его поведением:

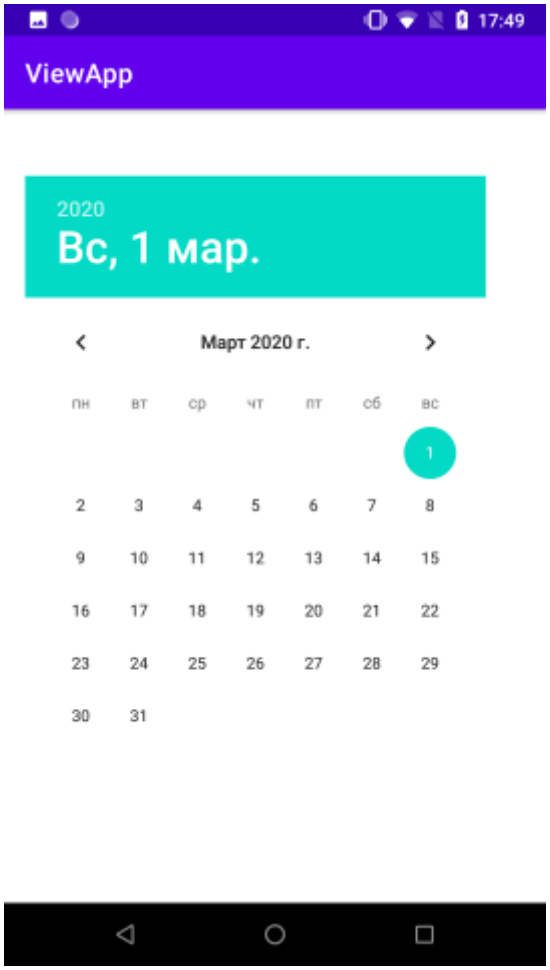
```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.DatePicker;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        TextView dateTextView = findViewById(R.id.dateTextView);
```

```
        DatePicker datePicker = this.findViewById(R.id.datePicker);


        // Месяц начиная с нуля. Для отображения добавляем 1.
        datePicker.init(2020, 02, 01, new DatePicker.OnDateChangedListener() {
            @Override
            public void onDateChanged(DatePicker view, int year, int monthOfYear,
int dayOfMonth) {


                // Отсчет месяцев начинается с нуля. Для отображения добавляем 1.
                dateTextView.setText("Дата: " + view.getDayOfMonth() + "/" +
                        (view.getMonth() + 1) + "/" + view.getYear());


                // альтернативная запись
                // dateTextView.setText("Дата: " + dayOfMonth + "/" + (monthOfYear
+ 1) + "/" + year);
            }
        });
    }
}
```

Используя метод datePicker.init(); устанавливаем дату по умолчанию - 1 марта 2020 года, так как отсчет месяцев идет с нуля. Кроме того, с помощью последнего параметра - объекта DatePicker.OnDateChangedListener устанавливается обработка выбора даты. Каждый раз, когда пользователь будет выбирать дату, будет срабатывать метод onDateChanged() объекта DatePicker.OnDateChangedListener. Этот метод принимает четыре параметра - view (элемент DatePicker), year (выбранный год), monthOfYear (выбранный месяц), dayOfMonth (выбранный день).

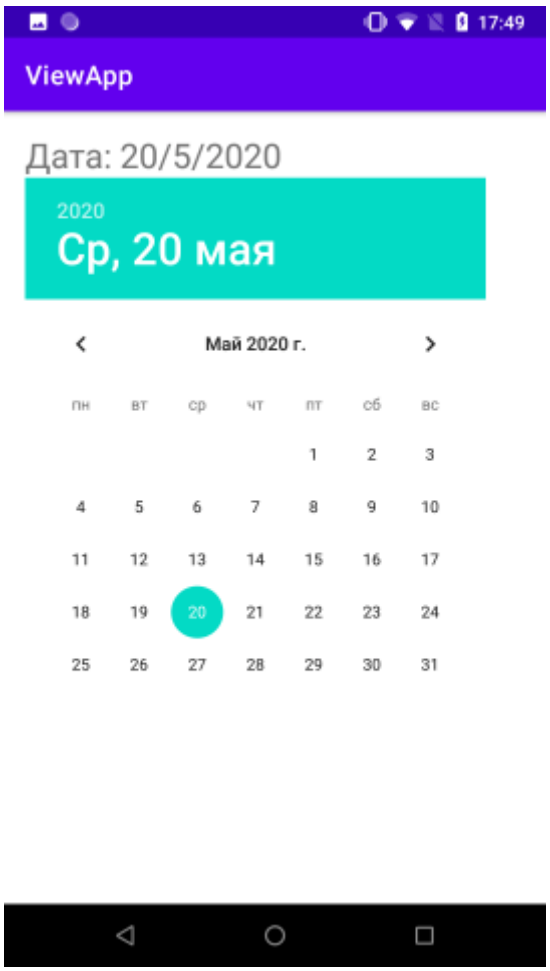
Далее мы можем получить выбранные день, месяц и год. Причем для можно использовать как параметры метода onDateChanged, так и методы самого DatePicker

Начальное состояние перед выбором - установлена дата 1 марта 2020 года.



Отображение даты и DatePicker в Android и Java

Выбор произвольной даты (20 мая 2020 года):



OnDateChangedListener и DatePicker в Android и Java

DatePicker по умолчанию отображается в режиме календаря, но мы можем использовать добавить другой режим - спиннер с помощью атрибута android:datePickerMode:

```
<DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:datePickerMode="spinner"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
```

Spinner в DatePicker в Android и Java

В данном случае спиннер отображается слева от календаря. Если мы вовсе не хотим отображать календаря, то можно установить атрибут android:calendarViewShown="false"

```
<DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:datePickerMode="spinner"
        android:calendarViewShown="false"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView" />
```

TimePicker

TimePicker представляет виджет для выбора времени, который может отображать время либо в 24-часовом, либо в 12-часовом формате.

Среди атрибутов TimePicker следует выделить **timePickerMode**, который позволяет режим отображения и может принимать одно из двух значений: **clock** (отображение в виде часов) и **spinner** (отображение в виде спиннера).

Среди методов TimePicker можно отметить следующие:

- int getHour(): возвращает час (в 24-часом формате)
- int getMinute(): возвращает минуты
- boolean is24HourView(): возвращает true, если используется 24-часовой формат
- void setHour(int hour): устанавливает час для TimePicker
- void setIs24HourView(Boolean is24HourView): устанавливает 24-часовой формат
- void setMinute(int minute): устанавливает минуты
- void setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener): устанавливает слушатель изменения времени в TimePicker в виде объекта **TimePicker.OnTimeChangedListener**

Определим TimePicker в **activity_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/timeTextView"
        android:layout_width="wrap_content"
```

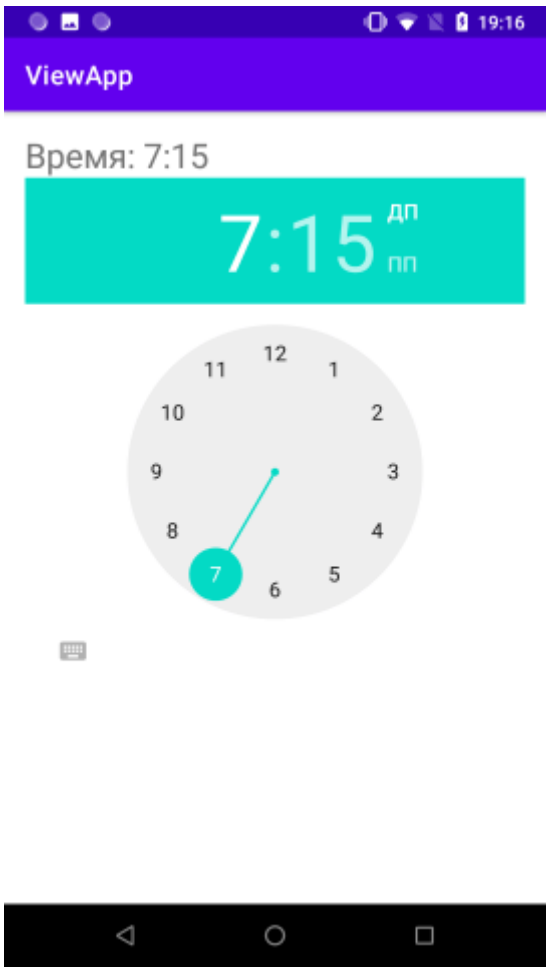
```
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/timeTextView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Применим некоторые методы TimePicker для управления его поведением:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.TextView;
import android.widget.TimePicker;

public class MainActivity extends AppCompatActivity {

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

```
        setContentView(R.layout.activity_main);


        TextView timeTextView = findViewById(R.id.timeTextView);
        TimePicker timePicker = findViewById(R.id.timePicker);


        timePicker.setOnTimeChangedListener(new
TimePicker.OnTimeChangedListener() {
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDay, int minute)
{


                timeTextView.setText("Время: " + hourOfDay + ":" + minute);
                // или так
                // timeTextView.setText("Время: " + view.getHour() + ":" +
view.getMinute());
            }
        });
    }
}
```

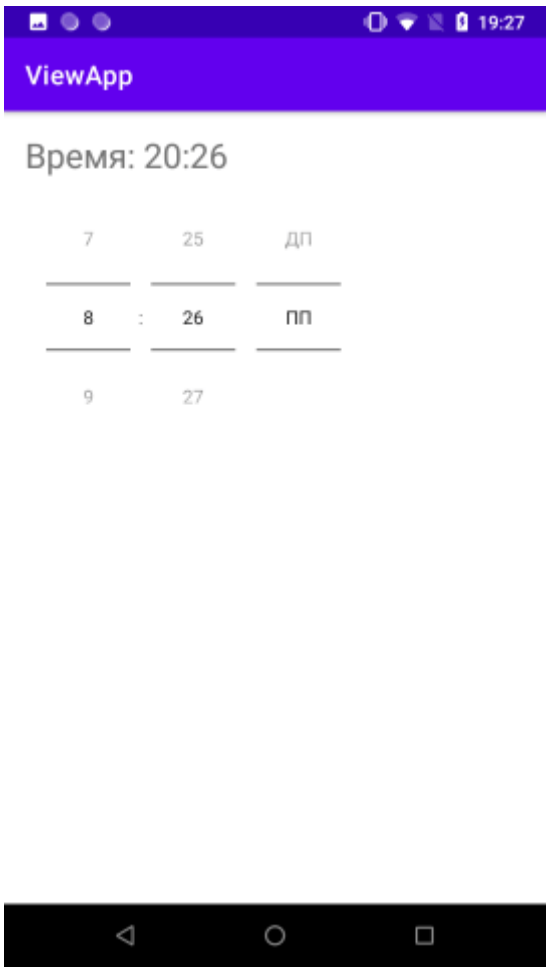
Для добавления слушателя изменения времени в TimePicker применяется метод setOnTimeChangedListener(), в который передается объект **TimePicker.OnTimeChangedListener**. Он имеет один метод - **onTimeChanged()**, который вызывается при каждом изменении времени в TimePicker. Этот метод принимает три параметра - сам элемент TimePicker, hourOfDay - установленный час и minute - установленные минуты. В данном случае просто передаем значение выбранного времени в TextView.



OnTimeChangedListener в TimePicker в Android и Java

По умолчанию TimePicker отображается в режиме "clock" или часы. Применим режим "spinner":

```
<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="spinner"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/timeTextView" />
```



Spinner в TimePicker в Android и Java

Ползунок SeekBar

Элемент SeekBar выполняет роль ползунка, то есть шкалу делений, на которой мы можем менять текущую отметку.

Среди его атрибутов можно отметить следующие:

- android:max: устанавливает максимальное значение
- android:min: устанавливает минимальное значение
- android:progress: устанавливает текущее значение, которое находится в диапазоне между минимальным и максимальным

Для управления SeekBar определяет ряд методов, из которых выделим следующие:

- void setProgress(int progress): устанавливает текущее значение ползунка
- void setMin(int min): устанавливает минимальное значение
- void setMax(int max): устанавливает максимальное значение

void incrementProgressBy(int diff): увеличивает текущее значение на diff

- int getMax(): возвращает максимальное значение
- int getMin(): возвращает минимальное значение
- int getProgress(): возвращает текущее значение
- void setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener l): устанавливает слушателя изменения значения в SeekBar

Определим SeekBar в разметке layout:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
```

```
    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:progress="20"
        android:max="50"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Атрибут android:progress задает число 20 в качестве текущего значения ползунка, а атрибут android:max - максимально возможное значение - число 50. В итоге мы получим следующий элемент:

Элемент SeekBar в Android

Теперь используем метод **setOnSeekBarChangeListener()**, который позволяет установить обработчики событий изменения значения ползунка. Так, определим в файле layout следующий код:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView android:id="@+id/seekBarValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:progress="20"
        android:max="50"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/seekBarValue" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определен элемент TextView, который будет выводить текущее значение ползунка при его изменении.

И изменим код MainActivity:

```
package com.example.viewapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.TimePicker;

public class MainActivity extends AppCompatActivity {

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

        SeekBar seekBar = findViewById(R.id.seekBar);
        TextView textView = findViewById(R.id.seekBarValue);
        seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
```

```
            textView.setText(String.valueOf(progress));
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });
  }
}
```

В метод setOnSeekBarChangeListener() передается объект SeekBar.OnSeekBarChangeListener, который позволяет установить три метода-обработчика:

- onProgressChanged: срабатывает при перетаскивании ползунка по шкале. Передаваемый в метод параметр progress позволяет получить новое значение ползунка, которое в данном случае передается в TextView для отображения на экране
- onStartTrackingTouch: срабатывает при начале перетаскивания ползунка по шкале
- onStopTrackingTouch: срабатывает при завершении перетаскивания ползунка по шкале

Элемент SeekBar в Android

Также мы можем получить текущее значение ползунка, использовав метод getProgress():

```
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
{
    textView.setText(String.valueOf(seekBar.getProgress()));
}
```

Задание

Создать примеры, реализующие основные элементы управления:

- Элемент TextView и его атрибуты. Установка элемента в коде. Используя атрибут **android:autoLink** вывести на экран ссылку и телефон
- Элемент EditText и его атрибуты. Используя атрибуты **android:hint** и **android:inputType** задать текст, который будет отображаться в качестве подсказки, если элемент **EditText** пуст и клавиатуру для ввода. Реализовать два поля. Первое поле - однострочное, а второе –

многострочное. Введенные символы в первом поле – отображаются во втором.

- Элемент Button и его атрибуты. Реализовать на экране кнопку с надписью “Ввод”. После нажатия на кнопку выводится текст из первого поля во второе. Реализовать аналогичный пример полностью в коде
- Класс Toast. Всплывающие окна. Toast можно использовать только в коде java. Реализуйте это, используя метод **Toast.makeText().** В качестве времени показа окна можете использовать целочисленное значение - колическо миллисекунд или встроенные константы **Toast.LENGTH_LONG** (2000 миллисекунд) и **Toast.LENGTH_SHORT** (2000 миллисекунд). Используйте метод setGravity для указания, в какой части контейнера надо позиционировать Toast,
- Элемент Snackbar. Реализуйте пример с помощью метода **make().** Прикрепление обработчика события. Реализуйте пример с помощью метода setAction().Реализуйте пример с настройкой визуального вида
- Элементы Checkbox. Реализуйте пример с несколькими флажками которые могут находиться в отмеченном и неотмеченном состоянии.
- Слушатель OnCheckedChangeListener. Реализуйте пример с помощью метода onCheckedChanged.
- Элемент ToggleButton. . Реализуйте пример с помощью атрибутов android:textOn и android:textOff. Реализуйте пример создания элемента ToggleButton в коде java.
- Класс RadioButton Реализуйте пример
- Элемент DatePicker. Реализуйте пример.
- Элемент TimePicker Реализуйте пример
- Элемент SeekBar (Ползунок). Реализуйте пример