Эргономика мобильных приложений

Преподаватель: **Меженная Марина Михайловна**

к.т.н., доцент,
доцент кафедры инженерной психологии и эргономики
а 609-2
mezhennaya@bsuir.by

Лекция 4: Программируем в MainActivity.java

План лекции:

1.Подключение графического представления к Activity.

2.Доступ к элементам экрана из кода.

3.Обработчик событий: 4 способа.

4.Горячие клавиши Android Studio.

Файл MainActivity.java:
Подключение графического представления к Activity

При запуске деятельности система должна получить ссылку на корневой узел дерева разметки, который будет использоваться для прорисовки графического интерфейса на экране мобильного устройства. Для этого в методе onCreate ( ) необходимо вызвать метод setContentView ( ).

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

Файл MainActivity.java:
Подключение графического представления к Activity

Метод **setContentView(int)** – устанавливает содержимое Activity из layout-файла. Но в качестве аргумента мы указываем не путь к layout-файлу (res/layout/activity_main.xml), а константу, которая является ID файла и хранится в файле **R.java**.

Имена этих ID-констант совпадают с именами файлов ресурсов (без расширений).

Можно создать новый xml-файл в папке res > layout и прописать его вместо activity_main в методе setContentView(int). После запуска приложения отобразится новый файл разметки.

Файл MainActivity.java:
Доступ к элементам экрана из кода

Чтобы обратиться к элементу экрана из кода, нам нужен его **ID**. Он прописывается в Properties либо в layout-файлах.

Для ID существует четкий формат - **@+id/name**,

где + означает, что это новый ресурс и он должен добавиться в **R.java** класс, если он там еще не существует.

Давайте уникальные и осмысленные ID-имена!!!

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    android:id="@+id/btnOK"
    android:layout_gravity="center_horizontal"
    android:layout_margin="10dp" />
```

Файл MainActivity.java:
Доступ к элементам экрана из кода

Зная ID View-элемента, обратиться к нему из кода можно по константе **R.id.btnOK.**

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    android:id="@+id/btnOK"
    android:layout_gravity="center_horizontal"
    android:layout_margin="10dp" />
```

Для этого нам понадобится метод **findViewById:**

View btnOK = findViewById(R.id. btnOK); // находим View

Button btnOK = (Button) findViewById(R.id.btnOK); // приводим к подклассу Button

Файл MainActivity.java:
Доступ к элементам экрана из кода:
пример реализации

```
public class MainActivity extends ActionBarActivity {

    private Button btnOK;

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

    private void initViews() {
        btnOK = (Button) findViewById(R.id.btnOK);
    }
```

Файл MainActivity.java:
Доступ к элементам экрана из кода:
пример реализации

1. В файле MainActivity.java объявите объекты от подклассов используемых Вами графических элементов.

2. В методе onCreate файла MainActivity.java вызовите метод (например, initViews()), в котором Вы будете подключать к объявленным ранее объектам Ваши графические элементы.

! initViews() – это не переопределяемый метод, это наш собственный метод; его целесообразно использовать, чтобы вынести код подключения графики и не загружать метод onCreate. Но он не является обязательным! Можно все писать в onCreate…

3. В методе initViews() ставим в соответствие объявленным переменным Ваши графические элементы.

Файл MainActivity.java:
Доступ к элементам экрана из кода

Далее можно работать с графическими элементами из кода, например:

```
btnOK.setText("OK"); // устанавливаем новый текст на кнопку

btnOK.setEnabled(false); // делаем кнопку неактивной
```

Файл MainActivity.java:
Обработчик событий
1. Самая простая реализация обработчика

В layout-файле (main.xml) при описании кнопки пишем:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    android:id="@+id/btnOK"
    android:onClick="onClickOK" />
```

Т.е. используем атрибут onClick. В нем указываем имя метода из Activity. Этот метод и сработает при нажатии на кнопку.

Далее, добавляем этот метод в Activity (MainActivity.java). Требования к методу: public, void и на вход принимает View:

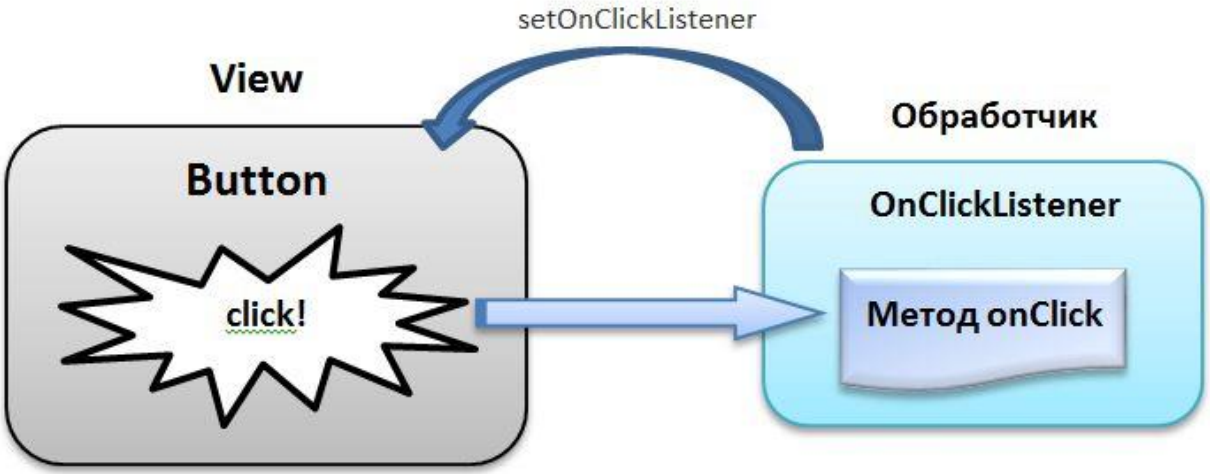
```
public void onClickOK(View v) {
  // действия при нажатии на кнопку
}
```

Файл MainActivity.java:
Обработчик событий
2. Свой обработчик для каждого View-элемента

Для реализации необходимо выполнить следующие шаги:

- создаем обработчик (объект от интерфейса View.OnClickListener),
- заполняем метод onClick (т.к. в интерфейсе он не реализован, а только заявлен),
- присваиваем обработчик кнопке (используем метод setOnClickListener).

и система обработки событий готова. Когда на кнопку нажимают, обработчик реагирует и выполняет код из метода onClick.



Файл MainActivity.java:
Обработчик событий
2. Свой обработчик для каждого View-элемента

```
private void initViews() {
    // находим View-элементы
    btnOK = (Button) findViewById(R.id.btnOK);
    // создаем обработчик нажатия
    View.OnClickListener oclBtnOK = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Меняем текст на кнопке
            btnOK.setText("Hello");
        }
    };
    // подключаем обработчик к кнопке OK
    btnOK.setOnClickListener(oclBtnOK);
}
```

Файл MainActivity.java:
Обработчик событий
3. Один обработчик для нескольких View-элементов

Присваиваем один обработчик нескольким кнопкам, а внутри обработчика определяем, какая именно кнопка была нажата.

```
private void initViews() {
    // находим View-элементы
    btnOK = (Button) findViewById(R.id.btnOK);
    btnCancel = (Button) findViewById(R.id.btnCancel);
    // создаем общий обработчик нажатия
    View.OnClickListener oclBtn = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // по id определеяем кнопку, вызвавшую этот обработчик
            switch (v.getId()) {
                case R.id.btnOK:
                    btnOK.setText("Hello");
                    break;
                case R.id.btnCancel:
                    btnCancel.setText("Goodbye");
                    break;
            }
        }
    };
    // подключаем обработчик к кнопкам
    btnOK.setOnClickListener(oclBtn);
    btnCancel.setOnClickListener(oclBtn);
}
```

Файл MainActivity.java:
Обработчик событий
4. Activity в качестве единого обработчика

Есть правило – чем меньше объектов вы создаете, тем лучше, т.к. под каждый объект выделяется память, а это достаточно ограниченный ресурс, особенно для телефонов. Поэтому создавать один обработчик для нескольких View это правильнее с точки зрения оптимизации. К тому же кода становится меньше и читать его удобнее.

Есть еще один способ создания обработчика, который вовсе не потребует создания объектов. Будет использоваться уже созданный объект – Activity.

Для этого Activity-класс должен подключить интерфейс View.OnClickListener и реализовать метод onClick.

Файл MainActivity.java: Обработчик событий

4. Activity в качестве единого обработчика

```
public class MainActivity extends ActionBarActivity implements View.OnClickListener{

    private Button btnOK, btnCancel;

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

    private void initViews() {
        // находим View-элементы
        btnOK = (Button) findViewById(R.id.btnOK);
        btnCancel = (Button) findViewById(R.id.btnCancel);
        // подключаем обработчик к кнопкам
        btnOK.setOnClickListener(this);
        btnCancel.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btnOK:
                btnOK.setText("Hello");
                break;
            case R.id.btnCancel:
                btnCancel.setText("Goodbye");
                break;
        }
    }
```

Горячие клавиши Android Studio (и др. сред разработки)

Оптимизация импорта	CTRL + ALT + O	
Поиск	CTRL + SHIFT + A	
Быстрое исправление проекта (подсказки по ошибкам)	ALT + ENTER	
Форматирование кода	CTRL + ALT + L (Win)	OPTION + CMD + L (Mac)
Показать параметры для выбранного метода	CTRL + P	
Создать метод	ALT + Insert (Win)	CMD + N (Mac)
Переопределить метод	CTRL + O	
Перейти к источнику	F4 или CTRL (Win)	CMD + down-arrow (Mac)
Удалить строку	CTRL + Y (Win)	CMD + Backspace (Mac)
Поиск по символу	CTRL + ALT + SHIFT + N (Win)	OPTION + CMD + O (Mac)

Работа с TextView

```
textViewResult.setText("0");

textViewResult.setText("-" + textViewResult.getText().toString());

textViewResult.setText(textViewResult.getText().toString().concat(newNumber));

textViewResult.setText(textViewResult.getText().toString().replace("-", ""));

double number = Double.parseDouble(textViewResult.getText().toString());

textViewResult.setText(String.valueOf(int_result));
textViewResult.setText(String.valueOf(result.doubleValue()));
```

Работа с TextView

```
public double calculateNumber() {

 double number=0;
        try
        {
            number = Double.parseDouble(textViewResult.getText().toString());
        }
        catch (Exception e)
        {
            Toast.makeText(this, "Error", Toast.LENGTH_SHORT).show();
        }
        return number;
}
```

Смена ориентации экрана

```
@Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putString("SavedMessage", tv_Message.getText().toString());
}


    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        tv_Message.setText(savedInstanceState.getString("SavedMessage"));
}
```