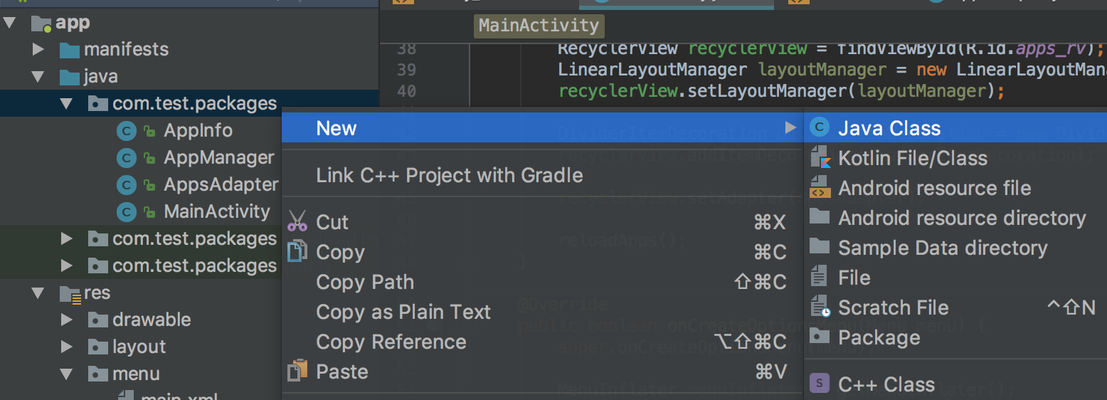
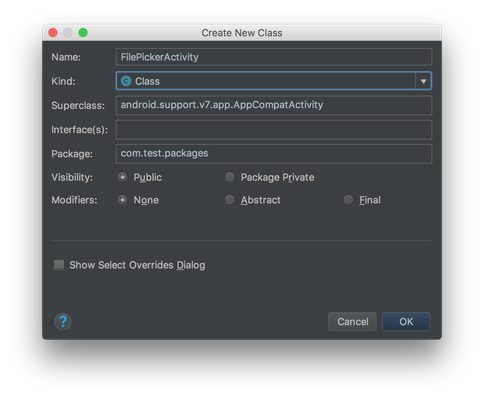
Существует не так много приложений, состоящих из всего лишь одного экрана. В более-менее серьёзных приложениях несколько экранов, а в крупных — десятки.

Как вы помните, экран приложения, то есть совокупность элементов интерфейса — это Activity. Что если мы захотели добавить в наше приложение функцию установки других приложений из APK? Для этого нам потребуется создать ещё одну Activity, в которой мы будем выбирать файл APK.

Создайте новый класс:

[](https://skillberg.com/media/uploads/2017/11/17/e7a0065b3e4140dd95f1d5afa68f9ff7.png)Создание нового класса

[](https://skillberg.com/media/uploads/2017/11/17/be7fea6bd0c44d28a8ead0e4a4e82374.png)Настройка нового класса

Для этой Acitivity так же нужно создать соответствующий файл разметки. Создайте новый лэйаут с именем activity\_file\_picker и следующим контентом:

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

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

android:layout\_width="match\_parent"

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

<android.support.v7.widget.RecyclerView

android:id="@+id/files\_rv"

android:layout\_width="match\_parent"

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

</FrameLayout>

Теперь в FilePickerActivity создайте метод onCreate():

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_file\_picker);

}

Последний шаг — осталось зарегистрировать созданную Activity в манифесте. Откройте AndroidManifest.xml. В нём вы увидите примерно следующее:

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

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

package="com.test.packages">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

Как видите, одна Activity (основная) уже зарегистрирована. Добавим новую Activity сразу после неё:

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

Эта запись Activity получилась заметно короче, потому что не содержит IntentFilter (что это такое — обсудим в следующих уроках).

Отлично, Activity полностью создана. Но как же её запустить?

Intent

В Android новые экраны запускаются не напрямую, а с использованием механизма Intent(намерение).

Intent — это специальный объект, с помощью которого мы можем сказать системе, что хотим выполнить какое-то действие. В основном Intentиспользуется для следующих трёх целей:

* Запуск Activity. Мы можем запустить Activity из своего или чужого приложения, и, опционально, получить результат выполнения.
* Запуск Service и привязка к сервису. Об этом поговорим попозже.
* Отправка Broadcast — широковещательных сообщений. Об этом тоже позже.

Нас интересует первый пункт — запуск Activity. Тут есть два способа использования Intent. Intent может быть двух типов — явный и неявный.

При использовании явного типа мы должны знать полное имя Activity, которую хотим запустить. Обычно явные Intent используются для запуска Activity из собственного приложения.

При использовании неявного типа мы сообщаем системе, какое действие мы хотим выполнить (например, открыть URL в браузере, или поделиться каким-то контентом в социальных сетях), а Android уже самостоятельно выбирает, какую Activity из какого приложения использовать, или же предоставляет этот выбор пользователю.

Запуск Activity

Мы узнали, что для запуска Activity нам нужно использовать Intent. Теперь нужно определиться, когда мы будем это делать.

Помните, как мы создавали меню? Помимо поиска мы добавили пункт с заголовком "Установить". Вот при нажатии на него мы и будем запускать новую Activity!

Создайте метод startFilePickerActivity() в MainActivity:

private void startFilePickerActivity() {

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

startActivity(intent);

}

В первой строчке мы создаем Intent, первым параметром передаем контекст, вторым — класс той Activity, которую хотим запустить.

Во второй строчке сообщаем системе, что хотим запустить Activity, используя созданный нами Intent.

Теперь добавьте вызов этого метода в onOptionsItemSelected():

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.install\_item:

startFilePickerActivity();

return true;

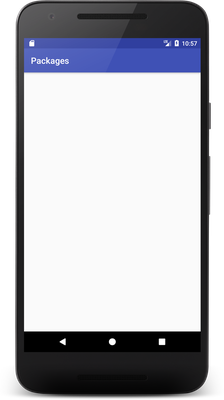
default:

return super.onOptionsItemSelected(item);

}

}

Запустите приложение, нажмите на overflow-меню, затем на пункт "Установить" и увидите, как Activityзапустилась! :)

[](https://skillberg.com/media/uploads/2017/11/17/2971e91a0ddd468eacb7b86379386a74.png)Запущенная Activity

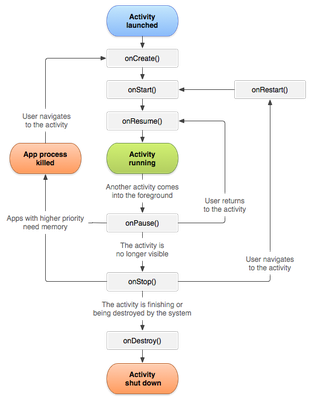
Пока что в ней пусто, но она уже отображается.

Жизненный цикл Activity

Пока что мы почти не сталкивались с жизненным циклом Activity, но обязательно столкнемся позднее, поэтому нужно разобрать эту тему.

Вообще, знание жизненного цикла различных компонентов Android крайне важно. Без этого вы не сможете создать качественное приложение.

Посмотрите на эту схему:

[](https://skillberg.com/media/uploads/2017/11/17/54c84e157ac8407a8a553ed6bcf5a45c.png)Жизненный цикл Activity в Android

В ней представлены переходы между состояниями в зависимости от различных событий. Основных состояний Activity — три:

* **Stopped** — существует, но не отображается. В таком состоянии Activity находится, к примеру, когда она только создается, когда пользователь сворачивает приложение, или когда другая Activity отображается поверх неё.
* **Paused** — существует, отображается, не в фокусе. В этом состоянии Activity находится, например, в процессе создания (когда она уже видна), или когда система собирается показать другую Activity поверх неё.
* **Resumed** — существует, отображается, в фокусе. В этом состоянии происходит взаимодействие пользователя с Activity.

Когда происходит переход из одного состояния в другое, система оповещает нас, вызывая соответствующий метод. Вот эти методы, перечисленные в порядке вызова:

* onCreate() — вызывается при создании Activity. В этом методе нужно, к примеру, привязать лэйаут вызовом setContentView(). Мы уже работали с этим методом. После него всегда вызывается onStart().
* onRestart() — вызывается перед перезапуском Activity после того, как она была остановлена. После этого метода всегда вызывается onStart().
* onStart() — Вызывается перед тем, как Activity станет видна для пользователя. После этого метода вызывается onResume(), если Activity переходит на передний план, либо onStop(), если она становится скрытой.
* onResume() — вызывается перед тем, как Activity станет доступна для взаимодействия. После этого метода всегда вызывается onPause().
* onPause() — вызывается перед тем, как Activity уйдёт на задний план, и система покажет другую Activity. В этом методе следует сохранять данные, но сделать это нужно очень быстро. После этого метода вызывается либо onResume(), если Activity возвращается на передний план, либо onStop(), если Activity становится скрытой. После вызова этого метода Activity может быть уничтожена системой, и вызова последующих методов не произойдёт.
* onStop() — вызывается в случае, когда Activity перестаёт быть видимой для пользователя. После этого метода может вызваться либо onRestart(), если Activity будет возобновлена, либо onDestroy(), если она будет уничтожена. После вызова этого метода Activity может быть уничтожена системой, и вызова последующих методов не произойдёт.
* onDestroy() — вызывается перед тем, как Activity будет уничтожена. После этого метода Activityперестаёт существовать.

Давайте создадим все эти методы в MainActivity, добавим в каждый из них логирование и посмотрим, в каком порядке они вызываются:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Log.i(TAG, "onCreate");

// ....

}

@Override

protected void onRestart() {

super.onRestart();

Log.i(TAG, "onRestart");

}

@Override

protected void onStart() {

super.onStart();

Log.i(TAG, "onStart");

}

@Override

protected void onResume() {

super.onResume();

Log.i(TAG, "onResume");

}

@Override

protected void onPause() {

super.onPause();

Log.i(TAG, "onPause");

}

@Override

protected void onStop() {

super.onStop();

Log.i(TAG, "onStop");

}

@Override

protected void onDestroy() {

super.onDestroy();

Log.i(TAG, "onDestroy");

}

Запустите приложение и посмотрите в логи:

09-10 11:46:35.421 com.test.packages I/MainActivity: onCreate

09-10 11:46:35.716 com.test.packages I/MainActivity: onStart

09-10 11:46:35.717 com.test.packages I/MainActivity: onResume

Теперь запустите вторую Activity, как мы это делали ранее. В логах появятся следующие строчки:

09-10 11:47:27.885 com.test.packages I/MainActivity: onPause

09-10 11:47:28.536 com.test.packages I/MainActivity: onStop

Нажмите кнопку "назад" один раз, чтобы вернуться к главной Activity. Логи:

09-10 11:48:06.363 com.test.packages I/MainActivity: onRestart

09-10 11:48:06.364 com.test.packages I/MainActivity: onStart

09-10 11:48:06.364 com.test.packages I/MainActivity: onResume

В отличие от запуска Activity, при восстановлении дополнительно вызвался метод onRestart().

Теперь нажмите "назад" ещё раз, чтобы выйти из приложения.

09-10 11:49:25.101 com.test.packages I/MainActivity: onPause

09-10 11:49:25.726 com.test.packages I/MainActivity: onStop

09-10 11:49:25.726 com.test.packages I/MainActivity: onDestroy

В отличие от приостановки Activity при запуске другой Activity, вызвался метод onDestroy().

Кстати, если вы просто свернёте приложение, нажав на кнопку **Home**, то onDestroy(), скорее всего, не будет вызван (если система посчитает, что имеет достаточно свободных ресурсов).

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/PackagesAndroid/tree/lesson-15)