sãojudas universidade	FTCE Faculdade de Tecnologia e Ciências Exatas

USJT - 2019 – Desenvolvimento de Sistemas para Dispositivos Móveis

Professor: Bossini

Aula: 04
Assunto: OO e Reciclagem de views

Passo 1 (Ajustando o projeto para usar OO) Nossos objetos de interesse são meras Strings. Vamos começar a aplicar orientação a objetos criando as classes Chamado e Fila, como mostram as listagens 1.1 e 1.2.

Listagem 1.1

```
import java.util.Date;
public class Chamado {
    private Fila fila;
    private String descricao;
    private Date dataAbertura;
    private Date dataFechamento;
    private String status;
    public Fila getFila() {
        return fila;
    }
    public void setFila(Fila fila) {
        this.fila = fila;
    }
    public String getDescricao() {
        return descricao;
    }
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
    public Date getDataAbertura() {
        return dataAbertura;
    }
    public void setDataAbertura(Date dataAbertura) {
        this.dataAbertura = dataAbertura;
    }
    public Date getDataFechamento() {
        return dataFechamento;
    }
    public void setDataFechamento(Date dataFechamento) {
        this.dataFechamento = dataFechamento;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}
```

Listagem 1.2

```
import java.util.List;
public class Fila {
    private String nome;
    private int iconId;
    private List<Chamado> chamados;
    public List<Chamado> getChamados() {
        return chamados;
    }
    public void setChamados(List<Chamado> chamados) {
        this.chamados = chamados;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public int getIconId() {
        return iconId;
    }
    public void setIconId(int iconId) {
        this.iconId = iconId;
    }
}
```

Passo 2 (Adicionando os ícones para as filas) A fim de exibir as filas em um layout mais interessante, vamos adicionar ícones que as representem. No Android Studio, clique com o direito em drawable e escolha new >> vector asset. Adicione os seguintes ícones ao projeto.

- network_check (para a fila de redes)
- computer (para a fila de Desktops)
- phone in talk (para a fila de telefones)
- poll (para a fila de servidores)
- new releases (para a fila de novos projetos)
- power (para a fila de manutenção)

Passo 3 (Criando a nova base) Na classe ListaChamadosActivity simulamos uma base na memória, com strings. Ela continuará na memória principal nesse momento, porém utilizando objetos.

3.1 A Listagem 3.1 mostra sua criação. O método geraListaChamados deve substituir o existente.

Listagem 3.1

```
public List <Chamado> geraListaChamados(){
    List <Chamado> chamados = new ArrayList<>();
    chamados.add(new Chamado (
            new Fila ("Desktops",
R.drawable.ic_computer_black_24dp),
            "Computador da secretária quebrado.",
            new Date(),
```

```
            null,
            "Aberto"
    ));
    chamados.add(new Chamado (
            new Fila ("Telefonia",
R.drawable.ic_phone_in_talk_black_24dp),
            "Telefone não funciona.",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Redes",
R.drawable.ic_network_check_black_24dp),
            "Manutenção no proxy.",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Servidores", R.drawable.ic_poll_black_24dp),
            "Lentidão generalizada.",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Novos Projetos",
R.drawable.ic_new_releases_black_24dp),
            "CRM",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Novos Projetos",
R.drawable.ic_new_releases_black_24dp),
            "Gestão de Orçamento",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Redes",
R.drawable.ic_network_check_black_24dp),
            "Internet com lentidão",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
            new Fila ("Novos Projetos",
R.drawable.ic_new_releases_black_24dp),
            "Chatbot",
            new Date(),
            null,
            "Aberto")
    );
    chamados.add(new Chamado (
```

```
            new Fila ("Novos Projetos",
R.drawable.ic_new_releases_black_24dp),
            "Chatbot",
            new Date(),
            null,
            "Aberto")
    );
    return chamados;
}
```

Note que usamos dois construtores diferentes do padrão. Um da classe Fila e outro da classe Chamado. Vá em frente e crie-os também.

3.2 Com isso, o método de busca deixou de compilar, pois ele ainda lida com coleções de strings. Ajuste-o como mostra a Listagem 3.2.

Listagem 3.2

```
public List<Chamado> busca (String nomeFila){
    List <Chamado> chamados = geraListaChamados();
    if (nomeFila == null || nomeFila.length() == 0)
        return chamados;
    List <Chamado> resultado = new ArrayList<>();
    for (Chamado chamado : chamados){
        if (chamado.getFila().getNome().
                        toLowerCase().contains(nomeFila.toLowerCase())){
            resultado.add(chamado);
        }
    }
    return resultado;
}
```

3.3 O método onCreate também precisa de alterações, pela mesma razão. Veja sua nova implementação na Listagem 3.3.

Listagem 3.3

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lista_chamados);
    chamadosListView = findViewById(R.id.chamadosListView);
    Intent origemIntent = getIntent();
    String nomeFila = origemIntent.getStringExtra("nome_fila");
    final List <Chamado> chamados = busca(nomeFila);
    ArrayAdapter <Chamado> adapter =
            new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, chamados);
    chamadosListView.setAdapter(adapter);
    chamadosListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
            Chamado chamadoSelecionado = chamados.get(position);
            Intent intent = new Intent (ListaChamadosActivity.this,
DetalhesChamadoActivity.class);
            intent.putExtra("chamado_selecionado", chamadoSelecionado);
            startActivity(intent);
        }
    });
}
```

3.4 Para que um chamado possa ser passado por meio de um Intent, é preciso fazer com que sua classe implemente a interface Serializable (também dá pra fazer com Parcelable, vale a pena pesquisar). Como Chamado usa a classe Fila, ela também precisa ser marcada como Serializable.Veja a Listagem 3.4.

Listagem 3.4

```
public class Chamado implements Serializable {

public class Fila implements Serializable {
```

Passo 4 (Sobrescrita do método toString de Fila) Caso executemos a aplicação agora, poderemos ver que a lista exibe o retorno do método toString de cada chamado em cada item. Caso queiramos, podemos sobrescrever o método toString para mudar essa texto. Veja a Listagem 4.1.

Listagem 4.1

```
@Override
public String toString() {
    return this.getDescricao();
}
```

Execute a aplicação para ver o resultado. Ela ainda irá falhar na segunda transição de telas.

Passo 5 (Layout para exibir os chamados) Vamos criar um novo arquivo de layout, chamado list_item.xml (o nome é arbitrário). Ele ficará na pasta res/layout e servirá para dizer como um chamado deve ser exibido. Para tal, clique com o direito em res/layout e escolha new >> layout resource file. Veja a Listagem 5.1.

Listagem 5.1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <ImageView
        android:id="@+id/filaIconImageView"
        android:layout_width="@dimen/fila_icon_width"
        android:layout_height="wrap_content"
        app:srcCompat="@mipmap/ic_launcher" />
    <GridLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:columnCount="3"
        android:rowCount="2">
        <TextView
            android:id="@+id/descricaoChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_columnSpan="3"
            android:layout_gravity="fill_horizontal"
            android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
            android:gravity="center"
            android:layout_row="0"
            android:layout_column="0"/>
```

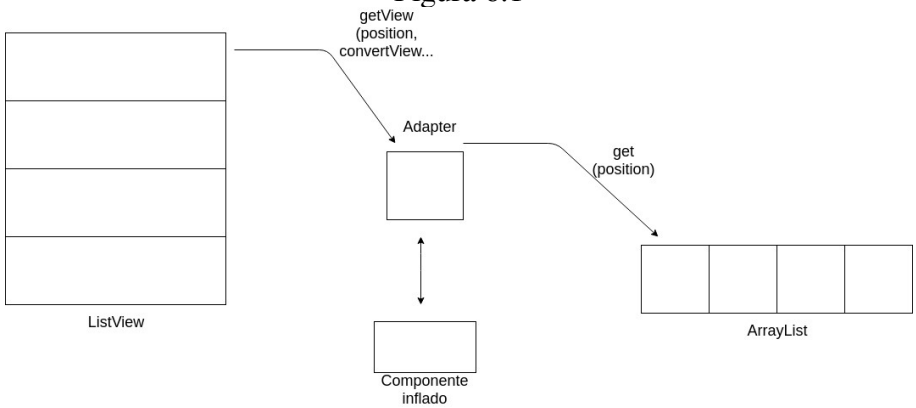
```
        <TextView
            android:id="@+id/statusChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
            android:layout_row="1"
            android:layout_gravity="center"
            android:layout_column="0"/>
        <TextView
            android:id="@+id/dataAberturaChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
            android:layout_row="1"
            android:gravity="center"
            android:layout_column="1"/>
        <TextView
            android:id="@+id/dataFechamentoChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
            android:layout_row="1"
            android:gravity="center"
            android:layout_column="2"/>
    </GridLayout>
</LinearLayout>

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <ImageView
        android:id="@+id/filaIconImageView"
        android:layout_width="@dimen/fila_icon_width"
        android:layout_height="match_parent"
        app:srcCompat="@mipmap/ic_launcher" />
    <GridLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:columnCount="3"
        android:rowCount="2">
        <TextView
            android:id="@+id/descricaoChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:layout_columnSpan="3"
            android:layout_gravity="fill_horizontal"
            android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
            android:gravity="center"
            android:layout_row="0"
            android:layout_column="0"/>
        <TextView
            android:id="@+id/statusChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
            android:layout_row="1"
            android:layout_gravity="center"
            android:layout_column="0"/>
        <TextView
            android:id="@+id/dataAberturaChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
            android:layout_row="1"
            android:gravity="center"
            android:layout_column="1"/>
        <TextView
            android:id="@+id/dataFechamentoChamadoNaFilaTextView"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_columnWeight="1"
```

```
            android:layout_row="1"
            android:gravity="center"
            android:layout_column="2"/>
    </GridLayout>
</LinearLayout>
```

Passo 6 (Usando o novo layout) Agora precisamos usar o layout recém criado. Para isso, é necessário escrever um novo Adapter e fazer a sobrescrita do método getView. Isso ocorre devido à forma como se dá a interação de ListView e Adapter.

6.1 Veja a Figura 6.1.

Figura 6.1



6.2 Assim, iremos sobrescrever o método getView de modo que o Adapter infle o layout que criamos. Veja a Listagm 6.1.

Listagem 6.1

```
public class ChamadoArrayAdapter extends ArrayAdapter <Chamado> {
    public ChamadoArrayAdapter (Context context, List<Chamado> chamados){
        super (context, -1, chamados);
    }
    @Override
    public View getView(int position,View convertView,  ViewGroup parent) {
        Chamado chamadoDaVez = getItem(position);
        Fila filaDaVez = chamadoDaVez.getFila();
        LayoutInflater inflater = LayoutInflater.from(getContext());
        View view = inflater.inflate(R.layout.list_item, parent, false);
        ImageView filaIconImageView = view.findViewById(R.id.filaIconImageView);
        TextView descricaoChamadoNaFilaTextView =
                view.findViewById(R.id.descricaoChamadoNaFilaTextView);
        TextView statusChamadoNaFilaTextView =
                view.findViewById(R.id.statusChamadoNaFilaTextView);
        TextView dataAberturaChamadoNaFilaTextView =
                view.findViewById(R.id.dataAberturaChamadoNaFilaTextView);
        TextView dataFechamentoChamadoNaFilaTextView =
                view.findViewById(R.id.dataFechamentoChamadoNaFilaTextView);
        filaIconImageView.setImageResource(filaDaVez.getIconId());
       descricaoChamadoNaFilaTextView.setText(chamadoDaVez.getDescricao());
       statusChamadoNaFilaTextView.setText(chamadoDaVez.getStatus());

dataAberturaChamadoNaFilaTextView.setText(DateHelper.format(chamadoDaVez.getDataAbert
ura()));
       if (chamadoDaVez.getDataFechamento() != null){

dataFechamentoChamadoNaFilaTextView.setText(DateHelper.format(chamadoDaVez.getDataFec
hamento()));
       }
```

```
        return view;
    }
}
```

6.3 Também é necessário ajustar o método MainActivity para que ele instancie um adapter novo. Veja a Listagem 6.2.

Listagem 6.2

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lista_chamados);
    chamadosListView = findViewById(R.id.chamadosListView);
    Intent origemIntent = getIntent();
    String nomeFila = origemIntent.getStringExtra("nome_fila");
    final List <Chamado> chamados = busca(nomeFila);
    ChamadoArrayAdapter  adapter =
     new ChamadoArrayAdapter(this, chamados);
  chamadosListView.setAdapter(adapter);
    chamadosListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
            Chamado chamadoSelecionado = chamados.get(position);
            //Toast.makeText(ListaChamadosActivity.this, chamadoSelecionado,
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent (ListaChamadosActivity.this,
DetalhesChamadoActivity.class);
            intent.putExtra("chamado_selecionado", chamadoSelecionado);
            startActivity(intent);
        }
    });
}
```

6.4 Veja que estamos utilizando uma classe chamada DateHelper para auxiliar na formatação de datas. Seu código é exibido na Listagem 6.3.

Listagem 6.3

```
public class DateHelper {
    private static SimpleDateFormat sdf = new
                                SimpleDateFormat("dd/MM/yyyy hh:mm");
    public static String format (Date date ){
        return sdf.format(date);
    }
```

Passo 7 (A nova tela de detalhes para um chamado) Precisamos adaptar o arquivo activity_detalhes_chamado.xml para que ele exiba todos os dados que um chamado possui. Ela simplesmente reutilizará o layout que criamos para um item na lista.

7.1 A Listagem 7.1 mostra o XML.

Listagem 7.1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DetalhesChamadoActivity"
    android:orientation="vertical">
    <include layout="@layout/list_item"/>
</LinearLayout>
```

7.2 A Listagem 7.2 mostra a classe DetalhesChamadoActivity adaptada, de modo que faça uso adequado da nova tela.

Listagem 7.2

```
public class DetalhesChamadoActivity extends AppCompatActivity {
    private TextView nomeFilaTextView, descricaoChamadoTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalhes_chamado);
        Intent origemIntent = getIntent();
        Chamado chamadoSelecionado =
                (Chamado)
origemIntent.getSerializableExtra("chamado_selecionado");
        ImageView filaIconImageView =
findViewById(R.id.filaIconImageView);
        TextView descricaoChamadoNaFilaTextView =
                findViewById(R.id.descricaoChamadoNaFilaTextView);
        TextView statusChamadoNaFilaTextView =
                findViewById(R.id.statusChamadoNaFilaTextView);
        TextView dataAberturaChamadoNaFilaTextView =

findViewById(R.id.dataAberturaChamadoNaFilaTextView);
        TextView dataFechamentoChamadoNaFilaTextView =

findViewById(R.id.dataFechamentoChamadoNaFilaTextView);

filaIconImageView.setImageResource(chamadoSelecionado.getFila().getI
conId());

descricaoChamadoNaFilaTextView.setText(chamadoSelecionado.getDescric
ao());

statusChamadoNaFilaTextView.setText(chamadoSelecionado.getStatus());
        dataAberturaChamadoNaFilaTextView.setText(

DateHelper.format(chamadoSelecionado.getDataAbertura()));
        if (chamadoSelecionado.getDataFechamento() != null){
            dataFechamentoChamadoNaFilaTextView.setText(

DateHelper.format(chamadoSelecionado.getDataFechamento()));
        }
    }
}
```

Passo 8 (O que há de errado com o projeto? Usando a view convertView) O componente ListView funciona em conjunto com um Adapter. O Adapter, por sua vez, conhece a coleção de dados que a ListView deseja exibir e os fornece em um layout inflado apropriadamente. Veja a Figura 6.1 novamente.

A interação se dá da seguinte forma.

8.1 Quando a coleção é alterada, o Adapter avisa a ListView, que é sua observadora.

8.2 A ListView reage disparando mensagens (ou chamando o método, dá na mesma) getView ao Adapter, uma vez para cada posição.

8.3 A cada mensagem recebida, o Adapter
- 8.1 Obtém o componente adequado na coleção de dados
- 8.2 Infla um componente visual para colocar seus dados
- 8.3 Devolve o componente inflado com os dados de interesse para a ListView.

Embora funcione, essa solução falha em reutilizar componentes previamente inflados. Ler um arquivo XML e dar origem a uma árvore de componentes visuais é uma das atividades mais caras computacionalmente e estamos fazendo isso muitas vezes sem necessidade. Por exemplo, o que ocorre com o componente visual que está sendo exibido no topo da ListView quando o usuário a desliza um pouco. Ele é jogado fora e um outro é inflado. Uma solução muito melhor seria mantê-lo ali e simplesmente trocar os dados existentes com os dados do novo componente a ser exibido naquela região.

Note, ainda na Figura 6.1, que a ListView envia ao Adapter um componente do tipo View, chamado convertView (é só o nome que foi usado na definição do método, pode ser qualquer outro, não é isso que importa). Quando uma mensagem getView é enviada ao Adapter, a ListView envia duas coisas: uma posição e uma view. Precisamos adaptar nossa solução para usar essa view adequadamente. Ela funciona assim. Se a posição pedida pela ListView ainda não tiver uma view inflada, convertView vale null. Caso contrário, a ListView envia a view previamente inflada e o Adapter pode reutizá-la, evitando muitas chamadas ao método inflate desnecessárias e caras. A Listagem 8.1 mostra essa adaptação.

Listagem 8.1

```
@Override
public View getView(int position,View convertView,  ViewGroup
parent) {
    Chamado chamadoDaVez = getItem(position);
    Fila filaDaVez = chamadoDaVez.getFila();
    LayoutInflater inflater = LayoutInflater.from(getContext());
    if (convertView == null){
        convertView = inflater.inflate(R.layout.list_item, parent,
false);
    }
    ImageView filaIconImageView =
convertView.findViewById(R.id.filaIconImageView);
    TextView descricaoChamadoNaFilaTextView =

convertView.findViewById(R.id.descricaoChamadoNaFilaTextView);
    TextView statusChamadoNaFilaTextView =

convertView.findViewById(R.id.statusChamadoNaFilaTextView);
    TextView dataAberturaChamadoNaFilaTextView =

convertView.findViewById(R.id.dataAberturaChamadoNaFilaTextView);
    TextView dataFechamentoChamadoNaFilaTextView =

convertView.findViewById(R.id.dataFechamentoChamadoNaFilaTextView);
    filaIconImageView.setImageResource(filaDaVez.getIconId());

descricaoChamadoNaFilaTextView.setText(chamadoDaVez.getDescricao());
   statusChamadoNaFilaTextView.setText(chamadoDaVez.getStatus());

dataAberturaChamadoNaFilaTextView.setText(DateHelper.format(chamadoD
aVez.getDataAbertura()));
   if (chamadoDaVez.getDataFechamento() != null){

dataFechamentoChamadoNaFilaTextView.setText(DateHelper.format(chamad
oDaVez.getDataFechamento()));
   }
   return convertView;
}
```

Exercício Prático

1. Ajuste sua aplicação para usar objetos do tipo localização. Cada localização tem uma latitude e uma longitude. Faça os ajustes necessários para que um adapter personalizado faça uso correto do convertView. Siga o passo a passo da aula para isso.

Bibliografia

Android API, package android.app; disponível em http://developer.android.com/reference/android/app/package-summary.html ; consultado em 02/09/15.