Tópicos Avançados em Programação

Aula 3

Prof. Marcelo Rodrigues do Nascimento

Conversa Inicial

Seja bem-vindo à terceira aula de Tópicos Avançados de Programação. Na aula anterior trabalhamos o primeiro dos quatro componentes de aplicação da plataforma Android, as Activities. Aprendemos sobre seu ciclo de vida, seus estados, envio de mensagens e criação de interfaces.

Nesta aula trabalharemos o conceito de comunicação entre Activities, Listeners, criação de objetos personalizados e sua serialização, além de expandirmos nosso conhecimento no desenvolvimento de interfaces utilizando novos componentes e estruturas de Layouts.

Ao finalizarmos esta etapa teremos desenvolvido um aplicativo que se baseará nos preceitos fundamentais da programação Android, tomará vantagem dela e agregará muito mais valor a nossos usuários.

No material online, o professor Marcelo apresenta os temas que serão abordados!

Contextualizando

Diferentemente do desenvolvimento para Desktop, em que a comunicação entre janelas ocorre dentro de uma mesma linha de execução, a comunicação entre diferentes Activities no ambiente Android se dá através da chamada direta ao sistema operacional.

Para cada Activity que desejamos iniciar, devemos enviar ao sistema operacional um objeto de Intenção que informe ao mesmo o que desejamos fazer. Esta particularidade torna especialmente complexo o envio e o recebimento de parâmetros entre módulos do sistema. Nesta aula trataremos sobre como o envio de mensagens é feito, além de criarmos objetos personalizados, serializando-os e enviando-os através de mensagens para novas Activities. Também ampliaremos nossa interface utilizando componentes como TextViews, Buttons e RadioButtons.

Tema 1 – Retornando resultados de uma Activity

Para iniciarmos uma nova Activity de dentro de uma Activity existente devemos passar sua referência a uma Intent (intenção) e chamar o método **startActivity()**. Este método se encarrega de criar a nova Activity, passada por referência dentro da Intent. No entanto a nova Activity criada não possui ligação com a Activity que a criou, uma vez que no Intent passamos como argumento apenas o Context e a Classe de Activity que servirá de base para sua construção.

Mas o que isto representa para o desenvolvedor Android?

Sua Activity neste momento não possui nenhuma forma de comunicação com quem a instanciou, ou seja, você não conseguirá, por exemplo, criar uma segunda Activity de seleção de valores e capturar o valor selecionado pelo usuário. E isto pode ser um tremendo problema. Para resolver esta situação utilizamos o método startActivityForResult(), que notificará à nova Activity que um resultado é aguardado, e também permitirá à Activity atual a inicialização de um Listener, ou seja, um método que “escuta” determinadas mensagens do sistema operacional e as processa de acordo com o que foi definido pelo desenvolvedor.

O primeiro passo para que possamos dar início a uma atividade com retorno é vincular um número inteiro à ação da qual aguardamos um retorno. Sempre que utilizarmos um valor qualquer, que não será modificado em nenhum momento durante a execução do aplicativo, o definimos através de uma variável estática, ou seja, imutável.

Desta maneira aumentamos a legibilidade de nosso código, afinal é muito mais fácil entender o propósito de:

- startActivityForResult(minhaintent , ACAO_BUSCA_PREFERENCIA_USER);

Do que:

- startActivityForResult(minhaintent , 1903);

Após vincularmos a ação, devemos também redefinir o comportamento do método **onActivityResult()** de nossa Activity, para que o mesmo capture as mensagens recebidas e as processe de acordo.

O método onActivityResult() inclui três argumentos:

- O código de ação passado para startActivityForResult (em nosso exemplo, ACAO_BUSCA_PREFERENCIA_USER).
- Um código de resultado especificado pela segunda Activity. Se a operação foi bem-sucedida, este código será RESULT_OK. Caso contrário, o resultado será RESULT_CANCELED.
- Uma Intent que transporta os dados do resultado.

  Lembre-se que as Intents não transportam apenas Activities, mas sim objetos.

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

}
```

Nossa Activity está agora pronta para receber dados da segunda Activity. No entanto vale ressaltar que é necessário que saibamos o formato do resultado que está vindo no Intent. Quando a Activity que retornará o resultado foi desenvolvida por você, o processo fica mais simples. No entanto, se você iniciar uma Activity de terceiros, aguardando um resultado, é importante verificar a documentação da mesma para identificar que tipo de objeto será retornado no resultado.

Por exemplo, se o aplicativo de Câmera for iniciado, o objeto que será retornado (caso a ação possua o resultCode RESULT_OK) será um Bitmap.

Além de preparar nossa Activity para receber o resultado da segunda Activity, devemos também dar a essa segunda Activity a possibilidade de retornar um resultado. Para isto devemos criar uma Intent que armazenará o objeto que desejamos retornar à nossa Activity principal e preenchê-la com o objeto desejado.

Após a criação da Intent e seu preenchimento, devemos executar o método **setResult()**, que receberá nosso código de retorno (no caso, RESULT_OK) e nossa Intent. Somente após estes passos é que devemos chamar o método que encerra nossa Activity, **finish()**;

Vamos ao nosso exemplo: iremos criar um novo projeto de aplicativo, composto de duas Activities. Em nossa Activity principal, nossa interface será composta de uma caixa de texto para título, uma caixa de texto (que informará ao usuário o retorno da Activity) e de um botão que terá como finalidade iniciar a segunda Activity.

Figura 1 – Activity 1

Na segunda Activity teremos um componente de seleção com três valores disponíveis para seleção. Ao selecionarmos qualquer valor neste componente, esta Activity será encerrada, passando para o método **setResult()** o valor escolhido pelo usuário.

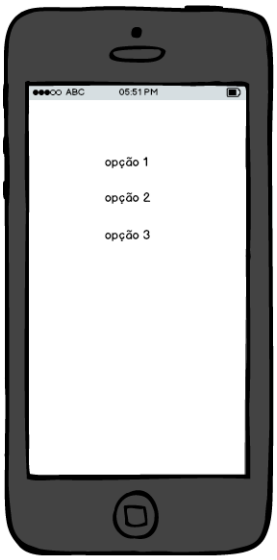

Figura 2 – Activity 2

Crie um novo projeto no Android Studio.

- Em Application name digite: ActivityResult
- Em Company Domain digite: aula3.grupouninter.com.br
- Selecione a localização do projeto e então clique em **Next**.

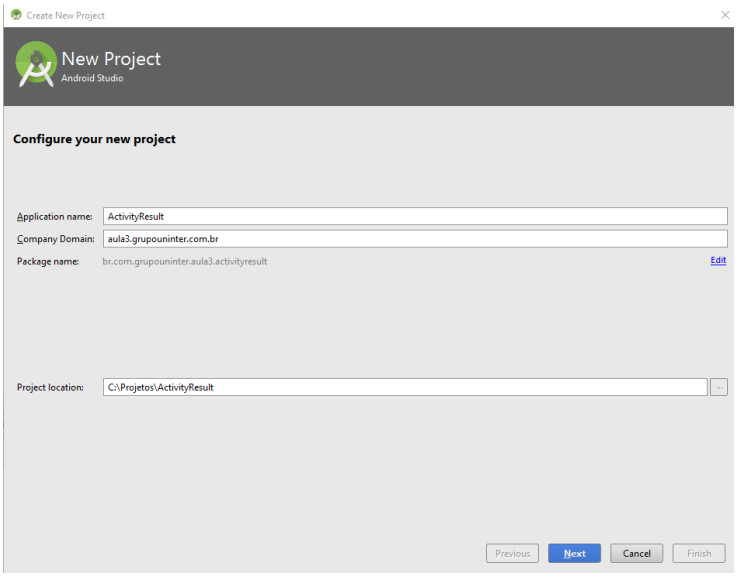

Figura 3 – Projeto ActivityResult

Em seleção de dispositivo alvo, deixe marcado a opção Phone and Tablet, com o SDK mínimo para API 19: Android 4.4 (KitKat).

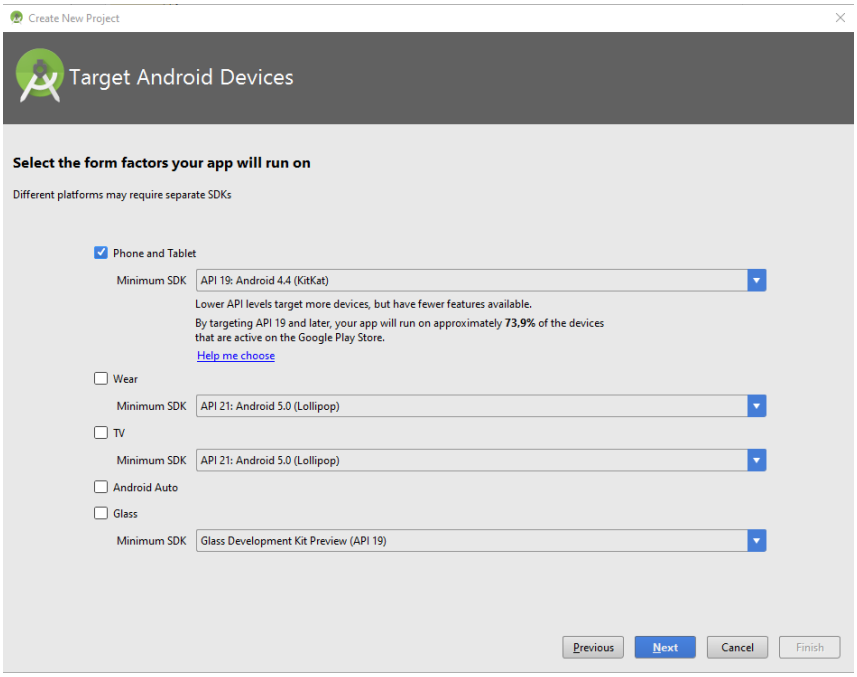

Figura 4 – Seleção de Dispositivos

Novamente começaremos com uma Activity Vazia, então selecione esta opção em Adicionar uma Activity para o Mobile e pressione **Next**.

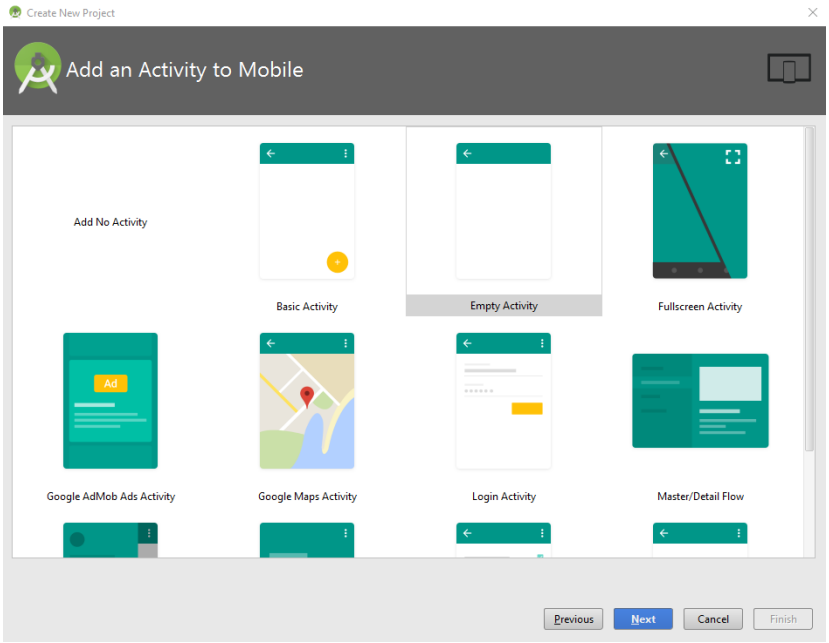

Figura 5 – Activity Vazia

Na tela de customização da Activity que acabamos de criar, em Activity Name, digite: **MainActivity**

- Em nome de layout digite: **activity_main**
- Pressione **Finish**

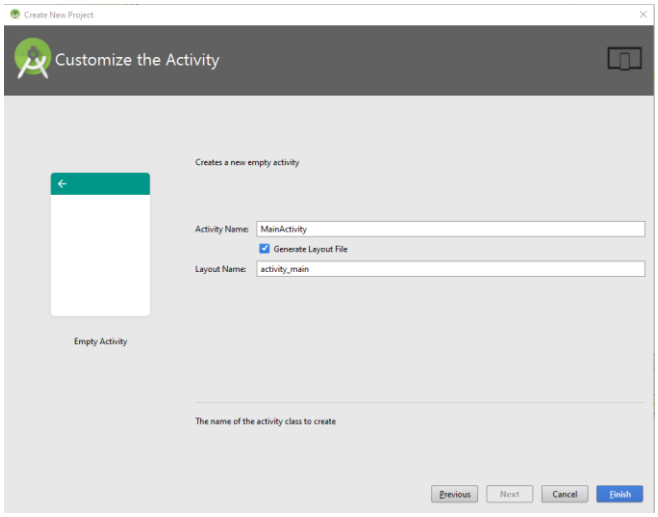

Figura 6 – Customizando a Activity

Assim, nosso projeto será criado. De acordo com nosso mockup (rascunho de interface), devemos adicionar ao layout duas caixas de texto e um botão.

Lembre-se que uma Activity é dividida em duas partes: a classe que a controla e o arquivo XML que representa seu layout. As alterações de layout devem, portanto, ser feitas no arquivo XML que se encontra na pasta **res->layout**. Em seu painel de projeto no Android Studio, expanda estas pastas até que o arquivo activity_main.xml esteja visível. Clique duas vezes sobre ele para que possamos editá-lo. O editor do Android Studio se comporta de acordo com o tipo de arquivo aberto. Como estamos editando um arquivo XML, o editor WYSIWYG é apresentado. Clique no componente TextView com o valor “Hello World” então pressione seu botão Delete, para removê-lo de nossa interface.

Nosso layout trabalha no modelo **RelativeLayout**, onde os componentes são alinhados em relação a um componente anterior. Conversaremos mais sobre os diversos tipos de Layout disponíveis nas próximas aulas.

Agora devemos recriar o layout que desenvolvemos em nosso mockup, portanto localize na paleta de componentes o componente **Medium Text** e arraste-o para seu layout. Tente posicioná-lo de acordo com a posição que definimos em nosso rascunho. Agora devemos mudar suas propriedades, para que este represente o texto desejado, em nosso caso "**A opção selecionada foi:**". Para isso, clique em seu componente na tela, e na janela Properties localize a propriedade **text**. Nela iremos digitar o texto que desejamos que seja apresentado por este componente.

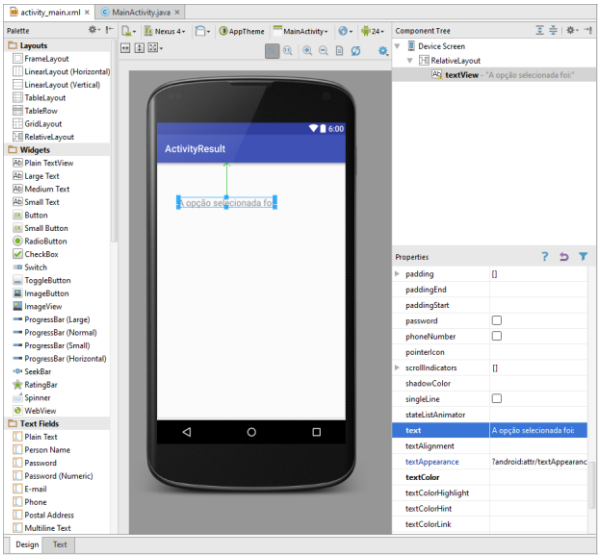

Figura 7 – Customizando o componente TextView

Agora vamos a um detalhe técnico. De acordo com a documentação do Android para recursos de aplicativos, considera-se uma boa prática colocar String, arrays de String, imagens, cores, tamanhos de fontes, dimensões e outros recursos de aplicativos em arquivos XML.

Este processo é conhecido como exteriorizar os recursos. Para ajudar na internacionalização (ou seja, tradução) de seu aplicativo, por exemplo, podemos criar arquivos em pastas separadas para cada idioma, onde todos utilizam a mesma referência a determinada String, porém com valores diferentes.

Na seção 2.8, na página 67 do livro “Uma Abordagem Baseada em Aplicativos”, de Deitel, Deitel e Wald, podemos encontrar um excelente exemplo a respeito da internacionalização de seu aplicativo, utilizando a técnica sugerida.

Para este projeto, iremos trabalhar com nossos valores “hard coded”. Necessitaremos também criar um identificador mais amigável a nossos componentes, para que possamos acessá-los sem chance de ambiguidade ou erro. Localize a propriedade ID e altere-a para “título”.

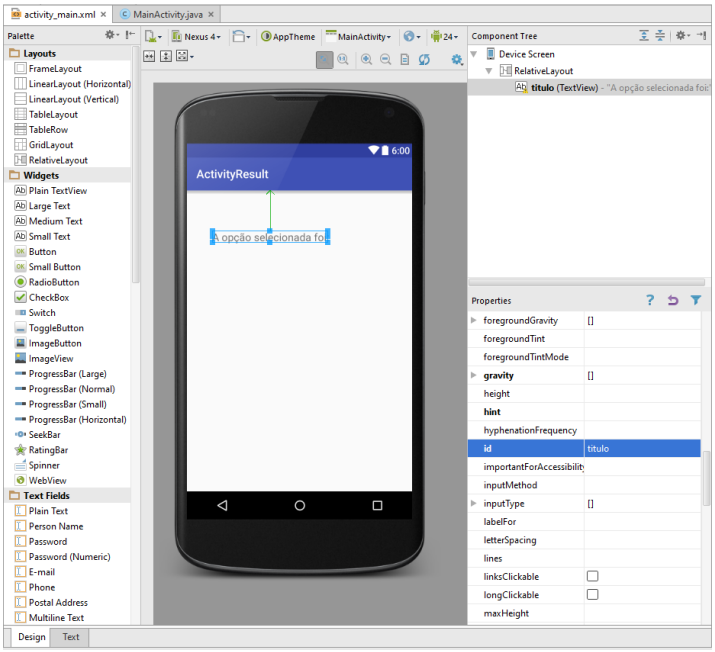

Figura 8 – modificando o ID do componente

Agora devemos colocar a segunda caixa de Texto, que receberá o valor de retorno da segunda Activity. Selecione o componente **Large Text** e arraste-o para sua interface.

Altere suas propriedades para:

- id: resultado
- text: OPÇÃO

Finalmente, selecione o componente **Button** e arraste-o para sua interface.

Altere suas propriedades para:

- id: btnOpcao
- text: Opções

Sua interface agora foi finalizada.

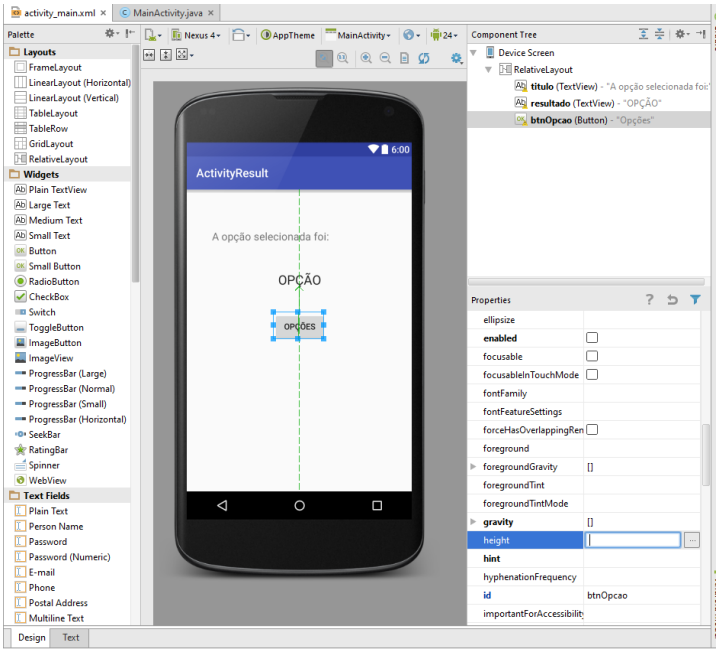

Figura 9 – Interface Finalizada

Para sua referência, no editor de interface, clicando-se na parte inferior na aba **Text**, temos acesso ao arquivo texto gerado por nossas ações. Caso você tenha encontrado alguma dificuldade, aqui está o arquivo XML gerado por nosso exemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="br.com.grupouninter.aula3.activityresult.MainActivity">

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textAppearance="?android:attr/textAppearanceMedium"
       android:text="A opção selecionada foi:"
       android:id="@+id/titulo"
       android:layout_marginTop="53dp"
       android:layout_alignParentTop="true"
       android:layout_alignParentStart="true"
       android:layout_marginStart="27dp" />

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textAppearance="?android:attr/textAppearanceLarge"
       android:text="OPÇÃO"
       android:id="@+id/resultado"
       android:layout_below="@+id/titulo"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="50dp" />


     <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Opções"
        android:id="@+id/btnOpcao"
        android:layout_below="@+id/resultado"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="42dp" />
  </RelativeLayout>
```

Antes de partirmos para a programação de nossa MainActivity, vamos criar a segunda Activity e também customizá-la. Em nosso Aplicativo **ActivityLifeCycle** aprendemos a criar cada arquivo separadamente e então como uni-los via programação. Desta vez vamos utilizar o Android Studio para criar estes dois arquivos mais facilmente, tal qual fizemos durante o início da configuração do projeto.

Em sua janela de Projetos, expanda a pasta **java->br.com.grupouninter.aula3.activityresult**.

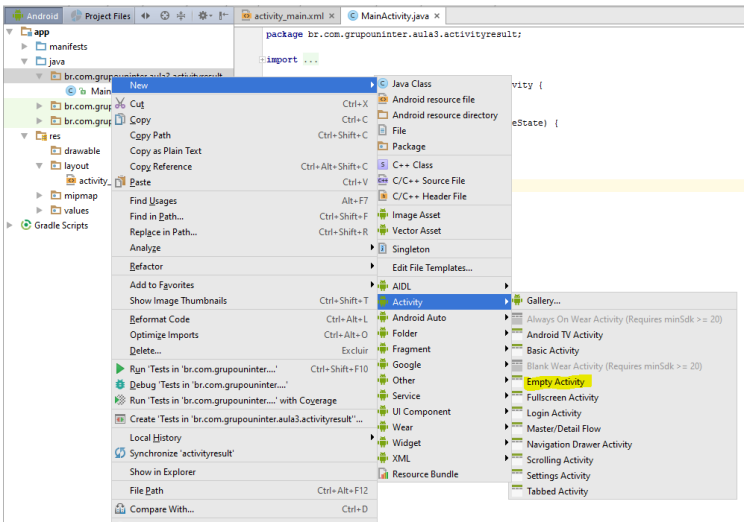

Figura 10 – Nova Activity

Clique com o botão direito sobre a pasta br.com.grupouninter.aula3.activityresult e então nas opções **New**, **Activity**, **Empty Activity**.

Na tela de configuração da Activity, preencha os seguintes valores:

- Activity Name: SegundaActivity
- Layout Name: activity_segunda
- Não selecione a opção “Launcer Activity”
- Package Name: br.com.grupouninter.aula3.activityresult

Clique em Finish. Seus arquivos SegundaActivity.java e activity_segunda.xml foram criados em suas pastas correspondentes. Também no arquivo Segunda Activity foi já redefinido o método onCreate, para que este agora utilize seu arquivo xml activity_segunda como conteúdo a ser mostrado com para esta Activity.

Finalmente, a SegundaActivity foi também corretamente declarada no AndroidManifest.xml, para que o sistema operacional possa localizá-la. Agora customizaremos nosso layout da segunda Activity, para que esta represente o Mockup demonstrado em nossa Figura 2.

Arraste três componentes **RadioButton** para sua Activity, e customize-os para que reflitam os seguintes valores:

- Primeiro RadioButton

id: rbOpcao1
text: Opção 1

- Segundo RadioButton

id: rbOpcao2
text: Opção 2

- Terceiro RadioButton

id: rbOpcao3
text: Opção 3

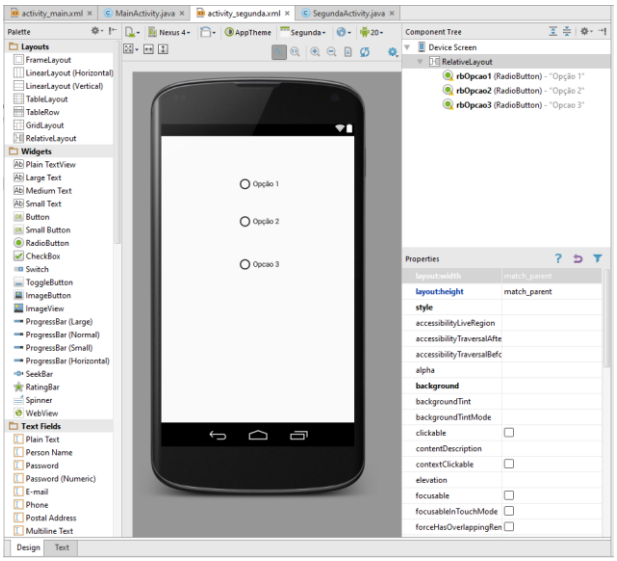

Figura 11 – Segunda Activity

Para sua referência, segue o xml de nossa segunda Activity:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"

tools:context="br.com.grupouninter.aula3.activityresult.SegundaActivity"
>
```

```
<RadioButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Opção 1"
   android:id="@+id/rbOpcao1"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="63dp" />

<RadioButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Opção 2"
   android:id="@+id/rbOpcao2"
   android:layout_below="@+id/rbOpcao1"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="42dp" />


  <RadioButton
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Opcao 3"
     android:id="@+id/rbOpcao3"
     android:layout_below="@+id/rbOpcao2"
     android:layout_centerHorizontal="true"
     android:layout_marginTop="53dp" />

</RelativeLayout>
```

Com nossos layouts finalizados, agora devemos desenvolver o código que orientará o comportamento de nossas Activities. Abra o arquivo MainActivity.java para que possamos dar início a nosso desenvolvimento. Antes que possamos utilizar um objeto devemos criar a referência a eles em nossa classe. De acordo com nosso Mockup, apenas um objeto sofrerá modificações de acordo com a decisão de nosso usuário, nosso TextView **resultado**. Também teremos apenas um componente que permitirá interações com o usuário, o Button **btnOpcao**.

Então, antes de mais nada, devemos iniciar estes componentes. Vamos aproveitar também para criar a variável estática de referência ao tipo de ação que desejamos passar à nossa Segunda Activity e monitorar.

```
...
public class MainActivity extends AppCompatActivity {

    TextView resultado;
    Button btnOpcao;
    static int ACAO_BUSCA_PREFERENCIA_USER = 1;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
...
```

Após criarmos estes objetos, devemos instanciá-los para que possam ser utilizados corretamente.

Em nosso método onCreate, criaremos uma instância do componente btnOpcao, e redefiniremos seu listener OnClickListener:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // ligamos o componente btnOpcao à representação no arquivo xml
    btnOpcao = (Button) findViewById(R.id.btnOpcao);
    // redefinimos o listener OnClickListener
    btnOpcao.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

        }
    });
}
```

Agora iremos criar a chamada a Segunda Activity, aguardando um resultado. Como sempre, devemos criar antes uma Intent, que armazenará o contexto e também o modelo de classe a ser instanciada pelo sistema operacional:

```
...
btnOpcao.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(view.getContext(), SegundaActivity.class);
        startActivityForResult(intent, ACAO_BUSCA_PREFERENCIA_USER);
    }
});
...
```

Neste momento nossa MainActivity está apta a iniciar uma nova Activity (Segunda Activity), solicitando a esta que retorne um valor para a ação ACAO_BUSCA_PREFERENCIA_USER, quando esta for finalizada. Tudo que precisamos fazer agora na MainActivity é escutar os resultados de retorno de ações e agir de acordo.

Para isso, devemos redefinir o método onActivityResult, para que este reflita nossas intenções:

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

}
```

No argumento requestCode receberemos o código de ação que está sendo retornado (ACAO_BUSCA_PREFERENCIA_USER, por exemplo). Em resultCode receberemos o resultado da ação (RESULT_OK ou RESUL_CANCELED) e finalmente em data receberemos uma Intent que conterá o resultado a ser transferido via resultado.

Para nosso exemplo, vamos verificar se: O requestCode foi ACAO_BUSCA_PREFERENCIA_USER, se o resultCode foi RESULT_OK e finalmente modificaremos nossa textView com o valor digitado.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){
   // verificando o tipo de request code
   if (requestCode == ACAO_BUSCA_PREFERENCIA_USER){
      // verificando se o usuário encerrou a activity retornando sua opcao
      if (resultCode == RESULT_OK){
         // ligamos o componente resultado à representação no arquivo xml
         resultado = (TextView) findViewById(R.id.resultado);
         // passamos o valor da Intent (uma String) para resultado.setText()
         resultado.setText(data.getStringExtra("retorno"));

      }
   }
}
```

Para sua referência, aqui está o código completo de nossa MainActivity.java

```
package br.com.grupouninter.aula3.activityresult;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

   TextView resultado;
   Button btnOpcao;
   static int ACAO_BUSCA_PREFERENCIA_USER = 1;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // ligamos o componente btnOpcao à representação no arquivo xml
    btnOpcao = (Button) findViewById(R.id.btnOpcao);
    // redefinimos o listener OnClickListener
    btnOpcao.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(view.getContext(), SegundaActivity.class);
            startActivityForResult(intent,
ACAO_BUSCA_PREFERENCIA_USER);
        }
    });
}


    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
 data){
        // verificando o tipo de request code
        if (requestCode == ACAO_BUSCA_PREFERENCIA_USER){
            // verificando se o usuário encerrou a activity retornando sua opcao
            if (resultCode == RESULT_OK){
                // ligamos o componente resultado à representação no arquivo xml
                resultado = (TextView) findViewById(R.id.resultado);
                // passamos o valor da Intent (uma String) para resultado.setText()
                resultado.setText(data.getStringExtra("retorno"));
            }
        }
    }
}
```

Agora devemos trabalhar em nossa SegundaActivity, para que esta retorne o valor selecionado pelo usuário. Abra o arquivo da Segunda Activity e vamos dar início a esta parte do desenvolvimento.

Analisando nosso Mockup para a segunda Activity percebemos que os três RadioButtons permitirão interação com o usuário. Então vamos instanciá-los:

```
...
public class SegundaActivity extends AppCompatActivity {

    RadioButton rbOpcao1;
    RadioButton rbOpcao2;
    RadioButton rbOpcao3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
...
```

Agora devemos iniciá-los e redefinir seu comportamento do onClickListener.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_segunda);

    rbOpcao1 = (RadioButton) findViewById(R.id.rbOpcao1);
    rbOpcao2 = (RadioButton) findViewById(R.id.rbOpcao2);
    rbOpcao3 = (RadioButton) findViewById(R.id.rbOpcao3);

    rbOpcao1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

        }
    });
```

```
        rbOpcao2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });

        rbOpcao3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });
    }
```

Finalmente, cada vez que o usuário pressionar um dos radio buttons, devemos setar o valor de retorno desta Activity e finalmente encerrá-la. Para isto, devemos antes de mais nada definir a Intent que carregará o valor selecionado pelo usuário e passar a mesma ao método setResult, juntamente com o resultCode correspondente.

Para retornar o valor via String, utilizaremos o método Intent.putExtra(), que nos permite retornar valores identificados:

```
rbOpcao1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.putExtra("retorno", "Primeira Opcao");
        setResult(RESULT_OK, intent);
        finish();
    }
});
```

```
rbOpcao2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.putExtra("retorno", "Primeira Opçao");
        setResult(RESULT_OK, intent);
        finish();
    }
});

rbOpcao3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.putExtra("retorno", "Primeira Opçao");
        setResult(RESULT_OK, intent);
        finish();
    }
});
```

Para sua referência, aqui está o código completo de nossa SegundaActivity.java

```
package br.com.grupouninter.aula3.activityresult;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.RadioButton;

public class SegundaActivity extends AppCompatActivity {

    RadioButton rbOpcao1;
    RadioButton rbOpcao2;
    RadioButton rbOpcao3;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_segunda);

        rbOpcao1 = (RadioButton) findViewById(R.id.rbOpcao1);
        rbOpcao2 = (RadioButton) findViewById(R.id.rbOpcao2);
        rbOpcao3 = (RadioButton) findViewById(R.id.rbOpcao3);

        rbOpcao1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                intent.putExtra("retorno", "Primeira Opção");
                setResult(RESULT_OK, intent);
                finish();
            }
        });



        rbOpcao2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                intent.putExtra("retorno", "Segunda Opção");
                setResult(RESULT_OK, intent);
                finish();
            }
        });

        rbOpcao3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                intent.putExtra("retorno", "Terceira Opção");
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }

}
```

Confuso? O professor Marcelo fala mais sobre o retorno de resultados de uma Activity e testa a sua aplicação no material online.

Tema 2 – Enviando Parâmetros para uma Activity

Receber o resultado de uma Activity foi um processo interessante, mas nos deixa com uma dúvida: **como podemos enviar parâmetros para uma nova Activity?**

Como você aprendeu em nosso exemplo anterior, utilizamos o Objeto Intent para armazenar o valor que desejávamos retornar através do método **setResult()**. Para o envio de parâmetros à nova Activity, o processo é basicamente o mesmo. Devemos criar uma Intent e passar a ela os objetos que desejamos que sejam transmitidos à nova Activity.

Vamos adicionar novas funcionalidades ao nosso projeto ActivityResult, adicionando a ele uma nova Activity que tenha a capacidade de receber um argumento passado dentro de sua Intent de criação.

Clique com o botão direito na pasta que representa o pacote de nossa aplicação (br.com.grupouninter.aula3.activityresult) e então selecione New -> Activity - > Empty Activity. Na tela de configuração da Activity preencha os seguintes valores:

- Activity Name: TerceiraActivity
- Layout Name: activity_terceira
- Package name: br.com.grupouninter.aula3.activityresult
- Deixe desmarcada a opção Launcher Activity
- Clique no botão Finish.

Para este exemplo, vamos criar na Terceira Activity uma TextView que mostrará o valor da String que enviamos à Intent durante sua criação. Portanto, selecione o arquivo activity_terceira.xml, e no edito de Layout adicione um componente Large Text com as seguintes propriedades:

- id: valorParametro
- text: Valor Parametro

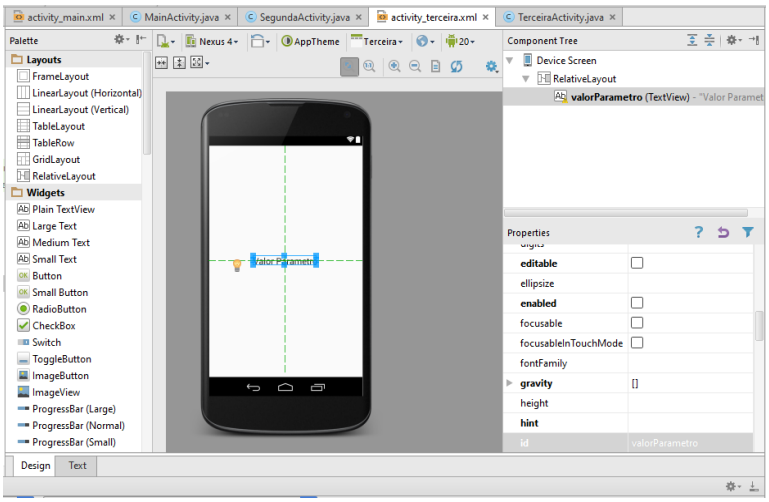

Figura 12 – Terceira Activity

Aqui está o código XML da Terceira Activity, para sua referência:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"

tools:context="br.com.grupouninter.aula3.activityresult.TerceiraActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Valor Parametro"
      android:id="@+id/valorParametro"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Agora, em nosso arquivo TerceiraActivity.java devemos descrever o comportamento desejado para nossa nova Activity. Abra o arquivo TerceiraActivity.java.

A proposta para esta Activity é que, no momento de sua criação, o valor do texto para o componente **TextView** valorParametro seja alterado para o valor passado junto a Intent que o criou. Portanto, devemos criar a referência a este componente:

```
...
public class TerceiraActivity extends AppCompatActivity {

    TextView valorParametro;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
...
```

Na criação de nossa Activity, instancie nosso objeto, vinculando-o a seu correspondente no layout:

- valorParametro = (TextView) findViewById(R.id.***valorParametro***);

Finalmente, durante a criação do componente devemos buscar o parâmetro desejado de dentro da Intent e utilizar seu valor em substituição à propriedade Text de nosso componente.

Nosso primeira tarefa é buscar a Intent que originou nossa Activity, utilizando o método **getIntent()**;

- Intent intent = getIntent();

Mas, de onde veio esta Intent?

Lembre-se de que Activities são criadas sempre a partir de uma Intent (mesmo nossa MainActivity). A Intent que a originou é armazenada na classe Activity, e pode ser recuperada através do método getIntent().

Agora que temos acesso à Intent, basta buscarmos o valor desejado utilizando-se o método **getStringExtra()**; Este método buscará dentro da Intent por um objeto identificado pelo argumento passado ao mesmo e retornará o valor no formato String. O objeto Intent oferece métodos para retorno de valores no formato Int, Boolean, Char, Long entre outros.

Em nosso exemplo, vamos supor que o Intent que deu origem à nossa Activity veio com um extra do tipo String chamado “retorno”.

- valorParametro.setText(intent.getStringExtra(**"retorno"**));

Nossa TerceiraActivity está pronta para receber o parâmetro, então devemos agora criar em nossa MainActivity o método que criará esta nova Intent, passando a ela o argumento “retorno”. Para sua referência, aqui está o código completo do arquivo TerceiraActivity.java:

```
package br.com.grupouninter.aula3.activityresult;

import android.content.Intent;
import android.support.v4.widget.TextViewCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class TerceiraActivity extends AppCompatActivity {

    TextView valorParametro;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_terceira);

        valorParametro = (TextView) findViewById(R.id.valorParametro);

        Intent intent = getIntent();
        valorParametro.setText(intent.getStringExtra("retorno"));


    }
}
```

No arquivo activity_main.xml, crie um componente Button com as seguintes propriedades:

- id: btnArgumento
- text: Enviar Argumento

Para sua referência, aqui está o código completo do arquivo activity_main.xml, após as modificações:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.com.grupouninter.aula3.activityresult.MainActivity">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="A opção selecionada foi:"
    android:id="@+id/titulo"
    android:layout_marginTop="53dp"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"
    android:layout_marginStart="27dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="OPÇÃO"
    android:id="@+id/resultado"
    android:layout_below="@+id/titulo"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp" />



<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Opções"
    android:id="@+id/btnOpcao"
    android:layout_below="@+id/resultado"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="42dp" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enviar Argumento"
    android:id="@+id/btnArgumento"
    android:layout_below="@+id/btnOpcao"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="49dp" />
</RelativeLayout>
```

No arquivo MainActivity.java, devemos agora criar o objeto que representa o novo botão, gerar seu vínculo com o arquivo de Layout e também redefinir o comportamento de seu listener OnClickListener: Button **btnArgumento**;

No método onCreate, vamos instanciar o objeto btnArgumento

btnOpcao = (Button) findViewById(R.id.***btnArgumento***);

E finalmente redefinir seu Listener:

```
btnArgumento.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```

Agora crie nossa Intent, passando o contexto e a classe que servirá como modelo para a Activity:

```
Intent intent = new Intent(view.getContext(), TerceiraActivity.class);
```

Devemos passar a esta intent o valor que desejamos enviar, no caso, um "extra"

```
intent.putExtra("retorno", "SEU ARGUMENTO CHEGOU!");
```

Em uma nota rápida, o método putExtra pode ser sobrecarregado, ou seja, pode ter seu comportamento alterado de acordo com o tipo de argumento passado, para os tipos Integer, byte, char, boolean, etc. No entanto, o método getStringExtra() não é sobrecarregado, mas sim encarrega-se de tentar converter o valor passado para o tipo desejado, portanto muita atenção à forma como enviamos o tipo de dado desejado.

Finalmente, iniciamos a Activity. Podemos iniciar a Activity diretamente ou então solicitando seu retorno. O processo para ambas permanece o mesmo anteriormente discutido.

```
startActivity(intent);
```

Para sua referência, aqui está o código completo do arquivo MainActivity.java, após as modificações propostas:

```
package br.com.grupouninter.aula3.activityresult;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView resultado;
    Button btnOpcao;
    Button btnArgumento;
    static int ACAO_BUSCA_PREFERENCIA_USER = 1;


  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // ligamos o componente btnOpcao à representação no arquivo xml
      btnOpcao = (Button) findViewById(R.id.btnOpcao);
      btnArgumento = (Button) findViewById(R.id.btnArgumento);
      // redefinimos o listener OnClickListener
      btnOpcao.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              Intent intent = new Intent(view.getContext(), SegundaActivity.class);
              startActivityForResult(intent,
ACAO_BUSCA_PREFERENCIA_USER);
          }
      });

      btnArgumento.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
        public void onClick(View view) {
            Intent intent = new Intent(view.getContext(), TerceiraActivity.class);
            intent.putExtra("retorno", "SEU ARGUMENTO CHEGOU!");
            startActivity(intent);
        }
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data){
    // verificando o tipo de request code
    if (requestCode == ACAO_BUSCA_PREFERENCIA_USER){
        // verificando se o usuário encerrou a activity retornando sua opcao
        if (resultCode == RESULT_OK){
            // ligamos o componente resultado à representação no arquivo xml
            resultado = (TextView) findViewById(R.id.resultado);
            // passamos o valor da Intent (uma String) para resultado.setText()
            resultado.setText(data.getStringExtra("retorno"));
        }
    }
}
}
```

Execute seu aplicativo. O que aconteceu quando você clicou no botão “Enviar Argumento”?

Uma nova Activity (TerceiraActivity) foi iniciada, e nela o componente TextView teve seu valor alterado. No entanto temos uma falha de design na Terceira Activity que não podemos deixar de discutir.

Como você deve ter percebido, aplicativos Android possuem vários “pontos de entrada”, ou seja, formas como o usuário pode dar início a interação com seu aplicativo. Você pode, por exemplo, permitir que outros aplicativos iniciem uma Activity específica de em seu APP, sem passar pela Activity Principal. Portanto, não podemos simplesmente supor que o argumento que aguardamos será passado por padrão à inicialização de nossa Activity.

Devemos então tratar esta situação apropriadamente. Em seu arquivo TerceiraActivity.java, faça a seguinte alteração:

```
if ((intent.hasExtra("retorno")) && (null != intent.getStringExtra("retorno")))
   valorParametro.setText(intent.getStringExtra("retorno"));
```

Ou seja: caso a Intent que deu origem à nossa Activity possua um extra chamado “retorno” e caso este extra não seja **null** (ou seja, foi instanciado corretamente), mostraremos o valor passado em nossa TextView. Para sua referência, segue o código completo da TerceiraActivity.java, após a correção sugerida:

```
package br.com.grupouninter.aula3.activityresult;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class TerceiraActivity extends AppCompatActivity {

    TextView valorParametro;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_terceira);

        valorParametro = (TextView) findViewById(R.id.valorParametro);

        Intent intent = getIntent();
        if ((intent.hasExtra("retorno")) && (null !=
intent.getStringExtra("retorno")))
            valorParametro.setText(intent.getStringExtra("retorno"));

    }
}
```

Lembre-se que podemos passar outros tipos “comuns” à linguagem, como booleans, integers, chars, bytes entre outros. Mas, como fazer quando preciso passar como argumento um objeto criado especificamente para este projeto?

Saiba mais sobre o processo de envio de parâmetros para uma Activity no material online.

Tema 3 - Passando objetos de uma Activity para outra

À medida que vamos desenvolvendo aplicativos, é natural que encontremos obstáculos que não podem ser superados utilizando-se os tipos de dados padrão da linguagem. Naturalmente necessitaremos criar novos objetos, que agrupem ou estendam funcionalidades. E, eventualmente, necessitaremos passar estes objetos como argumentos de uma Activity para outra. Então, antes de mais nada, vamos à definição de um cenário: nosso aplicativo agora necessita armazenar os dados comuns à um cliente, como nome, endereço e idade.

Para um único cliente, a saída mais simples seria a criação de uma variável String que represente o Nome, outra que represente o Endereço, além de uma variável Integer que represente sua Idade. O problema acontece quando necessitamos armazenar os dados de múltiplos clientes. Então, antes de mais nada, vamos criar uma classe que represente estes atributos e que encapsule corretamente os mesmos. Em sua janela de Projetos, clique com o botão direito em cima da pasta br.com.grupouninter.aula3.activityresult e então crie uma nova classe java chamada Cliente.

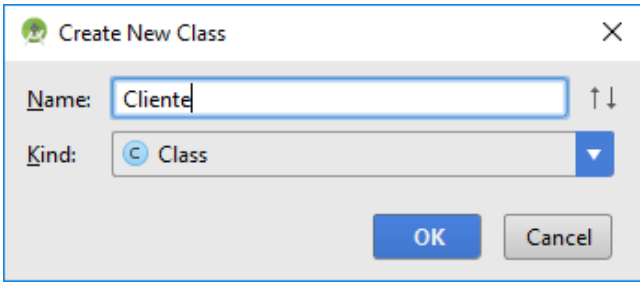

Figura 13 – Classe Cliente

Clique em OK e o arquivo Cliente.java será criado. Diferentemente dos exemplos anteriores, esta classe não estende a classe Activity (ou qualquer uma de suas filhas). Desejamos aqui criar um POJO (Plain Old Java Object), que represente um cliente e suas Propriedades.

Aqui está o conteúdo de Cliente.java para sua referência:

```
package br.com.grupouninter.aula3.activityresult;

/**
 * Created by Marcelo on 07/08/2016.
 */
public class Cliente {

    private String nome;
    private String endereco;
    private Integer idade;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }


    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

    public Integer getIdade() {
        return idade;
    }

    public void setIdade(Integer idade) {
        this.idade = idade;
    }
}
```

Se você tem alguma dificuldade em compreender o conceito de Objetos, Classes, Encapsulamento ou até mesmo POJO, é importantíssimo que você pesquise a respeito de Orientação à Objetos. Como sugestão, no livro A Linguagem de Programação Java, 4. Edição - ARNOLD, Ken; GOSLING, James; HOLMES, David - disponível em seu ambiente de aprendizagem, em seu capítulo 2, na página 62 oferece um excelente lugar para aumentar seu entendimento a respeito deste aspecto fundamental da programação.

Agora que criamos nosso POJO Cliente, vamos instanciá-lo em nossa MainActivity, definir seus valores e finalmente passá-lo como argumento para nossa TerceiraActivity.

No arquivo activity_main.xml, crie um novo componente Button com as seguintes propriedades:

- id: btnEnviaPojo
- Text: Enviar Pojo

Para sua referência, segue o arquivo XML após as alterações sugeridas:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="br.com.grupouninter.aula3.activityresult.MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="A opção selecionada foi:"
      android:id="@+id/titulo"
      android:layout_marginTop="53dp"
      android:layout_alignParentTop="true"
      android:layout_alignParentStart="true"
      android:layout_marginStart="27dp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="OPÇÃO"
    android:id="@+id/resultado"
    android:layout_below="@+id/titulo"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Opções"
    android:id="@+id/btnOpcao"
    android:layout_below="@+id/resultado"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="42dp" />



    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar Argumento"
        android:id="@+id/btnArgumento"
        android:layout_below="@+id/btnOpcao"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="49dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar POJO"
        android:id="@+id/btnEnviaPojo"
        android:layout_below="@+id/btnArgumento"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="61dp" />
</RelativeLayout>
```

Retorne agora ao arquivo MainActivity.java e crie o Objeto que representará este novo botão:

```
Button btnEnviaPojo;
```

Instancie o mesmo no método onCreate()

btnEnviaPojo = (Button) findViewById(R.id.***btnEnviaPojo***);

E finalmente redefina o comportamento de seu Listener OnClickListener();

```
btnEnviaPojo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```

Quando este botão for pressionado, desejamos que seja passado à Terceira Activity o valor de um objeto criado a partir da classe Cliente. Portanto, antes de mais nada vamos instanciar a classe cliente e preencher seus valores:

```
btnEnviaPojo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Cliente cliente = new Cliente();
        cliente.setNome("Marcelo");
        cliente.setEndereco("Rua tal, numero tal");
        cliente.setIdade(40);

    }
});
```

Agora devemos criar nossa Intent, passar este objeto como extra para a mesma e finalmente iniciar a Terceira Activity.

```
Intent intent = new Intent(view.getContext(), TerceiraActivity.class);
intent.putExtra("cliente", cliente);
startActivity(intent);
```

No entanto, perceba que agora quanto tentamos utilizar o método putExtra, somos informados de um erro. Conforme foi explicado anteriormente, o método putExtra pode ser sobrecarregado para tipos definidos, mas lembre-se que o POJO que acabamos de criar é um tipo completamente diferente, portanto não previsto pela linguagem. Para que possamos utilizar nosso POJO (ou qualquer classe personalizada que tenhamos criado), é necessário que

tornemos a mesma serializável, ou seja, que possa ser convertida à uma série de bytes, de forma que possa ser armazenada ou enviada através de um link de comunicações.

Esta serialização pode ser desserializada, ou seja, convertida em uma réplica do objeto original.

```
public class Cliente implements Serializable {
```

Ao tornamos a classe Cliente serializável, nosso método putExtra() entende o tipo de objeto (serializable) e aceita o sobrecarregamento do método. O que nos resta agora é alterar o comportamento de nossa classe Terceira Activity, para que a mesma esteja pronta para receber nosso POJO serializado, o desserializar e finalmente utilizar suas propriedades:

```
...
Intent intent = getIntent();
if ((intent.hasExtra("retorno")) && (null!= intent.getStringExtra("retorno")))
   valorParametro.setText(intent.getStringExtra("retorno"));
else if ((intent.hasExtra("cliente")) && (null !=
intent.getSerializableExtra("cliente"))){
   Cliente cliente = (Cliente) intent.getSerializableExtra("cliente");
   valorParametro.setText("Cliente informado: " + cliente.getNome() + " - " +
cliente.getIdade() + " anos");
}
...
```

Um ponto importante a comentarmos é o fato de agora utilizamos o método getSerializableExtra(), e que este nos retorna um objeto serializado, ou seja, sem forma definida. Para que possamos utilizá-lo, é necessário efetuarmos uma conversão de tipo, de objeto serializado para o objeto Cliente.

Muito cuidado ao efetuar este tipo de conversão, uma vez que a verificação de que o objeto pode ser serializado para o formato desejado só ocorre em tempo de execução, ou seja, eventualmente podemos tentar serializar o objeto convertendo-o para o tipo incorreto, o que gerará uma mensagem de erro.

Execute seu projeto agora!

No material online, o professor Marcelo fala sobre a passagem de objetos de uma Activity para outra.

Síntese

Nesta aula foram apresentados os conceitos de retorno de resultados para Activities, envio de argumentos entre Activities, criação de objetos vinculados ao layout xml, criação de classes POJO, serialização das mesmas e seu envio entre diferentes Activities.

Ao final desta aula o aluno está apto a vincular informações entre Activities, independente de sua tipificação, bem como receber respostas das mesmas.

Confira a síntese do professor Marcelo sobre os conteúdos abordados no material online.

Referências

ARNOLD, Ken; GOSLING, James; HOLMES, David. A **Linguagem de Programação Java**. 4. Edição. 2009.

DEITEL, Paul; DEITEL, Harvey; OSWALD, Alexander. **Android 6 para Programadores** – Uma Abordagem Baseada em Aplicativos. Boockman: 2015.