Ficha do Aplicativo

App05 – Listeners
Resumo Este é um aplicativo para aprendermos sobre Eventos e Listeners. Até agora temos usado apenas o evento de clique sobre botões (Button) e caixas de texto (TextView). Porém é preciso saber que existem outros tipos de eventos e é preciso saber como responder a eles.
Objetivos de Aprendizagem 1. Praticar o uso de classe Java puro já visto anteriormente 2. Implementar Listeners para tratamento de eventos 3. Usar a fila de mensagens com o emprego das classes Handler e Runnable
Dinâmica do Aplicativo Este aplicativo contém quatro botões com diferentes implementações do evento OnClick(). Ao clicar em um dos botões será gerado um número aleatório que representará um ano escolar do período fundamental. Em seguida dois componentes TextView serão preenchidos com o número do ano e a respectiva lista de material escolar. Há um terceiro TextView para mostrar o status de ação feita pelo usuário quando o listener `View.OnTouchListener()` for implementado.

Lista de Activities do Aplicativo

Nome	Layout
MainActivity.java	activity_main.xml

Drawables deste Aplicativo

Nome do arquivo	Dimensões	Tipo	Imagem
perfil_botoes.xml	24 x 24	XML – Vector	
opc1.xml	24 x 24	XML – Vector	
opc2.xml	24 x 24	XML – Vector	
opc3.xml	24 x 24	XML – Vector	
opc4.xml	24 x 24	XML – Vector	
listeners.png	740 x 720	Arquivo PNG	

Resources

strings.xml

```
<resources>
    <string name="app_name">Listeners</string>
    <string name="txtopcs">4 Opções de Listeners - Toque um botão para ver o Material Escolar</string>
</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="corActivity">#71F7A7</color>
    <color name="corBarra">#FFD0D0D0</color>
    <color name="corBotao">#EBDEFFEB</color>
    <color name="corTxt">#FFDAFCF2</color>
    <color name="corPisca">#C6CDFB</color>
</resources>
```

Detalhamento da Activity

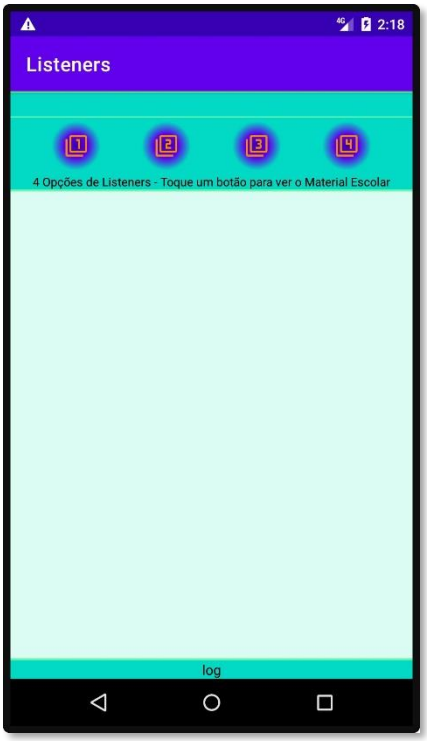
Layout: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:background="@color/corActivity"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtAno"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:background="@color/teal_200"
        android:gravity="center"
        android:textColor="@color/black"
        android:textSize="18sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="..." />

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="74dp"
        android:layout_marginTop="1dp"
        android:background="@color/teal_200"
        android:gravity="bottom|center_horizontal"
        android:text="@string/txtopcs"
        android:textColor="@color/black"
        android:textSize="12sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtAno" />

    <ImageButton
        android:id="@+id/btnOpc1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



```
        android:layout_marginTop="4dp"
        android:background="@drawable/perfil_botoes"
        android:onClick="btnOpc1OnClick"
        android:tint="#FB8C00"
        app:layout_constraintEnd_toStartOf="@+id/btnOpc2"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView"
        app:srcCompat="@drawable/opc1" />

    <ImageButton
        android:id="@+id/btnOpc2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/perfil_botoes"
        android:tint="#FB8C00"
        app:layout_constraintEnd_toStartOf="@+id/btnOpc3"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnOpc1"
        app:layout_constraintTop_toTopOf="@+id/btnOpc1"
        app:srcCompat="@drawable/opc2" />

    <ImageButton
        android:id="@+id/btnOpc3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/perfil_botoes"
        android:tint="#FB8C00"
        app:layout_constraintEnd_toStartOf="@+id/btnOpc4"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnOpc2"
        app:layout_constraintTop_toTopOf="@+id/btnOpc2"
        app:srcCompat="@drawable/opc3" />

    <ImageButton
        android:id="@+id/btnOpc4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/perfil_botoes"
        android:tint="#FB8C00"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnOpc3"
        app:layout_constraintTop_toTopOf="@+id/btnOpc3"
        app:srcCompat="@drawable/opc4" />

    <TextView
        android:id="@+id/txtMaterial"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="2dp"
        android:layout_marginBottom="2dp"
        android:background="@color/corTxt"
        android:padding="8dp"
        android:textIsSelectable="true"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/txtLogTouch"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <TextView
        android:id="@+id/txtLogTouch"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/teal_200"
        android:gravity="center"
        android:text="log"
        android:textColor="@color/black"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Código: MainActivity.java

```java
/* Observação:
   O nome do package está omitido, porque ao fazer o seu projeto esse nome será outro.
   Os imports também estão omitidos. O motivo para isso é que a cada nova versão do Android Studio pode
   haver mudanças nos nomes dos caminhos da biblioteca. */

public class MainActivity extends AppCompatActivity {

    private Random gerador;
    private int sorteado;
    private TextView txtAno;
    private TextView txtMaterial;
    // btnOpc1 não é necessário aqui porque o onClick dele foi configurado no XML
    private ImageButton btnOpc2;
    private ImageButton btnOpc3;
    private ImageButton btnOpc4;
    private TextView txtLogTouch;

    private int yIni;
    public int getyIni() {
        return yIni;
    }
    public void setyIni(int yIni) {
        this.yIni = yIni;
    }

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

        gerador = new Random();
        sorteado = 0;

        txtAno = findViewById(R.id.txtAno);
        txtMaterial = findViewById(R.id.txtMaterial);
        btnOpc2 = findViewById(R.id.btnOpc2);
        btnOpc3 = findViewById(R.id.btnOpc3);
        btnOpc4 = findViewById(R.id.btnOpc4);
        txtLogTouch = findViewById(R.id.txtLogTouch);

        setBtnOpc2OnClick();
        setBtnOpc3OnClick();
        setBtnOpc4OnClick();

        setTxtMaterialOnLongClick();
        setTxtMaterialOnTouch();
    }

    private int sorteia() {
        int i;
        do {
            i = gerador.nextInt(MaterialEscolar.qtdeMe);
        } while (i == sorteado);
        return i;
    }

    //  Este é o modo como temos feito até este ponto do curso
    public void btnOpc1OnClick(View view) {
        sorteado = sorteia();
        txtAno.setText(MaterialEscolar.me[sorteado].getAno());
        txtMaterial.setText(MaterialEscolar.me[sorteado].getMaterial());
    }

    //  Este é um modo não muito usado, no qual declaramos explicitamente um objeto da classe View.OnClickListener
    private void setBtnOpc2OnClick() {
        View.OnClickListener listener;
        listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                sorteado = sorteia();
                txtAno.setText(MaterialEscolar.me[sorteado].getAno());
                txtMaterial.setText(MaterialEscolar.me[sorteado].getMaterial());
            }
        };
        btnOpc2.setOnClickListener(listener);
    }
```

```
    //  Este é o modo usual, no qual utilizamos um objeto anônimo
    private void setBtnOpc3OnClick() {
        btnOpc3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                sorteado = sorteia();
                txtAno.setText(MaterialEscolar.me[sorteado].getAno());
                txtMaterial.setText(MaterialEscolar.me[sorteado].getMaterial());
            }
        });
    }

    //  É possível usar função lambda de Java 8 em Listeners - tendência futura, é bom conhecer
    private void setBtnOpc4OnClick() {
        btnOpc4.setOnClickListener((View view) -> {
            sorteado = sorteia();
            txtAno.setText(MaterialEscolar.me[sorteado].getAno());
            txtMaterial.setText(MaterialEscolar.me[sorteado].getMaterial());
        });
    }

    private void setTxtMaterialOnLongClick() {
        txtMaterial.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View view) {
                String s = txtMaterial.getText().toString();
                if (s != "") {
                    txtMaterial.setBackgroundResource(R.color.corPisca);
                    ClipboardManager clipboard = (ClipboardManager)getSystemService(Context.CLIPBOARD_SERVICE);
                    ClipData clip = ClipData.newPlainText("Lista de Material", txtMaterial.getText().toString());
                    clipboard.setPrimaryClip(clip);
                    runRestauraCor();
                    return true;
                }
                return false;
            }
        });
    }

    private void runRestauraCor() {
        Handler handler = new Handler(Looper.getMainLooper());
        Runnable restauraCor = new Runnable() {
            @Override
            public void run() {
                Toast.makeText(MainActivity.this, "Lista de Material Copiada", Toast.LENGTH_LONG).show();
                txtMaterial.setBackgroundResource(R.color.corTxt);
            }
        };
        handler.postDelayed(restauraCor, 300);
    }

    @SuppressLint("ClickableViewAccessibility")
    private void setTxtMaterialOnTouch() {
        txtMaterial.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                int x = (int)event.getX();
                int y = (int)event.getY();

                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    txtLogTouch.setText("touch down");
                    setyIni(y);
                }
                else if (event.getAction() == MotionEvent.ACTION_MOVE) {
                    txtLogTouch.setText("movendo: (" + x + ", " + y + ")");
                    if (getyIni() > 0 && getyIni()-y > 100) {
                        txtAno.setText("");
                        txtMaterial.setText("");
                    }
                }
                else if (event.getAction() == MotionEvent.ACTION_UP) {
                    txtLogTouch.setText("touch up");
                    setyIni(-1);
                }
                return false;
            }
        });
    }
}
```

Classe MaterialEscolar.java

```
public class MaterialEscolar {
    private String ano;
    private String material;

    public MaterialEscolar(String ano, String material) {
        this.ano = ano;
        this.material = material;
    }

    public String getAno() {
        return ano;
    }

    public String getMaterial() {
        return material;
    }

    public String toString() {
        return getAno();
    }


    public static final int qtdeMe = 9;
    public static final MaterialEscolar[] me  = {
            new MaterialEscolar("1º Ano", "1 cx lápis de cor\n5 lápis pretos HB\n1 borracha\n1 apontador de lápis\n1
tubo de cola\n6 folhas de papel de seda coloridas"),
            new MaterialEscolar("2º Ano", "1 cx lápis de cor\n5 lápis pretos HB\n25 folhas de papel sulfite\n1
borracha\n1 apontador de lápis\n1 tubo de cola\n1 tesoura infantil"),
            new MaterialEscolar("3º Ano", "2 canetas azuis\n1 cx lápis de cor\n5 lápis pretos HB\n1 borracha\n1
apontador de lápis\n1 tubo de cola\n1 tesoura infantil"),
            new MaterialEscolar("4º Ano", "2 canetas azuis\n1 cx lápis de cor\n5 lápis pretos HB\n1 borracha\n1
apontador de lápis\n100 folhas de papel sulfite\n1 régua de 30 cm"),
            new MaterialEscolar("5º Ano", "2 canetas azuis\n1 caneta preta\n10 lápis pretos HB\n1 borracha\n1 apontador
de lápis\n100 folhas de papel sulfite\n1 régua de 30 cm\n1 transferidor"),
            new MaterialEscolar("6º Ano", "200 folhas de sulfite\n2 canetas azuis\n5 lápis pretos\n1 dicionário de
português\n1 régua"),
            new MaterialEscolar("7º Ano", "150 folhas de sulfite\n4 canetas azuis\n1 caneta preta\n1 esquadro 45º\n1
dicionário de inglês"),
            new MaterialEscolar("8º Ano", "150 folhas de sulfite\n4 canetas azuis\n1 caneta preta\n1 esquadro 60º\n1
esquadro 45º"),
            new MaterialEscolar("9º Ano", "150 folhas de sulfite\n4 canetas azuis\n1 caneta preta\n1 transferidor\n1
compasso\n")
    };

}
```

Notas Técnicas

Listeners

Em linhas gerais pode-se dizer que um Listener é a forma disponível no ambiente Android para que exista a interação entre o usuário e o aplicativo. Ainda não tratamos formalmente desse assunto, mas já estamos usando Listeners desde a primeira aula. Fizemos isso toda vez que implementamos o OnClick em botão ou TextView.

Todo Listener é a implementação de alguma Interface definida na classe View. Aqui é necessário que vocês, alunos, compreendam bem o que é uma Interface no contexto de Programação Orientada a Objetos na linguagem Java. Para isso eu recomendo a leitura do capítulo 11 da apostila de Java da Caelum recomendada no início do semestre e que pode ser encontrada neste link: https://www.caelum.com.br/apostila-java-orientacao-objetos

Muito se usa a ideia de que uma interface em Java é como um "contrato". No mundo fora da programação de computadores um contrato é um documento que você pode optar por aderir ou não, mas uma vez que a opção de adesão seja feita, passa a ser obrigatório o cumprimento de todas suas cláusulas. Essa é a ideia de interface em programação orientada a objetos.

A linguagem Java, e as outras linguagens orientadas a objeto, preveem a existência de interfaces. Ao escrever uma classe a partir de uma superclasse existente e que contenha uma Interface, o programador pode optar por implementar a interface ou não. Se não implementar, tudo bem, sua classe funcionará sem isso (isso é algo diferente dos métodos abstratos, que são obrigatórios sempre). E, se optar, implementar deve respeitar o que está previsto.

Vejam o exemplo contido neste link: https://www.devmedia.com.br/entendendo-interfaces-em-java/25502

Assim, um Listener no ambiente Android é implementação de uma interface que contém um ou mais métodos que deverão ser escritos pelo programador para implementar a lógica esperada do aplicativo. Existem Listeners para todo tipo de eventos de entrada, tais como: OnClick(), OnLongClick(), onTouch(), onFocusChange(), entre outros. Uma lista completa de opções e suas explicações podem ser encontradas no link: https://developer.android.com/guide/topics/ui/ui-events.html

Agora precisamos descrever como configurar esses Listeners. Os quatro primeiros serão a implementação do evento OnClick(). O quinto listener será um evento OnLongClick() e o sexto listener será um OnTouch().

Implementação da 1ª opção de OnClick()

Este é o caso em que o método é criado no código Java e configurado no atributo onClick do botão no arquivo XML. É a forma que já estamos usando desde o primeiro aplicativo deste curso.

As próximas três opções terão a configuração do Listener através do método

Implementação da 2ª opção de OnClick() – com a declaração explícita de um objeto

Para configurar um listener é necessário usar um método *setter*, de modo que para configurar um evento OnClick usaremos o método setOnClickListener(). O exemplo abaixo mostra a criação de um objeto da classe View.OnClickListener denominado "`listener`". Em seguida, é feita a codificação da função call-back que é a responsável por executar o código desejado.

Por fim, o método setOnClickListener() é acionado passando-se o objeto criado e dessa forma configurando o componente para responder ao evento OnClick().

```
private void setBtnOpc2OnClick() {
    View.OnClickListener listener;            ← definição do objeto listener da classe View.OnClickListener
    listener = new View.OnClickListener() {   ← instanciação do objeto
        @Override
        public void onClick(View view) {                                    ← onClick é a função callback
            ...aqui é colocado o código que executa a tarefa necessária...
        }
    };
    btnOpc2.setOnClickListener(listener);     ← uso do setter para configurar o listener
}
```

Implementação da 3ª opção de OnClick() – com o uso de objeto anônimo

Nesta terceira implementação o parâmetro passado ao método setOnClickListener() é um objeto criado diretamente na chamada. A isso se dá o nome de objeto anônimo. Veja no trecho de código abaixo que o parâmetro passado ao *setter* inicia-se com "new ...", ou seja, está sendo criado um objeto ali. Porém, a este objeto não foi dado um nome. E nem é necessário porque o

interpretador Java não precisa de um nome de objeto para lidar com os objetos criados em memória. Para o interpretador Java basta que o objeto exista e isso é conseguido usando-se o comando *new*.

```
private void setBtnOpc3OnClick() {
    btnOpc3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ...aqui é colocado o código que executa a tarefa necessária...
        }
    });
}
```

Implementação da 4ª opção de OnClick() – com o uso de objeto anônimo e função lambda

A chamada função lambda é uma forma mais compacta da forma mostrada na 3ª opção. Compare os trechos de código acima e abaixo para perceber a diferença. Eles fazem exatamente a mesma coisa. Esta é uma forma nova introduzida no Java 8 de modo que para ser usada é preciso que a compilação do programa seja feita com essa versão do compilador.

Para saber mais sobre as funções lambda acesse: https://developer.android.com/studio/write/java8-support

```
private void setBtnOpc4OnClick() {
    btnOpc4.setOnClickListener((View view) -> {
            ...aqui é colocado o código que executa a tarefa necessária...
    });
}
```

Implementação do OnLongClick()

O quinto Listener deste aplicativo é um toque longo no componente TextView que é usado para exibir a lista de material escolar. Neste evento será implementada a cópia do texto para a área de transferência, seguido um efeito de mudança de cor que emulará uma "piscada" e por fim a exibição de um Toast.

Este Listener tem implementação que recebe um parâmetro da classe View e retorna um boolean. Esse retorno deve ser ***true*** se o listener tratou o evento e ***false*** se não tratou.

```
private void setTxtMaterialOnLongClick() {
    txtMaterial.setOnLongClickListener(new View.OnLongClickListener() {
            ...aqui é colocado o código que executa a tarefa necessária...
    });
}
```

Implementação do OnTouch()

O último Listener deste aplicativo será disparado quando o TextView for tocado e movido enquanto o toque permanece. Se for movida uma distância de no mínimo 100 pixels para cima. Para isso guardamos a posição do toque inicial e à medida que o dedo é movido é feito o cálculo da distância movida. Se for atingida a marca de 100 pixels o código limpa o componente.

Este Listener tem implementação que recebe um primeiro parâmetro da classe View e um segundo parâmetro que é da classe MotionMovement, que contém parâmetros relativos ao movimento executado pelo dedo do usuário. Ele retorna um boolean, que deve ser ***true*** se o listener tratou o evento e ***false*** se não tratou.

```
private void setTxtMaterialOnTouch() {
    txtMaterial.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent event) {
            ...aqui é colocado o código que executa a tarefa necessária...
    });
}
```

Adicional – Tela Splash

Telas Splash (Splashscreen) costumam ser usadas na abertura dos aplicativos. Neste app vamos criar uma, a título de tarefa Adicional. Acrescente uma nova Activity ao app e ajuste o laytout e o código Java conforme abaixo.

Layout: activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:background="@color/corTxt"
    tools:context=".SplashActivity">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:scaleType="centerInside"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/listeners" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Código: SplashActivity.java

```
public class SplashActivity extends AppCompatActivity {

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

    private void configuraAvanco() {
        Handler handler = new Handler(Looper.getMainLooper());
        Runnable avanca = new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(SplashActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        };
        handler.postDelayed(avanca, 8000);
    }
}
```

Neste código é criado um objeto Runnable e agendado para executar após alguns segundos. A tarefa deste Runnable é iniciar a MainActivity e e fechar a SplashActivity.

Para que a SplashActivity seja o ponto de entrada do aplicativo é preciso fazer uma alteração no arquivo manifesto AndroidManifest.xml

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ilp506.listeners">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Listeners">
        <activity android:name=".SplashActivity"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity">
        </activity>
    </application>

</manifest>
```

No manifesto o bloco <intent-filter> precisa ser transferido da MainActivity para a SplashActivity.

Além disso, foi alterada a cláusula android:theme para a o valor "@style/Theme.AppCompat.Light.NoActionBar" para que a SplashActivity não exiba a barra de títulos.