Ficha do Aplicativo

App03 – Aplicativo Brincando com Inteiros
Resumo Neste App03 poderemos gerar e exibir um número inteiro, verificar se ele é par ou ímpar, se é primo ou não e apresentar os divisores caso seja não primo não primo. Neste aplicativo vamos trabalhar com componentes de layout do tipo containers e views. No caso de containers usaremos o LinearLayout, tanto vertical como horizontal. No caso dos componentes usaremos TextView, EditText e Button, já vistos nos aplicativos anteriores e acrescentaremos o Divider.
Objetivos de Aprendizagem 1. Trabalhar a criação de layouts responsivos. 2. Usar componentes do tipo Container, no caso será usado o LinearLayout (vertical e horizontal) 3. Usar componentes do tipo View, no caso serão usados TextView e Button, já vistos no App01 da aula anterior e acrescentaremos o EditText e o Divider. 4. Criar layouts diferenciados para orientação Retrato (em pé) e Paisagem (deitado) do dispositivo. 5. Possibilitar a rolagem de texto em uma TextView. 6. Conhecer e começar a implementar métodos do Ciclo de Vida das Activities. 7. Conhecer e usar o tratamento de exceções com o comando try – catch da Linguagem Java. 8. Usar a classe Toast, agora com layout personalizado, para exibição de mensagens. 9. Conhecer os conceitos iniciais sobre ciclo de vida de uma Activity.
Dinâmica do Aplicativo Este aplicativo conterá 5 botões que programados para as seguintes tarefas **Gerar Nº**: Gera um número inteiro até 100000 e o coloca em um componente TextView **Transferir**: Transfere o número gerado para um componente EditText **Paridade?**: Implementa a verificação se um número é par ou ímpar e coloca o resultado em um TextView **Primo?**: Implementa a verificação se o número é primo ou não e coloca o resultado em um TextView **Limpar Tudo**: Limpa todos os componentes preenchidos pelos botões anteriores.

Lista de Activities do Aplicativo

Nome	Layout (2 layouts associados ao mesmo código Java)
MainActivity.java	layout\activity_main.xml e land\activity_main.xml

Resources

strings.xml

```
<resources>
    <string name="app_name">Brincando com Inteiros</string>
    <string name="lblgeracao">Geração do Número</string>
    <string name="lbldigitacao">Digitação do Número</string>
    <string name="edttransferehint">Digite um nº inteiro</string>
    <string name="lblpropriedades">Propriedades do Número</string>
    <string name="lbldivisores">Divisores do Número</string>
    <string name="btngerar">Gerar Nº</string>
    <string name="btntransferir">Trasnferir</string>
    <string name="btnparidade">Paridade?</string>
    <string name="btnprimo">Primo?</string>
    <string name="btnlimpar">Limpar Tudo</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="corAtividade">#C9CE9E</color>
    <color name="corFundo">#CDF8FFD0</color>
    <color name="cordivisores">#C8FAF6E8</color>
</resources>
```

Drawable deste aplicativo

Nome do arquivo	Dimensões	Tipo	Imagem
ic_alerta.xml	24 x 24	XML – Vector	
toast_shape.xml	---	XML de definição de Shape	

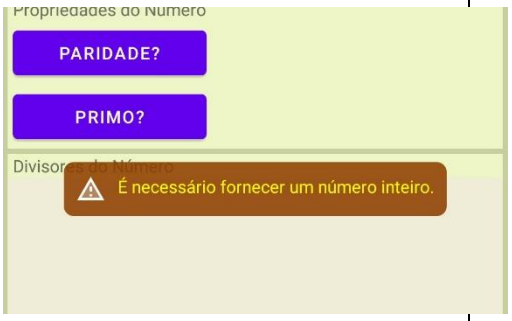
Layout do Toast personalizado – arquivo toast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/raiz_layout_toast"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:layout_margin="20dp"
    android:padding="10dp"
    android:background="@drawable/toast_shape">

    <ImageView android:id="@+id/imagem_toast"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginRight="10dp" />

    <TextView
        android:id="@+id/texto_toast"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textColor="#FFFF00"
        android:textSize="14sp" />

</LinearLayout>
```



Os arquivos dois drawables e o layout personalizado do Toast estão disponíveis como recursos da aula e podem ser baixados da minha página. Após baixar o arquivo compactado, descompacte seu conteúdo em alguma pasta do seu computador. São 3 conteúdos, 2 drawables e 1 layout:

- copie os dois drawables – ic_alerta.xml e toast_shape.xml – para a pasta de drawables do seu projeto Android Studio. Essa pasta estará no seguinte caminho **`{nome do projeto}\app\src\main\res\drawable`**.

- Copie o layout – toast.xml – para a pasta de layouts do seu projeto Android Studio. Essa pasta estará no seguinte caminho: **`{nome do projeto}\app\src\main\res\layout`**.

Detalhamento das Activities

Layout na posição retrato: layout\activity_main.xml


```
<?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"
    android:background="@color/corAtividade"
    android:orientation="vertical"
    android:padding="4dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:paddingStart="4dp"
        android:paddingEnd="4dp"
        android:text="@string/lblgeracao"
        android:textSize="14sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:orientation="horizontal"
        android:padding="2dp"
        android:paddingStart="4dp"
        android:paddingEnd="4dp">

        <Button
            android:id="@+id/btnGerar"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:onClick="btnGerarOnClick"
            android:text="@string/btngerar" />

        <TextView
            android:id="@+id/txtGerado"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="3"
            android:gravity="center"
            android:textSize="18sp" />
    </LinearLayout>

    <View
        android:id="@+id/divider1"
        android:layout_width="match_parent"
        android:layout_height="4dp"
        android:background="@color/corAtividade" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:paddingStart="4dp"
        android:paddingEnd="4dp"
        android:text="@string/lbldigitacao"
        android:textSize="14sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:orientation="horizontal"
        android:padding="2dp"
        android:paddingStart="4dp"
        android:paddingEnd="4dp">
```

```
        <Button
            android:id="@+id/btnTransferir"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:onClick="btnTransferirOnClick"
            android:text="@string/btntransferir" />

        <EditText
            android:id="@+id/edtTransfere"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="3"
            android:autofillHints="número"
            android:hint="@string/edttransferehint"
            android:inputType="number"
            android:textAlignment="center"
            android:textSize="18sp" />
    </LinearLayout>

    <View
        android:id="@+id/divider2"
        android:layout_width="match_parent"
        android:layout_height="4dp"
        android:background="@color/corAtividade" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:paddingStart="4dp"
        android:paddingEnd="4dp"
        android:text="@string/lblpropriedades"
        android:textSize="14sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:orientation="horizontal"
        android:padding="2dp"
        android:paddingStart="4dp"
        android:paddingEnd="4dp">

        <Button
            android:id="@+id/btnParidade"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:onClick="btnParidadeOnClick"
            android:text="@string/btnparidade" />

        <TextView
            android:id="@+id/txtParidade"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="3"
            android:gravity="center"
            android:text=""
            android:textSize="18sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:orientation="horizontal"
        android:padding="2dp"
        android:paddingStart="4dp"
        android:paddingEnd="4dp">

        <Button
            android:id="@+id/btnPrimo"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:onClick="btnPrimoOnClick"
            android:text="@string/btnprimo" />
```

```
        <TextView
            android:id="@+id/txtPrimo"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="3"
            android:gravity="center"
            android:text=""
            android:textSize="18sp" />
    </LinearLayout>

    <View
        android:id="@+id/divider3"
        android:layout_width="match_parent"
        android:layout_height="4dp"
        android:background="@color/corAtividade" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/corFundo"
        android:paddingStart="4dp"
        android:paddingEnd="4dp"
        android:text="@string/lbldivisores"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/txtDivisores"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/cordivisores"
        android:onClick="txtDivisoresOnClick"
        android:paddingStart="10dp"
        android:scrollbars="vertical"
        android:text=""
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnLimpar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="btnLimpaTudo"
        android:text="@string/btnlimpar" />

</LinearLayout>
```

Layout na posição paisagem: layout-land\activity_main.xml

```
<?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"
    android:background="@color/corAtividade"
    android:orientation="horizontal"
    android:padding="4dp"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginEnd="2dp"
        android:layout_weight="1"
        android:background="@color/corFundo"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingStart="4dp"
            android:text="@string/lblgeracao"
            android:textSize="14sp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:padding="2dp">

            <Button
                android:id="@+id/btnGerar"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:onClick="btnGerarOnClick"
                android:text="@string/btngerar" />

            <TextView
                android:id="@+id/txtGerado"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="3"
                android:gravity="center"
                android:text=""
                android:textSize="18sp" />

        </LinearLayout>

        <View
            android:id="@+id/divider1"
            android:layout_width="match_parent"
            android:layout_height="10dp"
            android:background="@color/corAtividade" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingStart="4dp"
            android:text="@string/lbldigitacao"
            android:textSize="14sp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:padding="2dp">

            <Button
                android:id="@+id/btnTransferir"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:onClick="btnTransferirOnClick"
```

```
                android:text="@string/btntransferir" />

            <EditText
                android:id="@+id/edtTransfere"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="3"
                android:autofillHints="número"
                android:inputType="number"
                android:textAlignment="center"
                android:textSize="18sp" />
        </LinearLayout>

        <View
            android:id="@+id/divider2"
            android:layout_width="match_parent"
            android:layout_height="10dp"
            android:background="@color/corAtividade" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingStart="4dp"
            android:text="@string/lblpropriedades"
            android:textSize="14sp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:padding="2dp">

            <Button
                android:id="@+id/btnParidade"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:onClick="btnParidadeOnClick"
                android:text="@string/btnparidade" />

            <TextView
                android:id="@+id/txtParidade"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="3"
                android:gravity="center"
                android:text=""
                android:textSize="18sp" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:padding="2dp">

            <Button
                android:id="@+id/btnPrimo"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:onClick="btnPrimoOnClick"
                android:text="@string/btnprimo" />

            <TextView
                android:id="@+id/txtPrimo"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="3"
                android:gravity="center"
                android:text=""
                android:textSize="18sp" />
        </LinearLayout>

    </LinearLayout>

    <LinearLayout
        android:layout_width="0dp"
```

```
        android:layout_height="match_parent"
        android:layout_marginStart="2dp"
        android:layout_weight="1"
        android:background="@color/corFundo"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingStart="4dp"
            android:text="@string/lbldivisores"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/txtDivisores"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="5"
            android:background="@color/cordivisores"
            android:onClick="txtDivisoresOnClick"
            android:paddingStart="10dp"
            android:scrollbars="vertical"
            android:text=""
            android:textSize="18sp" />

        <Button
            android:id="@+id/btnLimpar"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:onClick="btnLimpaTudo"
            android:text="@string/btnlimpar"
            android:textSize="18sp" />

    </LinearLayout>


</LinearLayout>
```

Código: MainActivity.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 TextView txtGerado;
    private EditText edtTransfere;
    private TextView txtParidade;
    private TextView txtPrimo;
    private TextView txtDivisores;

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

        txtGerado = findViewById(R.id.txtGerado);
        edtTransfere = findViewById(R.id.edtTransfere);
        txtParidade = findViewById(R.id.txtParidade);
        txtPrimo = findViewById(R.id.txtPrimo);
        txtDivisores = findViewById(R.id.txtDivisores);
    }

    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        super.onSaveInstanceState(savedInstanceState);
        savedInstanceState.putString("gerado", txtGerado.getText().toString());
        savedInstanceState.putString("paridade", txtParidade.getText().toString());
        savedInstanceState.putString("primo", txtPrimo.getText().toString());
        savedInstanceState.putString("divisores", txtDivisores.getText().toString());
    }

    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        String s;
        s = savedInstanceState.getString("gerado");
        txtGerado.setText(s);
        s = savedInstanceState.getString("paridade");
        txtParidade.setText(s);
        s = savedInstanceState.getString("primo");
        txtPrimo.setText(s);
        s = savedInstanceState.getString("divisores");
        txtDivisores.setText(s);
    }

    public void btnGerarOnClick(View view) {
        Random gerador = new Random();
        int valor = gerador.nextInt(100000);
        String s = String.format("%d", valor);
        txtGerado.setText(s);
    }

    public void btnTransferirOnClick(View view) {
        String s = txtGerado.getText().toString();
        if (s != "") {
            edtTransfere.setText(txtGerado.getText().toString());
        }
    }

    public void btnParidadeOnClick(View view) {
        String s = edtTransfere.getText().toString();
        try {
            int valor = Integer.parseInt(s);
            if (valor % 2 == 0)
                txtParidade.setText(String.format("%d é Par", valor));
            else
                txtParidade.setText(String.format("%d é Ímpar", valor));
        }
        catch (Exception e) {
            //Toast.makeText(this, "É necessário fornecer um número inteiro.", Toast.LENGTH_SHORT).show();
            exibeToast("É necessário fornecer um número inteiro.");
        }
    }

    public void btnPrimoOnClick(View view) {
        String s = edtTransfere.getText().toString();
        try {
```

```
            int valor = Integer.parseInt(s);
            if (ePrimo(valor))
                txtPrimo.setText(String.format("%d é Primo", valor));
            else
                txtPrimo.setText(String.format("%d não é Primo", valor));
        }
        catch (Exception e) {
            //Toast.makeText(this, "É necessário fornecer um número inteiro.", Toast.LENGTH_SHORT).show();
            exibeToast("É necessário fornecer um número inteiro.");
        }
    }

    private boolean ePrimo (int pN) {
        int i;
        int r = 1;
        double raiz;
        if (pN == 2)
            return true;
        else if (pN % 2 == 0)
            return false;
        else {
            raiz = Math.sqrt(pN);
            i = 3;
            while (i <= raiz && r != 0) {
                r = pN % i;
                i += 2;
            }
            return r != 0;
        }
    }

    public void txtDivisoresOnClick(View view) {
        String s = edtTransfere.getText().toString();
        try {
            int valor = Integer.parseInt(s);
            s = produzDivisores(valor);
            txtDivisores.setText(s);
            txtDivisores.setMovementMethod(new ScrollingMovementMethod());
            txtDivisores.scrollTo(0, 0);
        }
        catch (Exception e) {
            //Toast.makeText(this, "É necessário fornecer um número inteiro.", Toast.LENGTH_SHORT).show();
            exibeToast("É necessário fornecer um número inteiro.");
        }
    }

    private String produzDivisores(int valor) {
        int i;
        double fim;
        String s = "";
        fim = valor / 2;
        i = 2;
        while (i <= fim) {
            if (valor % i == 0)
                s = s + i + "\n";
            i++;
        }
        return s;
    }

    public void btnLimpaTudo(View view) {
        TextView v;
        v = findViewById(R.id.txtGerado);
        v.setText("");
        v = findViewById(R.id.txtParidade);
        v.setText("");
        v = findViewById(R.id.txtPrimo);
        v.setText("");
        v = findViewById(R.id.txtDivisores);
        v.setText("");
        v = findViewById(R.id.edtTransfere);
        v.setText("");
    }

    private void exibeToast(String txt) {
        LayoutInflater inflater = getLayoutInflater();
        View toastLayout = inflater.inflate(R.layout.toast, (ViewGroup)findViewById(R.id.raiz_layout_toast));
        ImageView imagem_toast = toastLayout.findViewById(R.id.imagem_toast);
        imagem_toast.setImageResource(R.drawable.ic_alerta);
        TextView texto_toast = toastLayout.findViewById(R.id.texto_toast);
        texto_toast.setText(txt);
```

```
        Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
        toast.setDuration(Toast.LENGTH_SHORT);
        toast.setView(toastLayout);
        toast.show();
    }
}
```

Notas Técnicas

Layout Responsivo

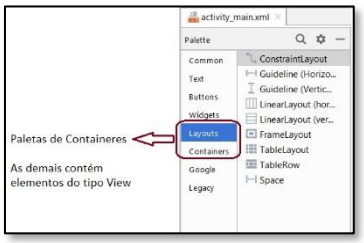
Existem dispositivos físicos com uma gama muito variados em termos de formatos e tamanhos. Deste modo um aplicativo para esses dispositivos precisa ter seu layout visual muito flexível. Um layout que seja bem visualizado, ou seja, tenha uma boa aparência, nos diversos dispositivos existentes é dito “responsivo”.

Desta forma, o desenvolvedor não pode criar seus layouts com proporções, posições de componentes e dimensões rígidas, pressupondo um formato e tamanho específico de aparelho. Um bom layout precisa ter a capacidade de responder adequadamente e de forma eficiente a variações de tamanho, densidade e orientação da tela. Daí vem o nome “Layout Responsivo”.

Até por questões comerciais, convém ao desenvolvedor que seu aplicativo possa ser executado na mais ampla gama possível de aparelhos em uso pelo seu público alvo.

Algumas técnicas e cuidados devem ser empregadas para atingir este objetivo:

- Usar o componente ConstraintLayout como elemento básico de layout da tela (interface do usuário). Essa é uma recomendação feita, porém como pode ser visto neste aplicativo esse componente de layout não foi utilizado, sendo substituído pelo LinearLayout;
- Na posição e dimensões de componentes usar a unidade Density Pixel (DP) e nunca usar pixels físicos;
- Nos tamanhos de fonte (texto) usar a unidade Scale Pixel (SP), que de forma automática faz a combinação de Density Pixels com o tamanho de fonte da preferência do usuário. Este último pode ser configurado no pelo dono do aparelho no menu de configurações do Android;
- Usar dimensões parametrizadas (wrap_content, match_constraint, match_parent) capazes de se adaptar a diferentes tamanhos físicos de tela, redimensionando o layout;
- Combinar de modo conveniente os componentes dos tipos Container e View.



Containers são componentes capazes de agrupar em seu espaço interno outros componentes, dos dois tipos. Os componentes desse tipo ficam nas paletas Layouts e Containers.

Views são componentes com os quais o usuário interage diretamente. Os componentes desse tipo ficam nas outras paletas.

Para saber mais consulte o link: https://developer.android.com/training/multiscreen/screensizes?hl=pt-br

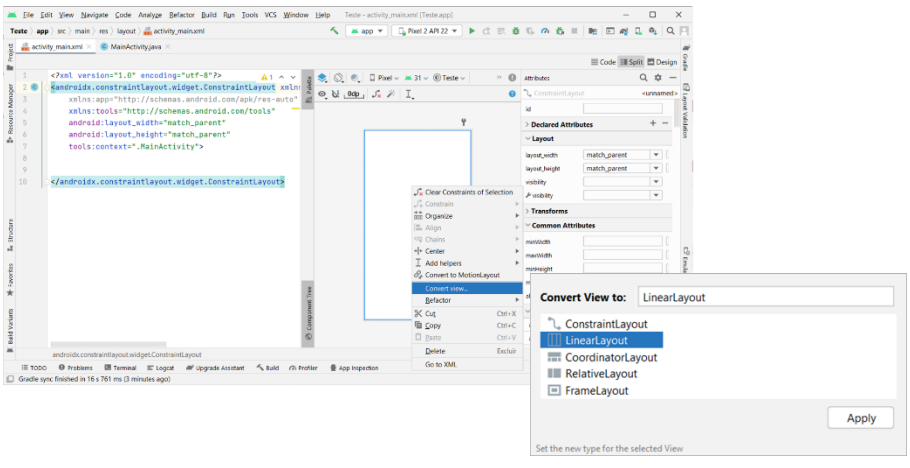
Substituição do ConstraintLayout padrão por outro Layout

Para trocar o layout base da sua activity proceda da seguinte forma: após criar uma nova activity ou um novo projeto clique com o botão direito sobre a área branca do layout.

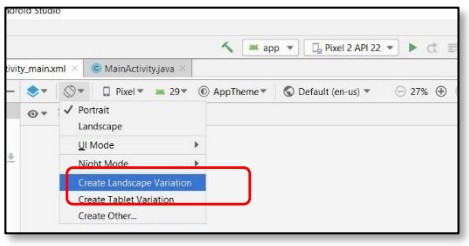
No menu que será aberto selecione a opção "Convert view...".

Será aberta uma janela onde será possível escolher o tipo de layout que deverá substituir o ConstraintLayout.

Nas duas activities deste aplicativo vamos utilizar o componente do tipo LinearLayout. Faça a troca e verifique que o arquivo do seu XML terá sido alterado de modo que o elemento raiz agora será o layout selecionado.



Uma Activity – Dois Layouts



Você já percebeu que neste aplicativo são usados dois layouts distintos: um para o modo *Portrait* (ou retrato, que condiz com o aparelho em pé = altura maior que a largura) e outro para o modo *Landscape* (ou paisagem, que condiz com o aparelho deitado = altura menor que a largura). Essa é uma situação corriqueira no desenvolvimento de aplicativos móveis e o Android Studio gerencia todos os detalhes para nós.

Por padrão, quando uma nova activity é criada o layout gerado pelo Android Studio é o modo retrato. Para ter os dois layouts, precisamos criar o layout alternativo no modo paisagem usando o comando "Create Landscape Variation" mostrado na imagem ao lado e desenhá-lo.

O XML do layout original, que é o modo retrato, ficará armazenado na pasta: `...\app\src\main\res\layout`

Enquanto a variação do layout para o modo paisagem ficará em: `...\app\src\main\res\layout-land`

Assim, após iniciar seu novo aplicativo basta criar a versão landscape do layout e desenhar os dois layouts. Faça isso e teste a aplicação. Ao girar o aparelho o layout automaticamente é alterado para a versão apropriada.

Sugestão: se você criar o layout landscape depois que o layout portrait estiver pronto, com todos os seus componentes, o Android Studio irá gerar o novo layout contendo todos os componentes do layout original.

Implementação da verificação da paridade

Vamos descrever o código implementado no método btnParidadeOnClick, que será acionado quando o for clicado o botão correspondente (reproduzimos o código completo a seguir). Nesse código será verificado se o número inteiro presente no EditText edtTransfere é par ou ímpar. Porém, caixas de texto contém apenas texto (String). O desenvolvedor precisa estar consciente disso.

Não sendo um conteúdo numérico, propriamente dito, é preciso fazer a conversão desse conteúdo de String para int. Usando a função findViewById fazemos o vínculo com os componentes View do layout. Usaremos os componentes edtTransfere que fornecerá o número a ser convertido e o txtParidade que receberá o resultado.

Para compreender o que está sendo feito nesse código veja os comentários em cada linha.

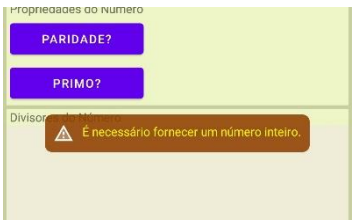
```
public void btnParidadeOnClick(View view) {
    TextView txtParidade = findViewById(R.id.txtParidade);
    EditText edtTransfere = findViewById(R.id.edtTransfere);
    String s = edtTransfere.getText().toString();   // carrega o objeto s com o texto contido em edtTransfere
    try {
        int valor = Integer.parseInt(s);   // converte o s para inteiro e armazena na variável valor do tipo int
        if (valor % 2 == 0)                // verifica se valor é divisível por zero
            txtParidade.setText(String.format("%d é Par", valor));   // carrega o txtParidade com o resultado
        else
            txtParidade.setText(String.format("%d é Ímpar", valor)); // carrega o txtParidade com o resultado
    }
    catch (Exception e) {
        exibeToast("É necessário fornecer um número inteiro.");
    }
}
```

Este código é sujeito a ocorrência de um erro que comprometerá o funcionamento do aplicativo, causando seu fechamento abrupto. Caso o conteúdo do componente EditText edtTransfere seja não numérico ocorrerá um erro no momento da conversão para número inteiro usando o método parseInt.

Para que o aplicativo não seja cancelado é preciso proteger esse trecho de código, seja condicionando-o a um if, seja usando o mecanismo de proteção com tratamento de exceções que é implementado com o comando try – catch. Neste comando, todo o trecho de código contido na parte try do comando estará protegido. No caso de ocorrência de erro, a execução será desviada para o catch, executando qualquer código que seja escrito ali.

Neste caso disparamos uma notificação Toast descrita a seguir.

Notificação Toast Personalizada



Conforme já foi visto antes, uma notificação Toast constitui um modo simples de exibir uma mensagem para o usuário, abrindo uma pequena janela pop-up. Esse pop-up só ocupa a quantidade de espaço necessária para a mensagem, e a atividade atual continua visível e interativa. Notificações Toast desaparecem automaticamente após um tempo limite.

No aplicativo anterior já usamos o Toast na forma padrão. Agora queremos personalizar sua aparência. A figura mostra como ficará o nosso Toast personalizado.

A personalização inclui os seguintes elementos:

- Há uma imagem à esquerda da mensagem;
- A área ocupada pelo Toast será preenchida por um retângulo de cantos arredondados colorido com uma cor que possui transparência, permitindo a visualização de parte do que está ao fundo;
- A posição vertical do Toast na tela será ligeiramente abaixo do centro da tela

Para atingir esses objetivos precisamos:

- Colocar o arquivo ic_alerta.xml na pasta `...\app\src\main\res\drawable`. Na aula eu vou mostrar como criá-lo usando o próprio Android Studio;
- Colocar o arquivo toast_shape na pasta `...\app\src\main\res\drawable`. Na aula eu vou mostrar como criá-lo usando o próprio Android Studio;
- Colocar o arquivo toast_xml na pasta `...\app\src\main\res\layout`. Na aula eu vou mostrar como criá-lo usando o próprio Android Studio;
- Implementar o código abaixo que é responsável por inflar a view e exibi-la;

```
private void exibeToast(String txt) {
    LayoutInflater inflater = getLayoutInflater();
    View toastLayout = inflater.inflate(R.layout.toast, (ViewGroup) findViewById(R.id.raiz_layout_toast));
    ImageView imagem_toast = toastLayout.findViewById(R.id.imagem_toast);
    imagem_toast.setImageResource(R.drawable.ic_alerta);
    TextView texto_toast = toastLayout.findViewById(R.id.texto_toast);
    texto_toast.setText(txt);

    Toast toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
    toast.setDuration(Toast.LENGTH_SHORT);
    toast.setView(toastLayout);
    toast.show();
}
```

Inflar uma View em Android é um conceito importante.

Pense da seguinte forma: como é que um arquivo xml se torna um conjunto de objetos das classes certas para serem desenhados na tela e terem o comportamento esperado? A resposta é: "inflando a view".

Talvez ainda não tenha ajudado muito, mas é isso que vamos fazer.

Na prática esse termo inflar se refere ao recurso comum de ler um arquivo xml, interpretá-lo e construir em memória os objetos necessários. Essa tarefa é tão comum em Android, que a API fornece as classes necessárias para a tarefa. Então basta utilizá-las:

`LayoutInflater inflater = getLayoutInflater();`	Precisamos de um objeto da classe LayoutInflater e vamos construí-lo usando o método getLayouInflater()

`View toastLayout = inflater.inflate(`
`  R.layout.toast,`
`  (ViewGroup) findViewById(R.id.raiz_layout_toast)`
`);`
	Usando o inflater gerado na linha anterior vamos construir a nossa view que será responsável pela exibição do Toast. Usa-se o método inflate() com dois parâmetros:
	• O resource do layout
	• O id do componente raiz que pertence ao layout

`ImageView imagem_toast =`
`  toastLayout.findViewById(R.id.imagem_toast);`
 	
`imagem_toast.setImageResource(R.drawable.ic_alerta);`
	Aqui é feita a carga da imagem que se quer exibir

`TextView texto_toast =`
`toastLayout.findViewById(R.id.texto_toast);`
`texto_toast.setText(txt);`
	Aqui é feita a carga do texto que se quer exibir

`Toast toast = new Toast(getApplicationContext());`
`toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);`
`toast.setDuration(Toast.LENGTH_SHORT);`
`toast.setView(toastLayout);`
`toast.show();`
	Por fim, com o layout já inflado cria-se o objeto da classe Toast, e antes de exibi-la com o show pode-se especificar as suas particularidades. Neste caso são três:
	• A gravidade para centralizar na vertical (com um deslocamento de 100dp para baixo)
	• A duração
	• A aparência – aqui usando o toastLayout pronto, produzido nas linhas acima

Veremos mais sobre inflar views em outros aplicativos.

Mais sobre Toast neste link: https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=pt-br

Ciclo de Vida de uma Atividade

O Ciclo de Vida de uma Atividade é um conceito que deve ser bem conhecido pelos desenvolvedores de aplicativos para a plataforma Android. É preciso conhecer a maneira como as atividades são iniciadas e organizadas, a forma como as atividades interagem com o sistema operacional e umas com as outras. E é disso que vamos tratar agora.

A seguir fazemos uma descrição completa do Ciclo de Vida, embora neste aplicativo não usaremos todos os métodos envolvidos.

Uma atividade (Activity) está associada à uma tela (Layout) onde é desenhada a interface do usuário. Em geral, os aplicativos possuem múltiplas atividades com suas respectivas telas.

Normalmente, escolhe-se uma atividade para ser a principal (MainActivity), que será a primeira tela a aparecer quando o usuário inicia o aplicativo. A partir daí, podem-se iniciar outras atividades para executar ações específicas dentro do aplicativo. Por exemplo, a atividade principal pode mostrar uma lista geral de produtos e, a partir dessa lista, ao tocar em um produto, dispara-se uma segunda atividade para visualizar em tela cheia os detalhes do produto que não caberiam na lista geral. Desta forma existe no Android um mecanismo que permite que uma Atividade inicie outra.

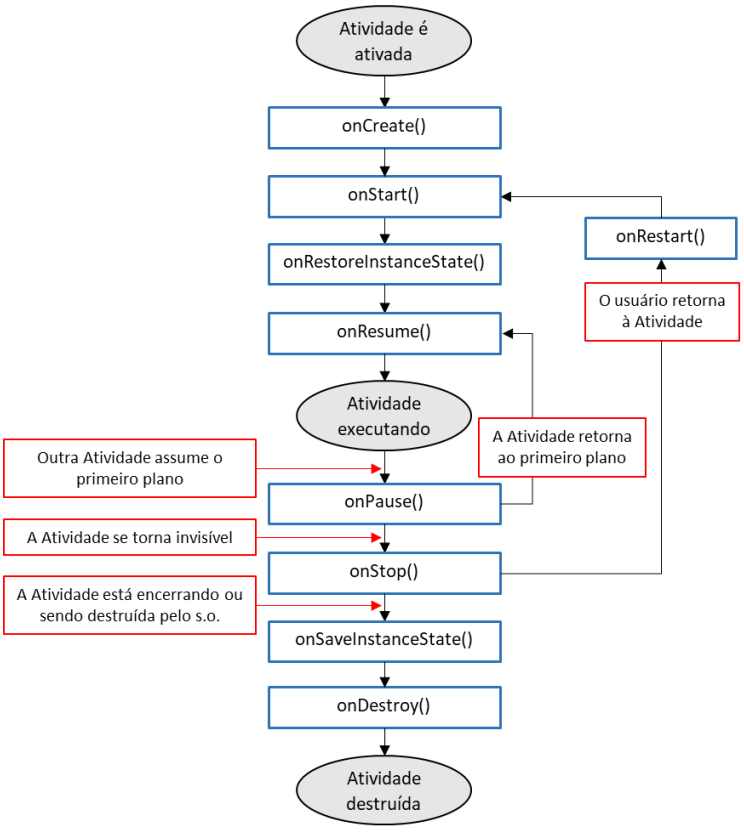
Um Aplicativo é, portanto, composto por uma coleção de atividades, que são planejadas para compor um conjunto coeso e coerente. No entanto, o grau de dependência entre elas é muito baixo, cada atividade pode ser iniciada a qualquer momento (vamos aprender a fazer isso mais adiante) e nada impede que esse início ocorra a partir de uma atividade de outro aplicativo, feito por terceiros.

Isso mesmo, uma atividade do seu aplicativo pode ser iniciada pelo aplicativo feito pelo seu colega e vice-versa. Isso traz inúmeros benefícios, permitindo, por exemplo, que o seu aplicativo inicie um outro para envio de e-mails, abra a galeria, acione a câmera ou quaisquer outras atividades de aplicativos instalados no dispositivo e que aceitem interação externa. Além disso, para tudo isso acontecer não é obrigatório que o seu aplicativo inicie a MainActivity do outro aplicativo. Ele pode iniciar a atividade que for necessária.

Diferentemente dos paradigmas de programação nos quais os aplicativos são iniciados com o método main () ou similar, o sistema Android inicia o código em uma instância de uma atividade – que pertence a um aplicativo, mas que pode ser iniciada diretamente. Para manter a coerência e garantir o correto funcionamento das aplicações, o programador deve conhecer os métodos (funções inseridas nas classes) que são invocados a cada etapa da execução da atividade. Tais métodos podem ou não ser implementados e cada um corresponde a um estágio específico do seu ciclo de vida.

Essa é uma estratégia muito própria do Android e confere grande flexibilidade ao iniciar, rodar e encerrar cada atividade. À medida que o usuário navega pelo aplicativo cada Atividade muda de estado dentro de um conjunto de possibilidades que compõem o que é conhecido como Ciclo de Vida da Atividade.

A figura ao lado mostra os estados em que uma Atividade pode estar (elipses) e os métodos (caixas de borda azul) que podem ser implementados para tratar as transições de um estado para outro. A causa das transições são destacadas nas caixas de borda vermelha.



Seis desses métodos – onCreate(), onStart(), onResume(), onPause(), onStop() e onDestroy() – dizem respeito ao ciclo de vida da atividade propriamente dito. Os outros dois – onSaveInstanceState() e onRestoreInstanceState() – são usados, respectivamente, para salvar e restaurar o estado da atividade.

Descrição dos métodos do ciclo de vida da Atividade

onCreate(Bundle) – é executado quando a Activity é criada. Pode ser usado para instanciar quaisquer objetos a serem usados na Activity, para inicializar variáveis locais e para criar Listeners e associá-los a Views. Esse método fornece um Bundle que permite recuperar um estado salvo anteriormente.

onStart() – é executado quando a atividade está se tornando visível. Lembrando que estar visível não é o mesmo que ter o foco. Uma segunda atividade que ocupe apenas parte da tela ou possua transparência pode ter o foco, ao mesmo tempo que permite que a outra esteja visível.

onResume() – é executado quando a atividade recebe o foco.

onPause() – é executado quando a atividade deixa de estar no primeiro plano, ou seja, perde o foco. Isto ocorre porque outra atividade está iniciando. A atividade seguinte não inicia de fato até que este método termine. Por isso o código do mesmo deve sempre ser rápido.

onStop() – é executado quando a atividade deixa de estar visível. Ocorre em duas situações: ou porque outra atividade a está cobrindo ou porque a atividade está sendo destruída. É sucedido por onRestart() se a atividade se tornar visível outra vez, ou por onDestroy() se a atividade vai ser destruída.

onRestart() – é executado quando a atividade está se tornando novamente visível depois de ter se tornado invisível.

onDestroy() – é executado quando a atividade está sendo destruída, seja porque houve reconfiguração da mesma (o usuário virou o aparelho, p.ex) ou porque o aplicativo foi fechado

onRestoreInstanceState(Bundle) – é executado imediatamente após o método onStart() quando a atividade está sendo reinicializada a partir de um estado que tenha sido previamente salvo usando onSaveInstanceState(). A maior

onSaveInstanceState(Bundle) – é executado imediatamente antes de onDestroy(). Este método oferece um meio de salvar o estado da atividade através de um Bundle que pode ser recuperado se a atividade for reiniciada.

Para saber mais consulte estes links oficiais do Android

Básico sobre Activities: https://developer.android.com/guide/components/activities/intro-activities

Sobre o ciclo de vida de uma Activity: https://developer.android.com/guide/components/activities/activity-lifecycle