## Создание второго окна приложения

### Цель работы

Ознакомиться со способом создания и запуска второго окна в составе мобильного приложения.

### Методические указания

Создадим новый проект. По умолчанию, новый проект состоит из одной активити с пустым расположением.

Во-первых, создадим расположение, состоящее из одной кнопки. При нажатии на эту кнопку мы будем переходить во второе окно приложения:

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

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Go to Activity Two"

android:id="@+id/btnActTwo">

</Button>

</LinearLayout>

Создадим для этой кнопки обработчик нажатия, пока пустой:

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.btnActTwo:

// TODO Call second activity

break;

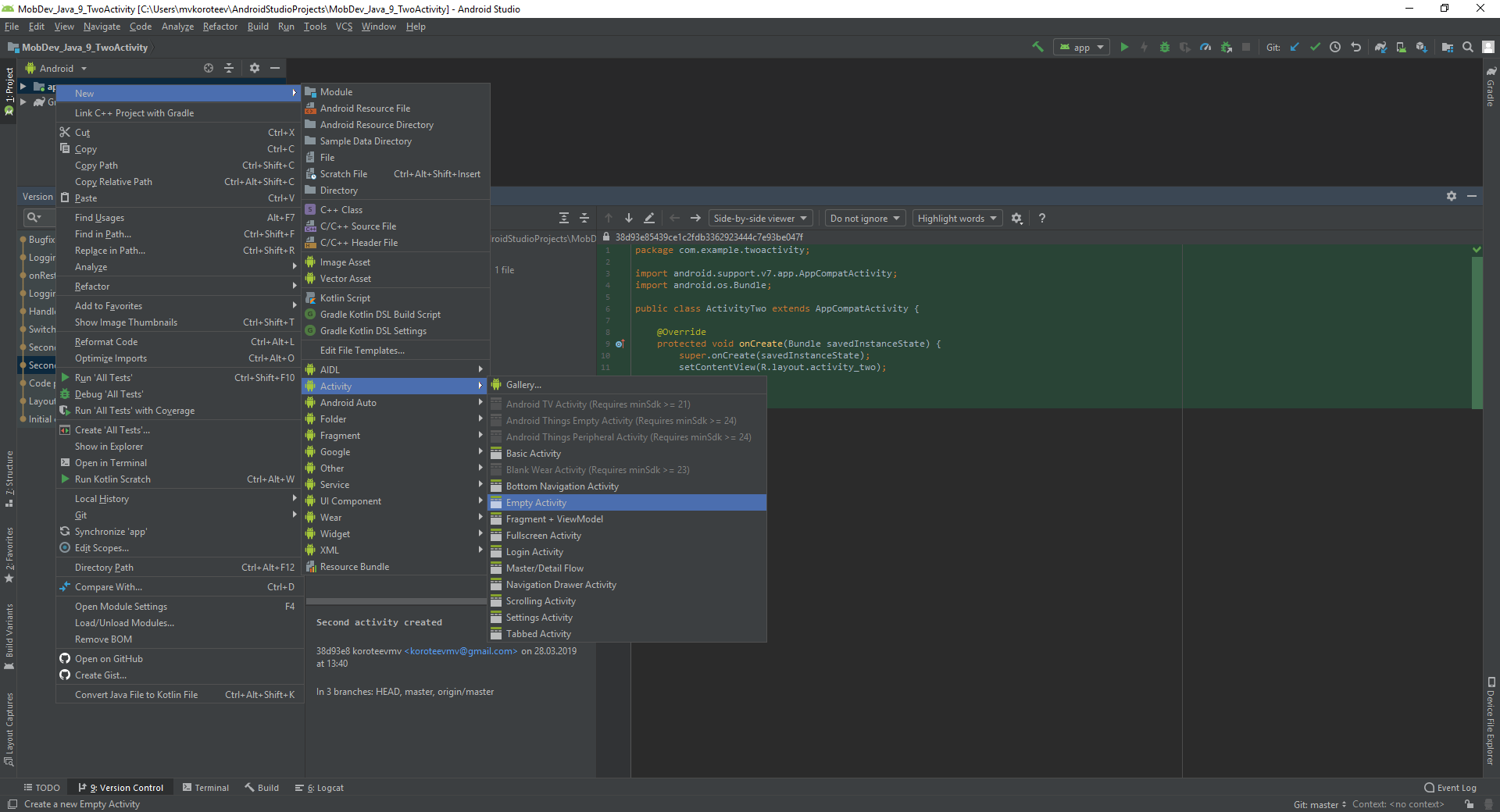
default:

break;

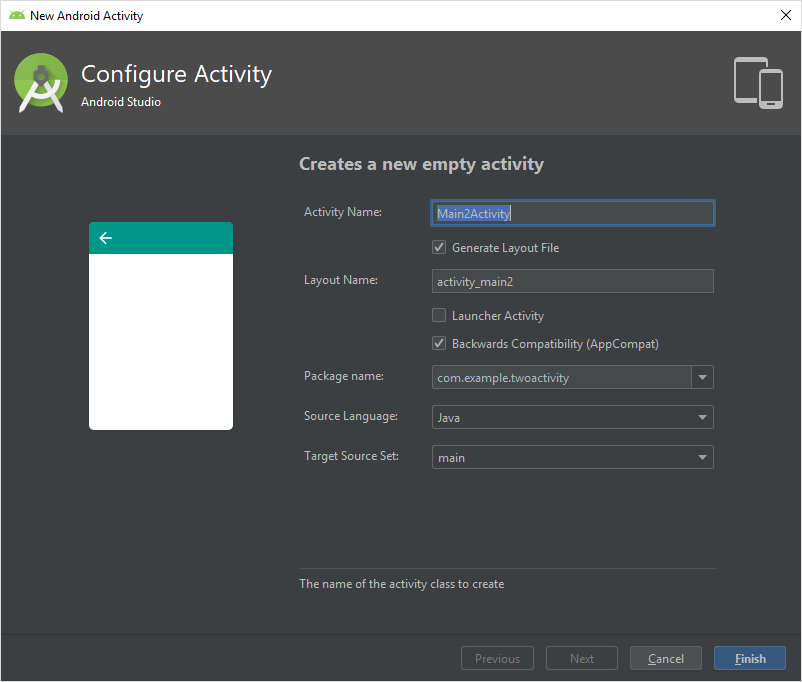
}

}

Теперь попробуем добавить новую активити. Самый простой способ это сделать - щелкнуть на проекте в панели навигатора правой кнопкой и выбрать пункт меню New -> Activity -> Empty Activity, как показано на рисунке:



Мы должны увидеть окно добавления нового окна (активити), которое весьма похоже на окно, появляющееся при создании нового проекта. Здесь мы можем ввести имя нашего нового компонента, имя файла с расположением элементов и некоторые другие параметры. Преимуществом этого способа создания новой активити является то, что студия автоматизирует весь процесс за нас, создает все необходимые файлы и вносит нужные дополнения в файл манифеста.



При нажатии на кнопку Finish мы должны увидеть два новых файла в нашем проекте - один с программным кодом и один с XML расположением. Обратите внимание, что оба окна приложения, несмотря на то, что являются независимыми компонентами приложения, существуют в одном адресном пространстве.

Обратите внимание на то, что в файле манифеста появилась новая строка:

<activity android:name=".ActivityTwo"></activity>

То есть, наша новая активити полноценно зарегистрирована в нашем приложении.

Теперь для иллюстрации создадим расположение второго окна. Создадим в нем одно текстовое поле:

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is Activity Two">

</TextView>

</LinearLayout>

Теперь для организации перехода в новое окно нам осталось дописать в обработчике тапа по кнопке создание интента и запуск активити. В данном случае воспользуемся явным вызовом - передадим в интент конкретное название только что созданной активити:

public void onClick(View v) {

switch (v.getId()) {

case R.id.btnActTwo:

Intent intent = new Intent(this, ActivityTwo.class);

startActivity(intent);

break;

default:

break;

}

}

Регистрацию обработчика тапа оставляем на самостоятельную реализацию.

Теперь при запуске приложения мы должны иметь возможность перейти в новое окно по тапе на кнопке. Обратите внимание, что для возврата к предыдущему окну нам можно воспользоваться стандартной кнопкой Назад.

Для того, чтобы лучше понять жизненный цикл активности давайте добавим в наше основное окно реализации некоторых стандартных методов жизненного цикла:

@Override

protected void onStart() {

super.onStart();

Log.*d*(TAG, "MainActivity: onStart()");

}

@Override

protected void onResume() {

super.onResume();

Log.*d*(TAG, "MainActivity: onResume()");

}

@Override

protected void onPause() {

super.onPause();

Log.*d*(TAG, "MainActivity: onPause()");

}

@Override

protected void onStop() {

super.onStop();

Log.*d*(TAG, "MainActivity: onStop()");

}

@Override

protected void onDestroy() {

super.onDestroy();

Log.*d*(TAG, "MainActivity: onDestroy()");

}

@Override

protected void onRestart() {

super.onRestart();

Log.*d*(TAG, "MainActivity: onRestart()");

}

Эти методы, как Вы помните, вызываются автоматически системой при смене состояния активности, например, при приостановке активности. В данном случае, мы в каждом из этих методов пишем в поток логов специальное сообщение, поясняющее, какой метод когда был вызван. Первый аргумент, это просто метка, позволяющая в общем потоке методов выделить созданные нами сообщения:

final String TAG = "States";

Так как метод onCreate также относится к методам жизненного цикла, добавим соответствующее сообщение и в него:

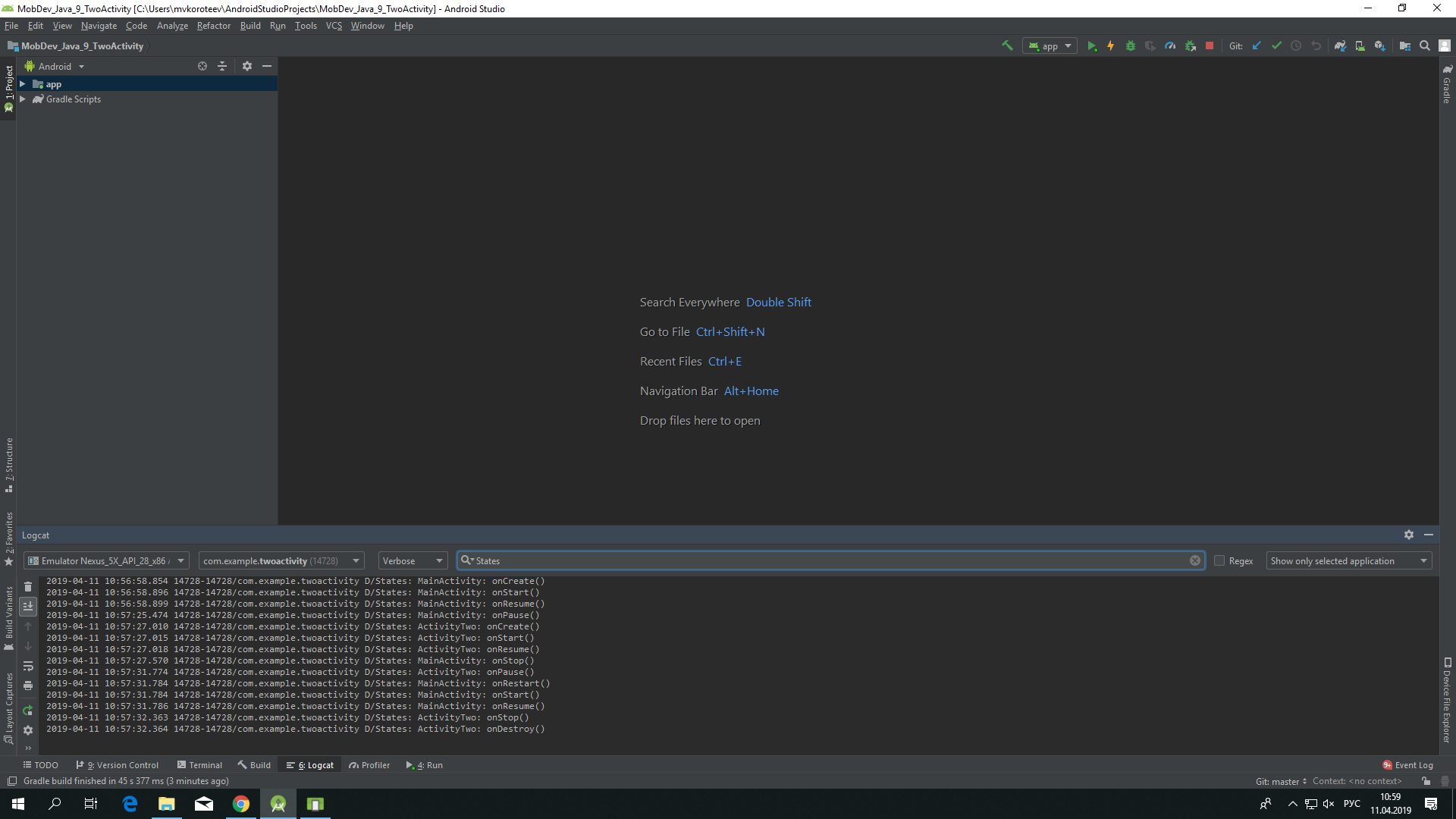
Log.*d*(TAG, "MainActivity: onCreate()");

Для более полной информации можете самостоятельно добавить соответствующие сообщения во вторую активити.

Теперь запустим наше приложение и убедимся, что все работает, как нужно:

|  |  |
| --- | --- |
|  |  |

Мы можем посмотреть диагностические сообщения, которые посылает и сама система, и студия, и непосредственно наше приложение в окне Logcat студии. Для удобства отфильтруем сообщения по нашей метке States:



Теперь вы знаете, как создавать и вызывать новое окно в составе вашего приложения.

Повторение методического материала:



### Контрольные вопросы

1. Зачем делить приложение на несколько окно? Почему нельзя использовать разные расположения?

Чтобы компоненты, неопределенные в различные промежутки времени, использовали несколько окон, что является значительно проще для реализации.

1. Что такое интент и зачем он нужен?

Намерение (Intent) - это механизм для описания одной операции - выбрать фотографию, отправить письмо, сделать звонок, запустить браузер и перейти по указанному адресу. В Android-приложениях многие операции работают через намерения.

1. Как вызвать определенное окно своего приложение? А другого?

Своего приложения:

 Другого приложения:

Изображение выглядит как текст

Автоматически созданное описание

1. Что такое таск? Почему при перемещении между окнами работает кнопка “Назад”?

Task – группа из нескольких Activity, с помощью которых пользователь выполняет определенную операцию. Обычно стартовая позиция для создания Task – это экран Домой (Home).

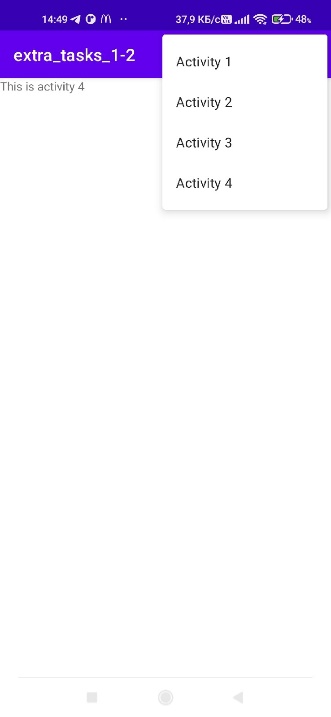
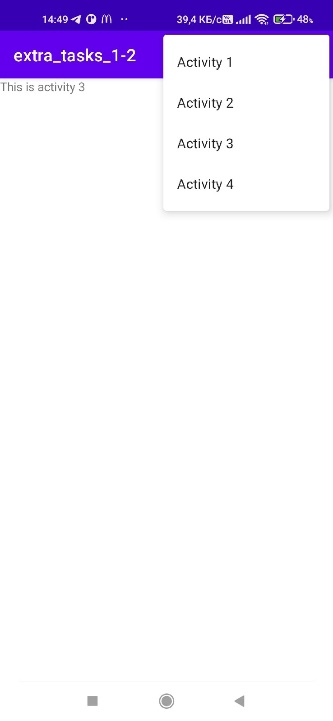
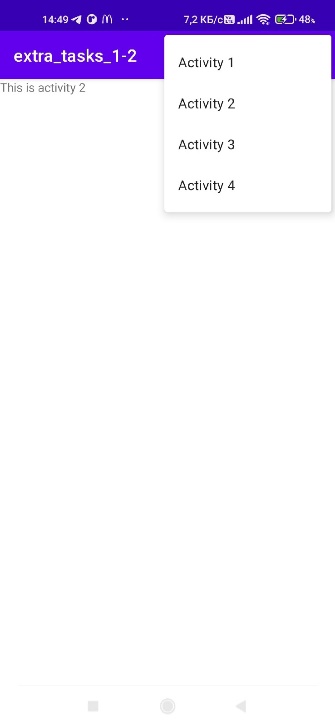
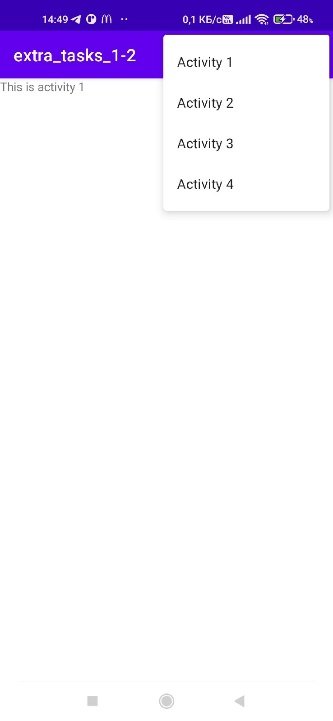
Находясь в Home вы вызываете какое-либо приложение из списка приложений или через ярлык. Создается Task. И Activity приложения (которое отмечено как MAIN в манифест-файле) помещается в этот Task как корневое. Task выходит на передний фон. Если же при вызове приложения, система обнаружила, что в фоне уже существует/

Task, соответствующий этому приложению, то она выведет его на передний план и создавать ничего не будет.

### Дополнительные задания

1. Создайте приложение, состоящее из четырех активностей и реализуйте переходы между ними.
2. Реализуйте переходы между активностями используя меню приложения. Меню должно быть описано в XML файле и быть общим для всех четырех активностей.

Работа приложения:



Activity1.java, все остальные файлы активити имеют идентичное содержание:

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

activity\_1.xml, все остальные файлы активити имеют идентичное содержание:

Изображение выглядит как текст

Автоматически созданное описание

AndroidManifest.xml:

Изображение выглядит как текст

Автоматически созданное описание

menu.xml:

Изображение выглядит как текст

Автоматически созданное описание

1. (\*) Реализуйте передачу данных между активностями. В первом окне создайте текстовое поле для ввода имени и кнопку. При тапе на кнопку должно открываться второе окно, в котором отображается имя, введенное пользователем.

Работа приложения:

Изображение выглядит как текст

Автоматически созданное описание

activity\_main.xml:

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

MainActivity.java:

Изображение выглядит как текст

Автоматически созданное описание

activity\_side.xml:

Изображение выглядит как текст

Автоматически созданное описание

SideActivity.java:

Изображение выглядит как текст

Автоматически созданное описание