FIAP

DESENVOLVIMENTO MOBILE - ANDROID

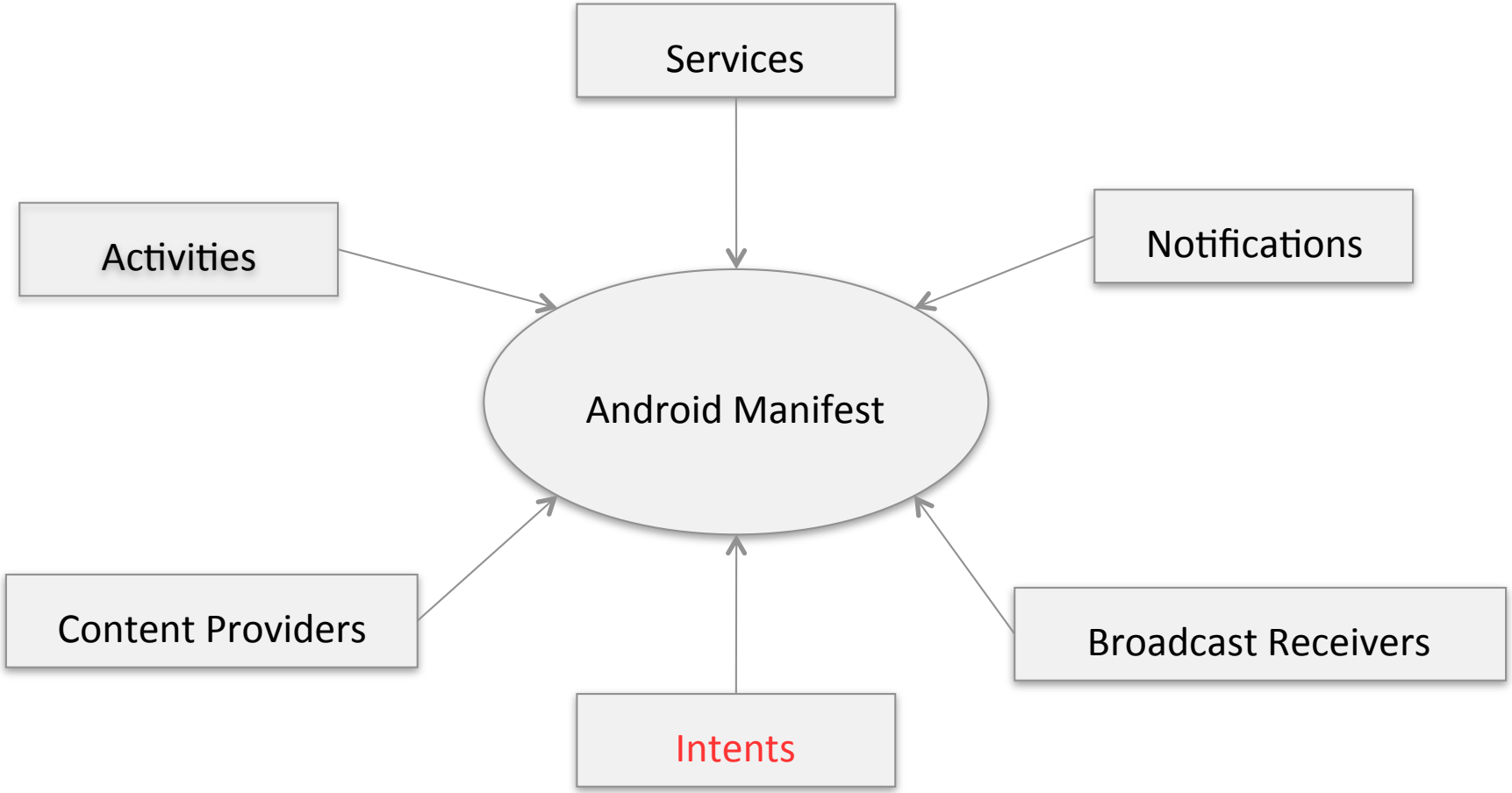
INTENTS

PROF. EDSON A. SENSATO
profedsonsensato@fiap.com.br

Versão 1– 2016

FIAP

PRINCIPAIS COMPONENTES DE UM APP



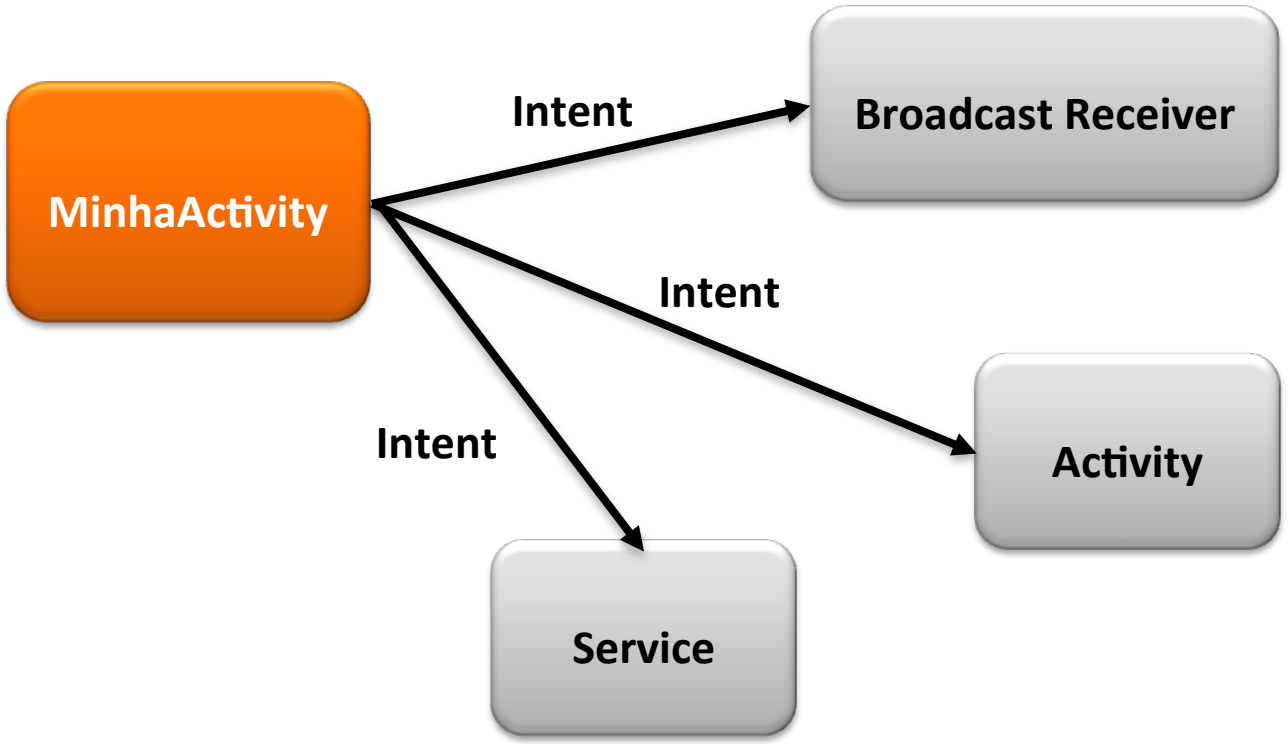
Andoid Manifest é um arquivo XML (*AndroidManifest.xml*) que define e integra os componentes de uma aplicação vistos acima.

CONCEITO DE INTENT

Intents são mensagens assíncronas trocadas entre componentes de uma mesma aplicação ou entre componentes de aplicações distintas;

Tais componentes podem ser **Activities**, **Services** ou **Broadcast Receivers**;

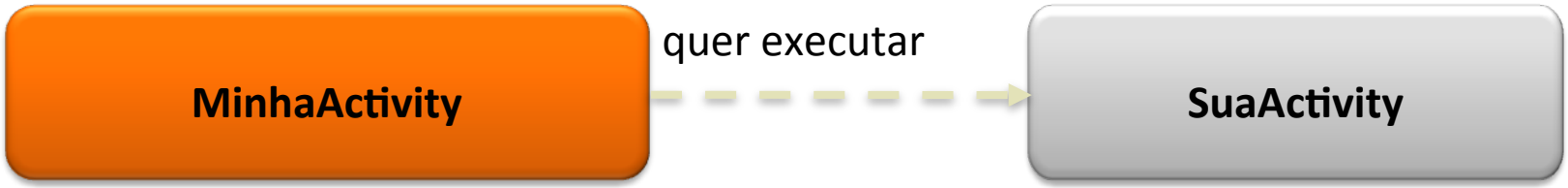
Um **Intent** pode conter informações que serão utilizadas na comunicação entre dois componentes;



FIAP

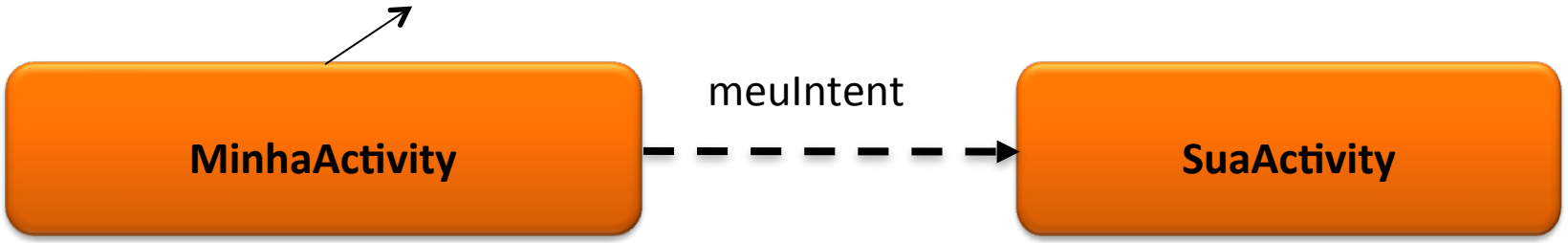
INTENT EXPLÍCITO

Suponha que a **MinhaActivity** queira executar a **SuaActivity** (dentro da mesma aplicação):



Então, a **MinhaActivity** deverá instanciar um *Intent* e acionar a **SuaActivity** enviando-lhe o *Intent* criado:

```
// Cria o Intent
Intent  meuIntent = new Intent(this, SuaActivity.class)
// Inicia a Activity enviando o meuIntent
startActivity(meuIntent);
```



FIAP

PASSAGEM DE PARÂMETROS

Parâmetros podem ser associados ao *Intent* e encaminhados para a *Activity* destino;

Para tanto, basta utilizar o método **putExtra** da classe *Intent*:

putExtra(String P1, ? P2)

Onde:

P1 → nome do parâmetro
P2 → valor do parâmetro (**?** indica o tipo de dados... *String*, *int*, *boolean*...)

Os valores são encapsulados em um objeto do tipo **Bundle** que é recuperado na *Activity* destino conforme abaixo:

Bundle param = getIntent().getExtras();

Já os valores armazenados no **Bundle** são recuperados pelo método:

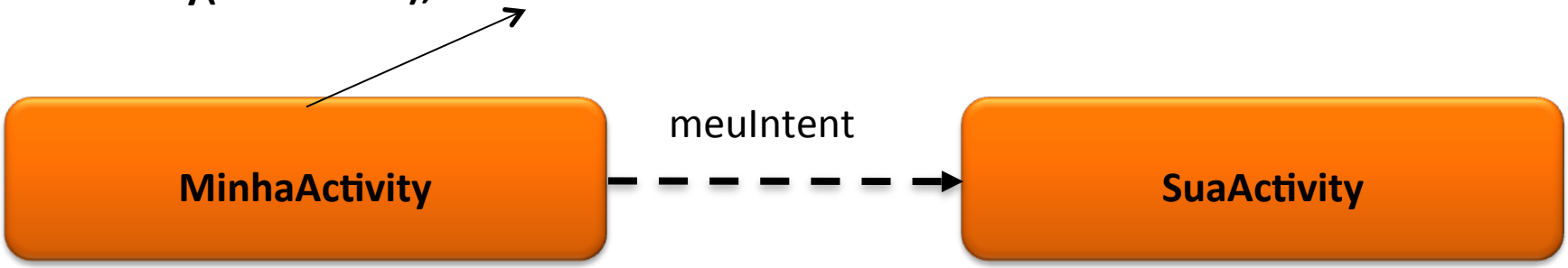
get?(String P1)

Onde

? → indica o tipo de dados... String, int, …
P1→ nome do parâmetro

EXEMPLO

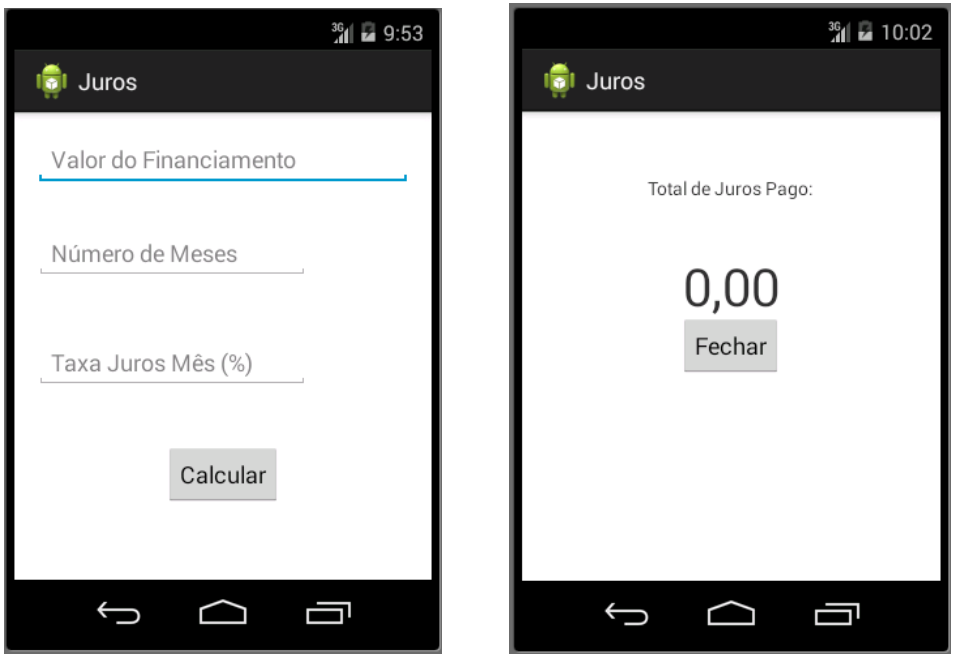
```
Intent  meuIntent = new Intent(this, SuaActivity.class)
// Associa valores ao Bundle
meuIntent.putExtra("mensagem", "Boa Noite!!!! ");
startActivity(meuIntent);
```



Bundle	
Nome	**Valor**
mensagem	Boa Noite

```
// Obtém o Bundle
Bundle param = getIntent().getExtras();
// Obtém os valores associados ao Bundle
String msg = param.getString ("mensagem")
```

EXERCÍCIO



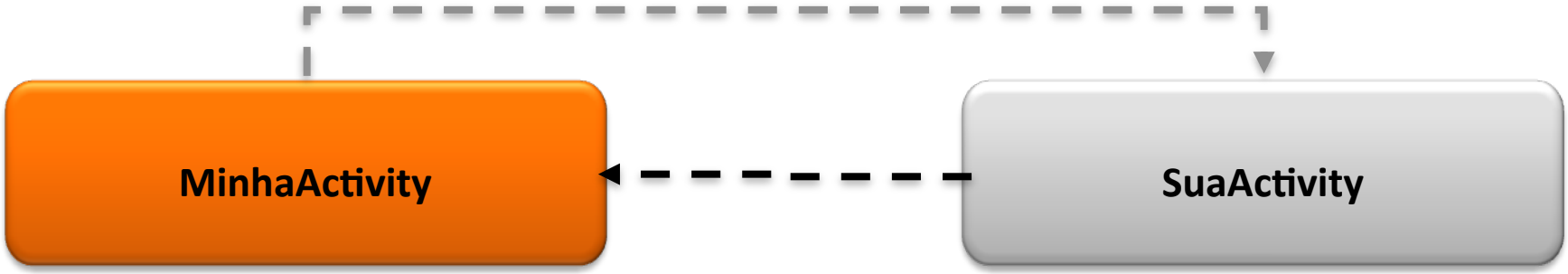
Implemente uma aplicação para o cálculo de juros simples.
Criar 2 activities: uma para receber os parâmetros do cálculo (valor, prazo e taxa de juros) e outra para exibir o valor dos juros pagos conforme a fórmula:

Juros Pagos = Valor * Taxa de Juros * Total Meses

Dica: para finaizar uma activity basta chamar o método finish()

RETORNO EXECUÇÃO

Suponha agora que a **SuaActivity** já executou e um resultado deve ser retornado para a **MinhaActivity**:



Para tanto, ao iniciar o *Intent* é preciso utilizar o método:

startActivityForResult(Intent P1, int P2)

Onde **P1** é o **Intent** e **P2** pode ser qualquer número para identificar a requisição;

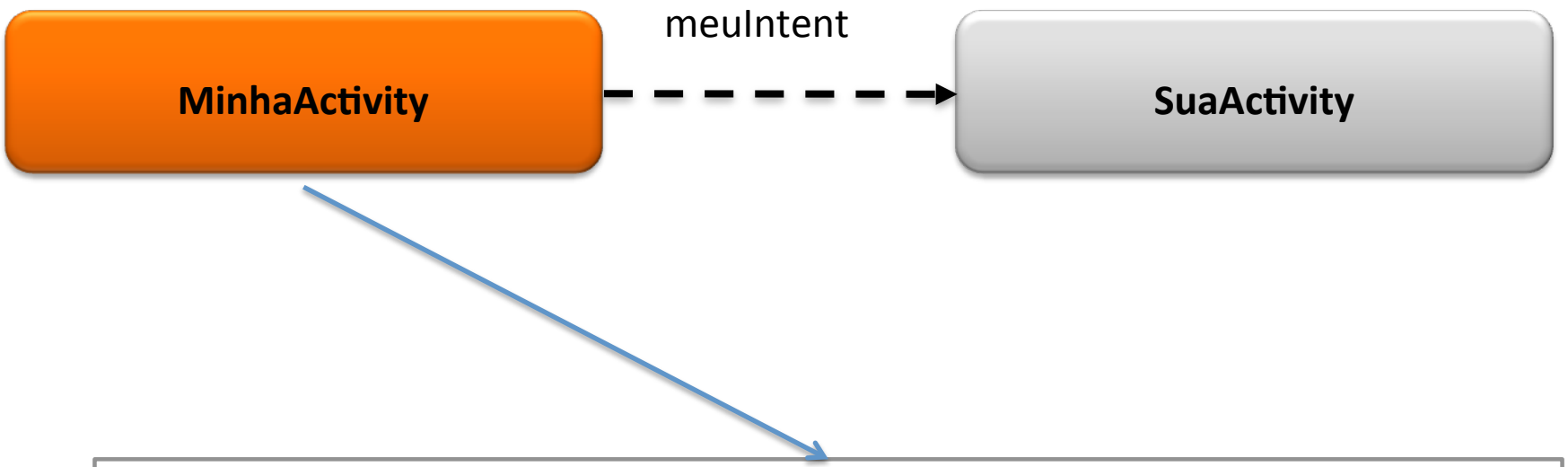
Quando o componente acionado terminar a execução o método abaixo (sobrescrevê-lo) é iniciada no componente que emitiu o *Intent*:

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {

}
```

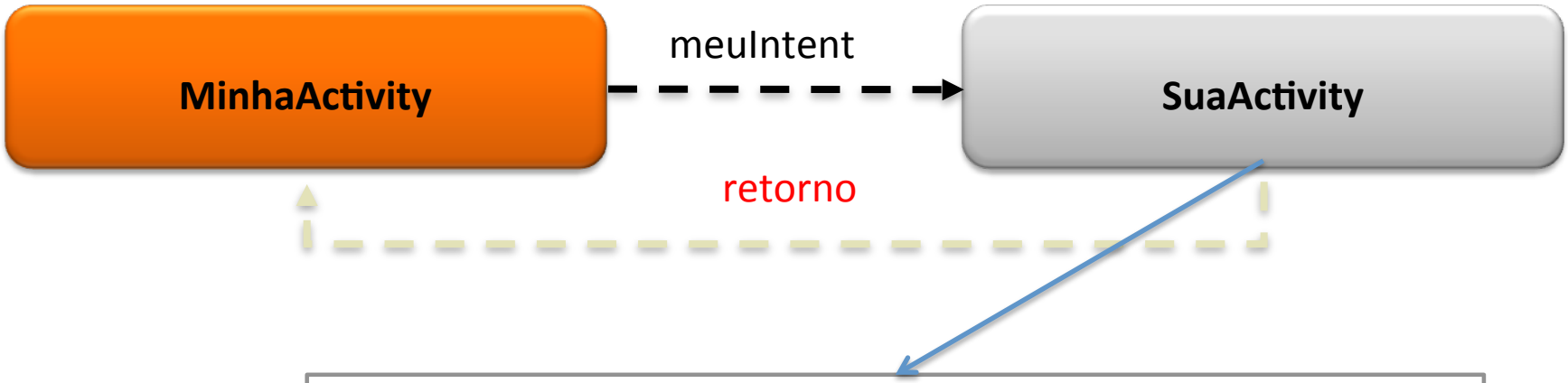
FIAP

PASSO 1 - CHAMADA



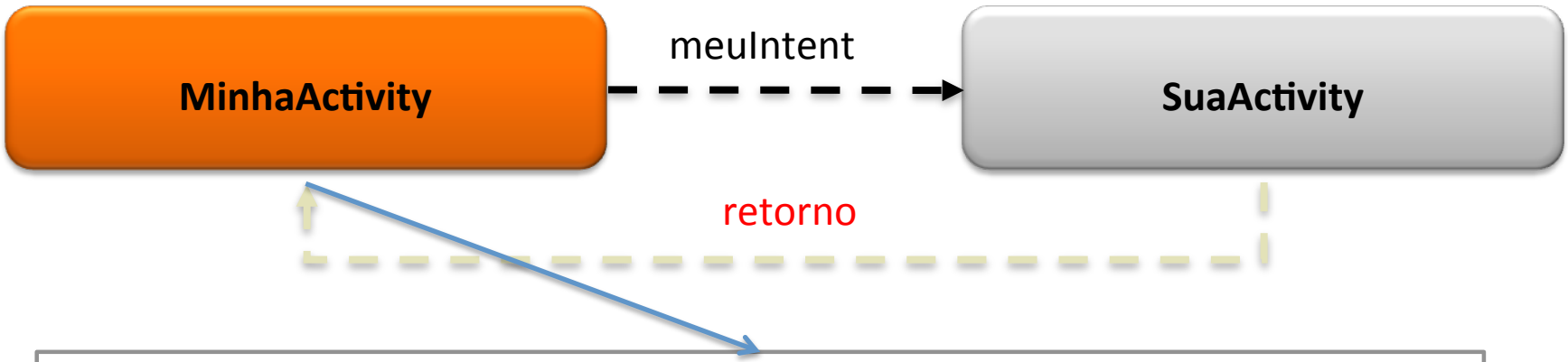

```
Intent  meuIntent = new Intent(this, SuaActivity.class)
// Inicia a Activity esperando um retorno
// Esta requisição será identificada pelo número 1 (pode ser qualquer número)
startActivityForResult(meuIntent, 1);
```

PASSO 2 - RETORNANDO




```
@Override
public void finish() {
        // Cria o Intent de retorno
        Intent ret = new Intent();
        ret.putExtra("retorno", "Obrigado!");
        // Indica que a execução foi OK
        setResult(RESULT_OK, ret);
        // Finaliza a SuaActivity retornando para MinhaActivity
        super.finish();
}
```

PASSO 3 – LENDO PARÂMETROS RETORNO




```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
// Verifica que a execução foi OK e qual foi o intent que efetuou a chamada
if (resultCode == RESULT_OK && requestCode == 1) {
        // Verifica se existe o parâmetro retorno
        if (data.hasExtra("retorno")) {
                // Obtém o valor do parâmetro retorno
                String ret = data.getExtras().getString("retorno");
                Toast.makeText(this, ret, Toast.LENGTH_SHORT).show();
} } }
```

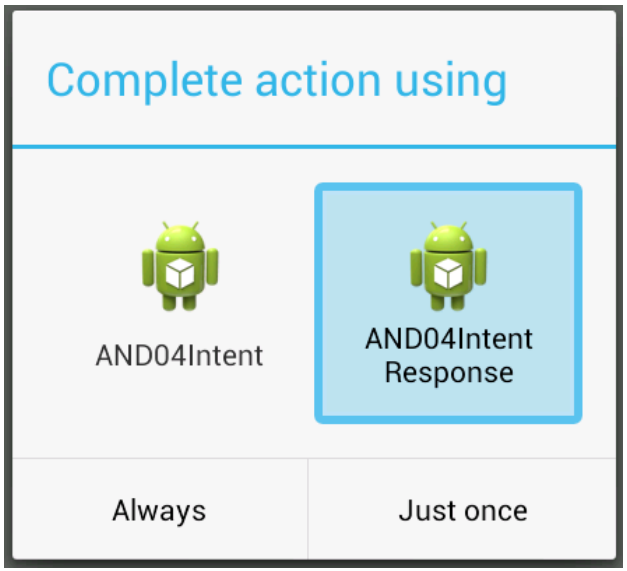
INTENTS IMPLÍCITOS

Vimos anteriormente a comunicação **Explícita** onde as classes das *Activities* eram informadas explicitamente na criação do *Intent*

Mas existe também a comunicação **Implícita** onde um determinado componente é solicitado por um nome e o sistema operacional é quem decide quem vai executá-lo

Tais nomes são registrados por meio do **Intent-Filter**

Caso dois ou mais componentes tenham o mesmo nome...



No exemplo ao lado, temos duas Activites (**AND04Intent** e **AND04IntentResponse**) registradas com o mesmo nome no Intent-Filter...

INTENT FILTER

Para criar um **intent-filter** e registrar um nome para a sua Activity, basta adicionar ao **AndroidManifest.xml** a tag **<intent-filter>** e **<action>**:

```
<activity android:name=".MinhaActivitiy" ...>
<intent-filter>
      <action android:name= "meu.pacote.MinhaActivity" />
      <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
```

Onde:

action → identificação da Activity;
category → categoria da Activity;

Agora, **MinhaActivity** pode ser acionada de **QUALQUER APLICAÇÃO** assim:

```
Intent i = new Intent("meu.pacote.MinhaActivity");
startActivity (i);
```

INTENT FILTER DATA

O intent-filter pode ser complementado definido-se o tipo de dado que a activity espera receber utilizando a tag data:

```
<activity android:name=".MinhaActivitiy" ...>
<intent-filter>
      <data android:scheme="http"/>
       <action android:name= "meu.pacote.MinhaActivity" />
       <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
```

Veja agora duas formas de acionar a MinhaActivity:

```
Intent i1 = new Intent("meu.pacote.MinhaActivity");
startActivity (i1);

Uri uri = Uri.parse("http://teste123.com.br");
Intent i2 = new Intent("meu.pacote.MinhaActivity", uri);
startActivity (i2);
```

ACTIONS PADRÃO

Existem algumas activities já instaladas nos dipositivos e que podem ser acionadas pelas nossas aplicações

Por exemplo: discagem telefônica, navegador web, agenda de contatos, etc...

Sendo assim, algumas actions podem ser visualizadas na classe Intent na forma de constantes

Tais constantes iniciam com **ACTION_**

Exemplo:

```
String uriFiap = Uri.parse("http://www.fiap.com.br"))

Intent i = new Intent(Intent.ACTION_VIEW, uriFiap);

startActivity(i);
```

FIAP

ACTIONS PADRÃO

Alguns tipos de serviços e suas URIs correspondentes são:

Serviço	Descrição	URI
Intent.ACTION_VIEW	Visualização conteúdo	http://www.fiap.com.br geo:0,0?z=19 content://contacts/people
Intent.ACTION_CALL	Chamada telefônica	tel://(+55)1133858010
Intent.ACTION_DIAL	Discagem número	tel://(+55)1133858010
Intent.ACTION_EDIT	Edição de conteúdo	content://contacts/people/1

EXEMPLO: ACESSO CÂMERA

Uma aplicação prática dos Intents é o acesso à câmera do dispositivo

Para tanto, basta seguir os passos abaixo

1. Habilitar o uso da câmera e definir permissões necessárias (AndroidManifest.xml):

```
<uses-feature android:name="android.hardware.camera" />

<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

2. Acionar a Activity responsável pela câmera interna passando como parâmetro o caminho do arquivo onde a imagem será armazenada

3. Ler o arquivo da imagem capturada no método onActivityResult

EXEMPLO: ACESSO CÂMERA

A câmera pode ser acessada por meio do Intent implícito **MediaStore.ACTION_IMAGE_CAPTURE**

```
public class CameraActivity extends Activity {

private String caminho;

public void capturarFoto(View v) {
Intent i = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
// diretório de armazenamento de imagens do dispositivo
File storageDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
File image = null;
try {
        // cria o arquivo da imagem
        image = File.createTempFile("foto", ".jpg", storageDir);
} catch (IOException e) {
        e.printStackTrace();
}
// a variável caminho deve ser definida globalmente na Activity pois será utilizada no onActivityResult
caminho = image.getAbsolutePath();
// define o arquivo de armazenagem por meio do parâmetro MediaStore.EXTRA_OUTPUT
i.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(image));
// inicia a câmera
startActivityForResult(i, 0);
}
```

EXEMPLO: ACESSO CÂMERA

Após capturar a imagem, o método onActivityResult é executado e então, pode-se acessar o arquivo com a imagem capturada

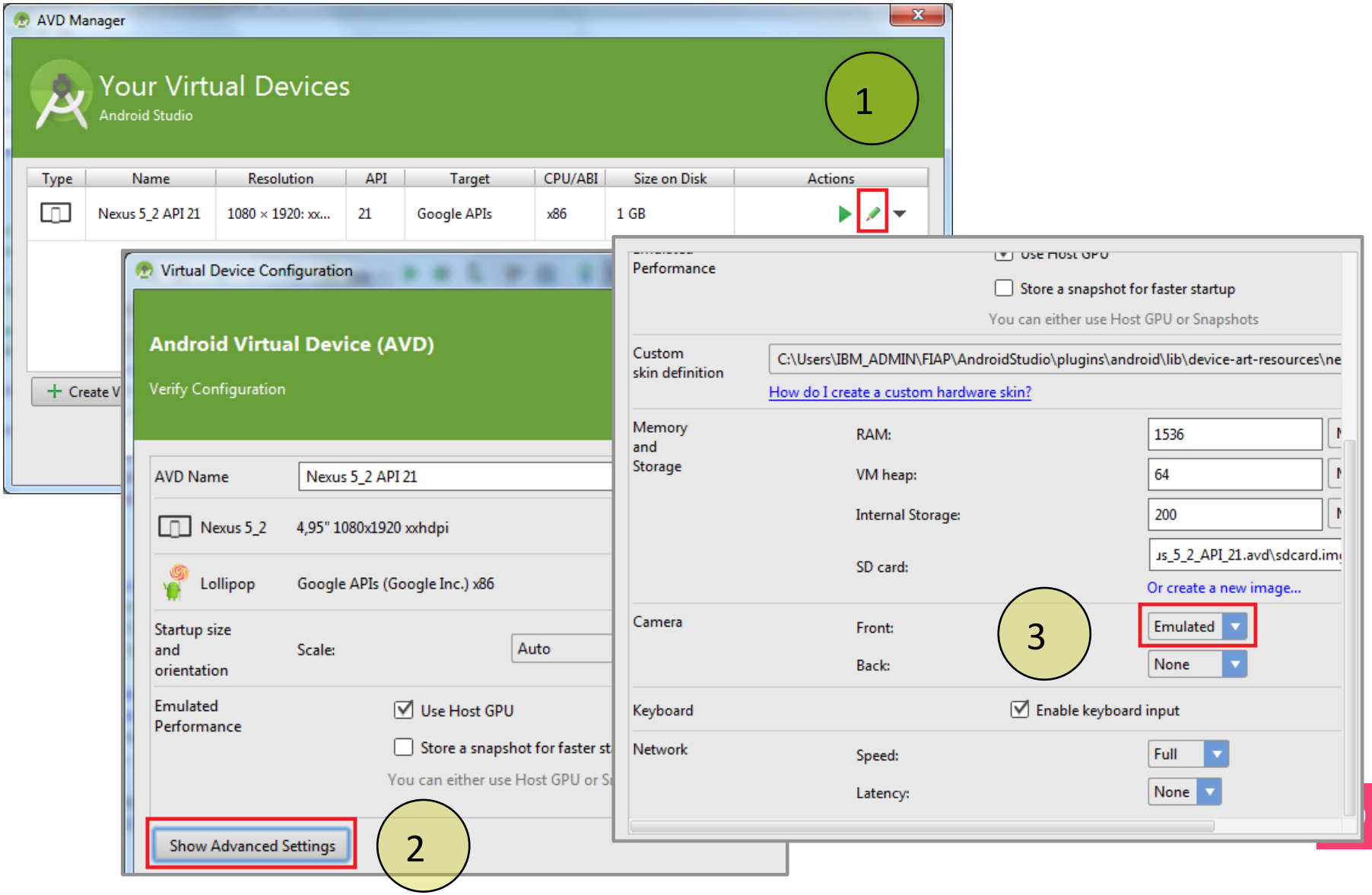
Lembrar que a varável caminho contém o caminho completo do arquivo e foi declarada globalmente na Activity

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

   // cria a imagem (Bitmap) a partir do arquivo da imagem
   BitmapFactory.Options bmOptions = new BitmapFactory.Options();
   Bitmap bitmap = BitmapFactory.decodeFile(caminho, bmOptions);
   img.setImageBitmap(bitmap);

}
```

HABILITAR CÂMERA AVD



FIAP