Android system implements the **principle of last privilege**

that is, each app by default has access only to the components that is requires to do its work and no more

This creates a very secure environment in which an app cannot access parts of the system for which it is not given permission.

However, there are ways for an app to share data with other apps and for an app to access system services:

It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files.

To conserve system resources, apps with the same user ID can also arrange to run in the same Linux process and share the same VM.

The apps must also be signed with the same certificate.

An app can request permission to access device data such as the device's location, camera,

and Bluetooth connection. The user has to explicitly grant these permissions.

There are four different types of app components:

Activities

Services

Broadcast receivers

Content providers

=================

Activities

=================

An activity is the entry point for interacting with the user.

It represents a single screen with a user interface

Although the activities work together to form a cohesive user experience

in the email app, each one is independent of the others.

As such, a different app can start any one of these activities if the email app allows it.

For example, a camera app can start the activity in the email app that composes new mail to allow

the user to share a picture.

An activity facilitates the following key interactions between system and app:

\* Keeping track of what the user currently cares about (what is on screen) to ensure that the

system keeps running the process that is hosting the activity.

\* Knowing that previously used processes contain things the user may return to (stopped activities),

and thus more highly prioritize keeping those processes around.

\* Helping the app handle having its process killed so the user can return to activities with their

previous state restored.

\* Providing a way for apps to implement user flows between each other, and for the system to

coordinate these flows. (The most classic example here being share.)

You implement an activity as a subclass of the Activity class

The Activity class is designed to facilitate this paradigm. When one app invokes another,

the calling app invokes an activity in the other app, rather than the app as an atomic whole.

In this way, the activity serves as the entry point for an app's interaction with the user.

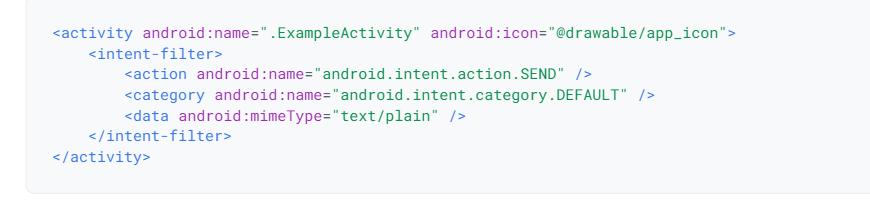
You implement an activity as a subclass of the Activity class.

To use activities in your app, you must register information about them in the app’s manifest,

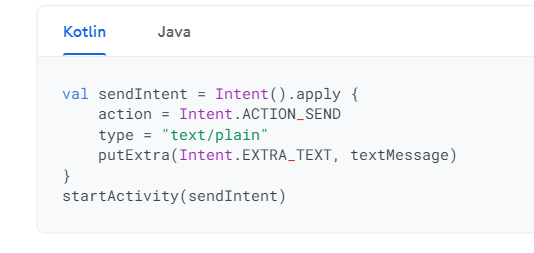
and you must manage activity lifecycles appropriately.

To declare your activity, open your manifest file and add an <activity>

element as a child of the <application> element.



In this example, the <action> element specifies that this activity sends data. Declaring the <category> element as DEFAULT enables the activity to receive launch requests. The <data> element specifies the type of data that this activity can send. The following code snippet shows how to call the activity described above:



Вызов Activity с помощью такого Intent – это **явный**вызов. Т.е. с помощью класса мы **явно**указываем какое Activity хотели бы увидеть. Это обычно используется**внутри одного приложения**.

Существует также **неявный**вызов Activity. Он отличается тем, что при создании **Intent**мы используем **не класс**, а **заполняем**параметры **action**, **data**, **category**определенными значениями. Комбинация этих значений определяют цель, которую мы хотим достичь. Например: отправка письма, открытие гиперссылки, редактирование текста, просмотр картинки, звонок по