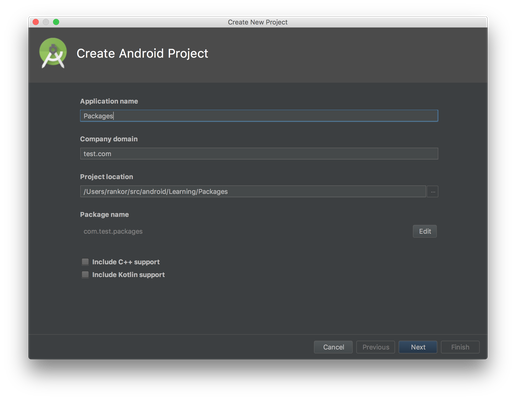
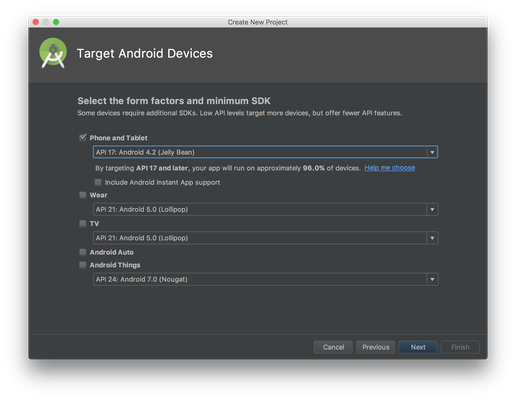
В этом уроке мы с вами наконец-то начнем создавать настоящее полнофункциональное приложение, которое можно будет даже выгрузить в Google Play!

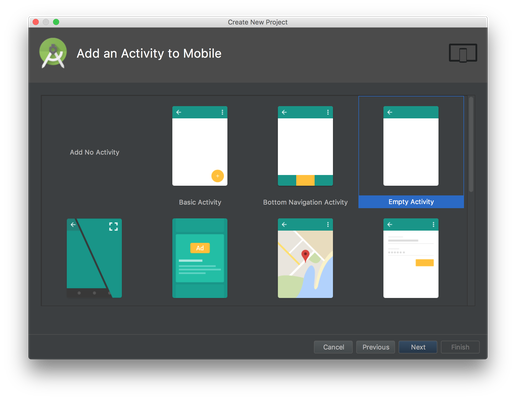
Наше приложение будет отображать список установленных приложений, информацию о них, и давать возможность удалить их, в том числе с помощью Root.

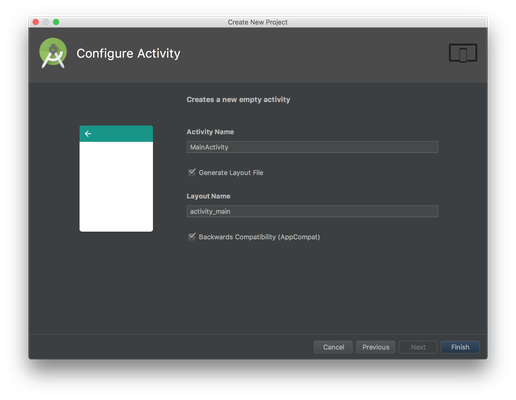
Создание проекта

В первую очередь, создайте новый проект. Вы уже знаете, как это сделать, но на всякий случай я прикреплю пошаговые скриншоты мастера создания проекта:

[](https://skillberg.com/media/uploads/2017/11/17/bb42f71a29104124af616fca6dc1e0b3.png)Выбор имени пакета и директории

[](https://skillberg.com/media/uploads/2017/11/17/286d71db115a4427acef607a7114266b.png)Выбор платформ и версий

[](https://skillberg.com/media/uploads/2017/11/17/212cb1b2f358410caf89382c70e6d87a.png)Выбор Activity

[](https://skillberg.com/media/uploads/2017/11/17/cf9554f74cb541519530875874713f86.png)Настройка Activity

Теперь измените activity\_main.xml, чтобы он выглядел вот так:

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

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

xmlns:tools="http://schemas.android.com/tools"

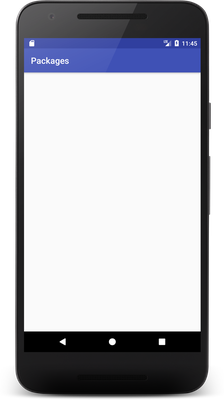
android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.test.packages.MainActivity">

</FrameLayout>

Запустив приложение, вы увидите пустой экран:

[](https://skillberg.com/media/uploads/2017/11/17/3775be233376497084500e3de9dc50c8.png)Пустой проект

Так и должно быть.

Context

Context — очень важное понятие в Android. Контекст — это объект, который описывает текущее состояние среды, в которой выполняется код. Многие классы, с которыми вы будете работать, наследуют контекст, например:

* Activity
* Application
* Service

Контекст потребуется вам для доступа ко многим вещам: к ресурсам, к другим компонентам приложения, к интерфейсу и т.д. Помните, мы использовали в Activity методы getResources() или getString()?

На самом деле, доступ к этим методам мы можем получить только из контекста. Если у нас нет инстанса Context, то мы не сможем получить доступ к тем же локализованным строкам.

Если вы сейчас все же до конца не поняли, что такое контекст — не переживайте, понимание этого придёт со временем.

Пакет

Понятие пакета (package) тоже очень важно. Каждое приложение в Android — это не только код, но и связанные с ним компоненты: ресурсы, манифест и т.д.

При сборке все это пакуется в файл APK (по сути — ZIP-архив) и подписывается ключом разработчика. Итого, package = код + ресурсы + мета-информация.

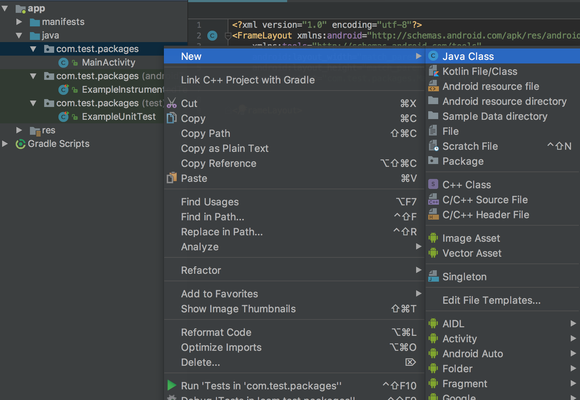
В Android есть менеджер пакетов, отвечающий за установку, обновление, удаление приложений, и хранящий информацию о них.

Программно получить к нему доступ можно через класс **PackageManager**. Инстанс же PackageManager, в свою очередь, можно получить, вызвав метод Context.getPackageManager().

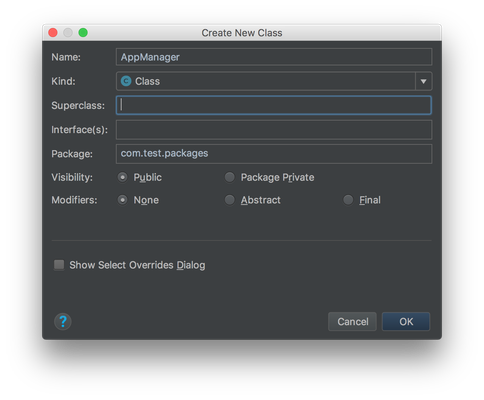
Получение списка пакетов

Давайте же начнем создание нашего первого приложения! Мы уже выяснили, что нам нужно для начала: PackageManager и, соответственно, Context.

Будет не очень правильно писать весь код в Activity, поэтому создадим отдельный класс, который будет получать список установленных приложений. Нажмите правой кнопкой на наш основной пакет, после чего выберите **New** -> **Java Class**.

[](https://skillberg.com/media/uploads/2017/11/17/ca86d3c307a046aeb91bf1c6d64b19ff.png)Создание нового класса

Введите имя класса AppManager и нажмите на кнопку **ОК**.

[](https://skillberg.com/media/uploads/2017/11/17/5b50042addad4fa6a3267e05de22fa17.png)Имя нового класса

Создастся новый класс:

package com.test.packages;

/\*\*

\* Класс, получающий список пакетов

\*/

public class AppManager {

}

Нам понадобится контекст, чтобы получить PackageManager. Создайте конструктор, который получает в качестве аргумента Context:

public AppManager(Context context) {

}

Теперь создайте поле типа PackageManager и проинициализируйте его в конструкторе:

package com.test.packages;

import android.content.Context;

import android.content.pm.PackageManager;

/\*\*

\* Класс, получающий список пакетов

\*/

public class AppManager {

private final PackageManager packageManager;

public AppManager(Context context) {

packageManager = context.getPackageManager();

}

}

Кроме этого, нам не помешает вспомогательный класс, в котором мы будем хранить информацию о приложении.

Создайте еще один класс AppInfo:

package com.test.packages;

import android.graphics.drawable.Drawable;

/\*\*

\* Информация о приложении

\*/

public class AppInfo {

private final String packageName;

private final int versionCode;

private final String versionName;

private final String name;

private final Drawable icon;

public AppInfo(String packageName, int versionCode, String versionName, String name, Drawable icon) {

this.packageName = packageName;

this.versionCode = versionCode;

this.versionName = versionName;

this.name = name;

this.icon = icon;

}

public String getPackageName() {

return packageName;

}

public int getVersionCode() {

return versionCode;

}

public String getVersionName() {

return versionName;

}

public String getName() {

return name;

}

public Drawable getIcon() {

return icon;

}

}

Теперь создадим метод в классе AppManager, который будет отдавать нам список установленных приложений.

В первую очередь, этот список нужно получить.

Это делается вызовом метода PackageManager.getInstalledPackages():

public List<AppInfo> getInstalledApps() {

List<PackageInfo> installedPackages = packageManager.getInstalledPackages(0);

}

Список пакетов мы получили. Теперь просто проходимся по нему циклом, преобразуем PackageInfo в AppInfoи добавляем в список. А после этого возвращаем получившийся список.

Теперь метод будет выглядеть вот так:

public List<AppInfo> getInstalledApps() {

List<AppInfo> installedApps = new ArrayList<>();

List<PackageInfo> installedPackages = packageManager.getInstalledPackages(0);

for (PackageInfo installedPackage : installedPackages) {

AppInfo appInfo = new AppInfo(

installedPackage.packageName, // Имя пакета

installedPackage.versionCode, // Код версии

installedPackage.versionName, // Имя версии

installedPackage.applicationInfo.loadLabel(packageManager).toString(), // Имя приложения

installedPackage.applicationInfo.loadIcon(packageManager) // Иконка приложения

);

installedApps.add(appInfo);

}

return installedApps;

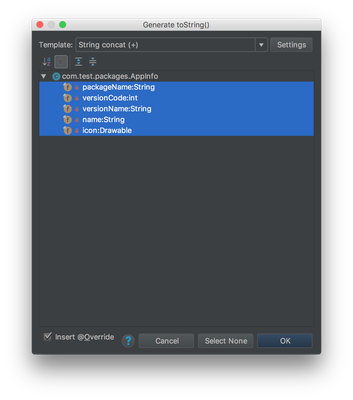
}

Список приложений мы получать научились, а вот отображать пока нет. Как же нам по-быстрому проверить, все ли корректно работает?

Во-первых, добавьте метод toString() в класс AppInfo. Сделать это можно очень просто: откройте класс, нажмите Ctrl+Enter и выберите **toString**:

[](https://skillberg.com/media/uploads/2017/11/17/904941e5501c4f86b66daf6458214325.png)Создание метода toString()

В появившемся окне убедитесь, что все поля выделены, и нажмите на кнопку **ОК**:

[](https://skillberg.com/media/uploads/2017/11/17/af262fc7b89447c896c6a63583db3576.png)Выбор полей для метода toString()

Готово, у вас сгенерируется примерно такой метод:

@Override

public String toString() {

return "AppInfo{" +

"packageName='" + packageName + '\'' +

", versionCode=" + versionCode +

", versionName='" + versionName + '\'' +

", name='" + name + '\'' +

", icon=" + icon +

'}';

}

А теперь поговорим о логировании.

Логирование

Логирование (журналирование) информации — важная часть разработки. Наверное, нет ни одного разработчика, написавшего что-то серьезнее, чем "Hello, World" и не использовавшего логи в том или ином виде.

Разработка под Android, конечно же, не будет исключением.

Существует отдельный класс Log, содержащий следующие статические методы:

* d() — отладочная информация (debug).
* w() — предупреждение (warning).
* v() — какие-либо подробности (verbose).
* i() — информационные сообщения (info).
* e() — ошибка (error).
* wtf() — что-то экстраординарное. Официально в документации расшифровано как "What a Terrible Failure!", но мы-то все понимаем… ;)

Все эти методы имеют две общие сигнатуры:

* String tag — тэг (метка для понимания, из какого места в приложении пришел этот лог).
* String message — собственно, сообщение.

И вторая:

* String tag — аналогично первому варианту.
* String message — аналогично первому варианту.
* Throwable throwable — Exception, который нужно залогировать.

Вторая сигнатура особо не понадобится, поэтому можете не заморачиваться на тему того, зачем же нам третий параметр.

Зачем же нужно так много методов, если они одинаковые? Дело в том, что они выводят информацию с разным приоритетом. Это нужно для двух вещей:

1. Нам проще будет фильтровать логи (а их бывает очень много! в реальных приложениях легко может идти по 10-20 строчек логов в секунду!).
2. В релизной сборке можно будет отключить подробные логи, оставив только критичные.

Давайте попробуем что-то залогировать. Откройте MainActivity, и измените метод onCreate() так, чтобы он выглядел следующим образом:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Log.d("MainActivity", "Debug!");

Log.w("MainActivity", "Warning!");

Log.v("MainActivity", "Verbose!");

Log.i("MainActivity", "Info!");

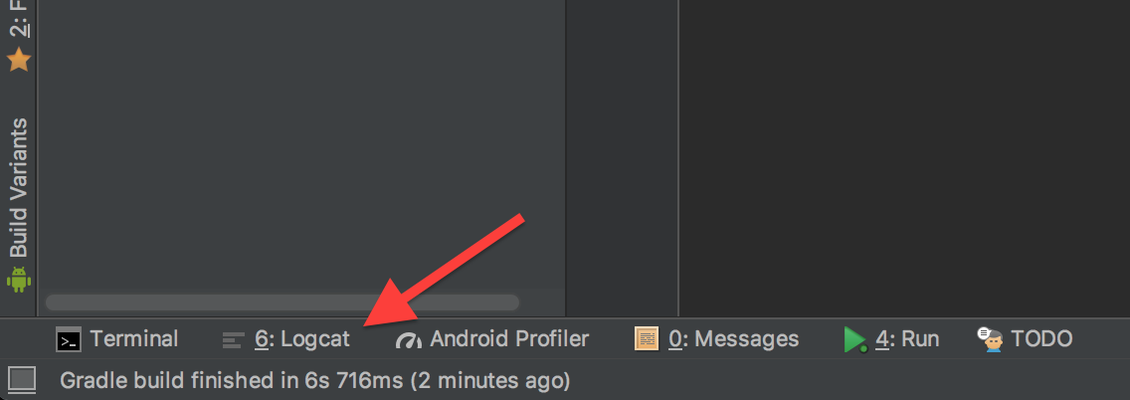
Log.e("MainActivity", "ERROR!");

Log.wtf("MainActivity", "WTF???!");

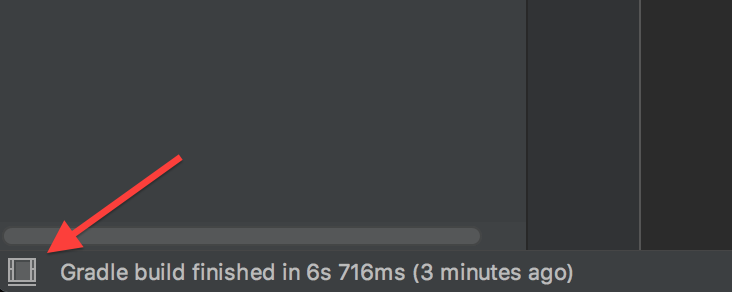
}

Запустите приложение. Вроде ничего не изменилось, да? Логов все еще не видно :(

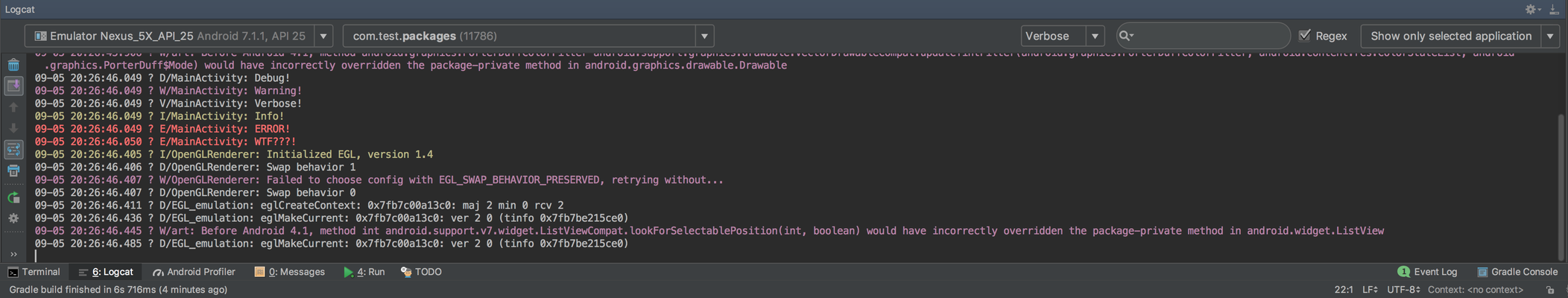
На самом деле, для просмотра логов есть отдельный "компонент" Android SDK, встроенный, в том числе, в Android Studio. В самом низу Android Studio вы можете увидеть кнопочку **Logcat**:

[](https://skillberg.com/media/uploads/2017/11/17/37222d3359d34b5bb7172a270a93cfc7.png)Кнопка Logcat

Если вдруг там ничего такого нет, то нажмите вот сюда, и оно появится:

[](https://skillberg.com/media/uploads/2017/11/17/c0352adb421b439e9c81124c0e34bd66.png)Кнопка показа панелей

Теперь нажмите на кнопку **Logcat** и увидите вот такое окошко:

[](https://skillberg.com/media/uploads/2017/11/17/b365aaccb9ab4b02a5037ee7d22cc3e8.png)Окно Logcat

И вот как раз в нем (ближе к верхней части) те самые логи, которые мы выводили ранее.

Также хорошей практикой считается вынос тэга (первого аргумента логов) в константу. Давайте сделаем это. В итоге наш класс будет таким:

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MainActivity";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Log.d(TAG, "Debug!");

Log.w(TAG, "Warning!");

Log.v(TAG, "Verbose!");

Log.i(TAG, "Info!");

Log.e(TAG, "ERROR!");

Log.wtf(TAG, "WTF???!");

}

}

А теперь давайте выведем список пакетов в логи. Удалите строки, в которых мы используем логирование. После этого давайте создадим наш AppManager:

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MainActivity";

private AppManager appManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

appManager = new AppManager(this);

}

}

Как видите, мы передали в AppManager this в качестве контекста — this в данном случае это наша Activity, а Activity, как вы помните, является контекстом.

Теперь получим список приложений, пройдемся по нему циклом и выведем информацию о каждом приложении в логи:

public class MainActivity extends AppCompatActivity {

private static final String TAG = "MainActivity";

private AppManager appManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

appManager = new AppManager(this);

List<AppInfo> installedApps = appManager.getInstalledApps();

for (AppInfo installedApp : installedApps) {

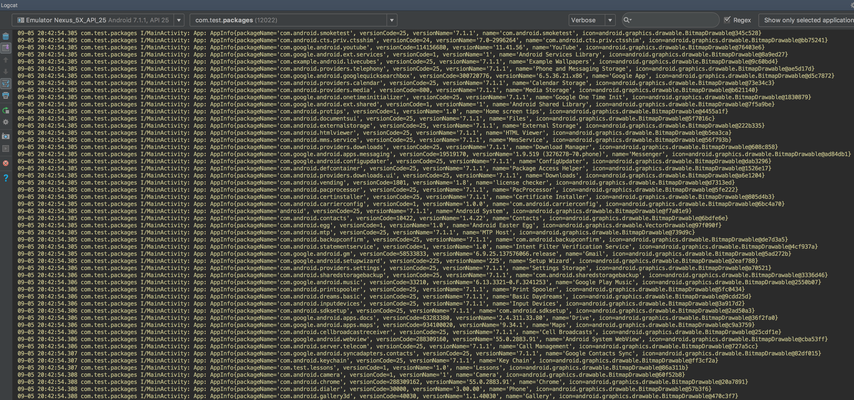
Log.i(TAG, "App: " + installedApp);

}

}

}

Запустив приложение, вы увидите в логах примерно следующее:

[](https://skillberg.com/media/uploads/2017/11/17/bc0aaa5ab29b4866ab015bd139daab77.png)Логи в панели Logcat

Супер, все работает! В следующих уроках мы будем доделывать приложение — ведь мало просто получить информацию об установленных приложениях, их нужно еще и отобразить :)

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/PackagesAndroid/tree/lesson-11)