Android
Intent, Ciclo de Vida e Bundle

SUMÁRIO

1. Informativo 3
2. Objetivo 4
3. Pré-Requisitos 5
4. Versão 6
   4.1. minSdkVersion 6
   4.2. targetSdkVersion 6
5. Intent 7
   5.1. Intent Explícito 7
   5.2. Intent Implícito 7
6. Bundle 9
7. Ciclo de Vida 10
8. Novo Projeto 11
   8.1. Intent 14
      8.1.1. putExtra() 17
   8.2. Bundle 21
9. Resumo 27
10. Referências 28

1. Informativo

Autor(a): Helena Strada
Data de Criação: jan/2018

2. Objetivo

O objetivo desta apostila é explicar o que são intents, explicar o ciclo de vida de uma aplicação Android e como utilizar o Bundle.

3. Pré-Requisitos

Java

- Orientação a Objetos;
- APIs;
- Bibliotecas.

Android

- Estrutura do Projeto;
- XML;
- Activity;
- Layout.

4. Versão

Android Studio 3.0.1

4.1. minSdkVersion

15

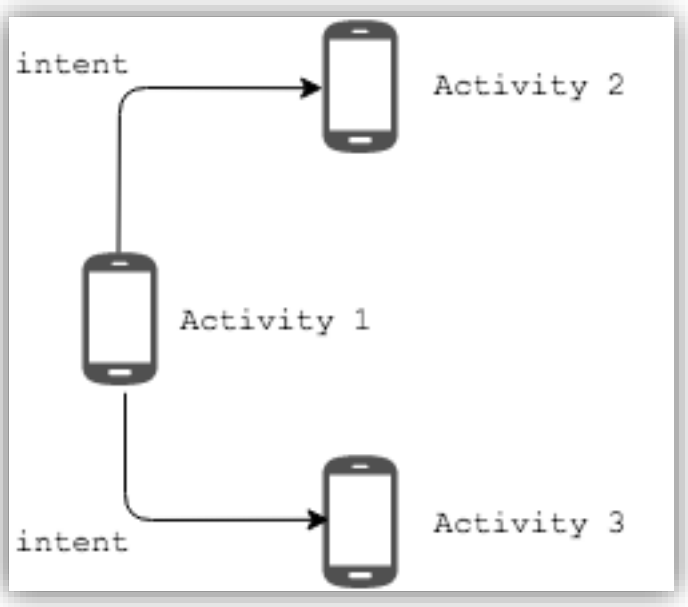
4.2. targetSdkVersion

26

5. Intent

As *intents* permitem que você possa interagir com componentes do mesmo aplicativo (outra *Activity*), bem como iniciar uma *Activity* externa (de outro aplicativo).

Ela nada mais é do que um objeto de mensagem que pode ser utilizado para realizar uma ação de outro componente.



5.1. Intent Explícito

A intente explícita é utilizada para inicializar um componente específico do aplicativo, pode ser uma outra *activity* (atividade) ou um serviço em particular.

```
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

5.2. Intent Implícito

A intent implícita especifica uma ação que possa chamar qualquer aplicativo no dispositivo que possa realizer a ação desejada. Isto é útil quando o seu aplicativo não pode realizar a ação, mas outros aplicativos podem e o usuário pode escolher qual aplicativo ele quer utilizar para a ação desejada.

```
// Create the text message with a string
Intent sendIntent = new Intent();
```

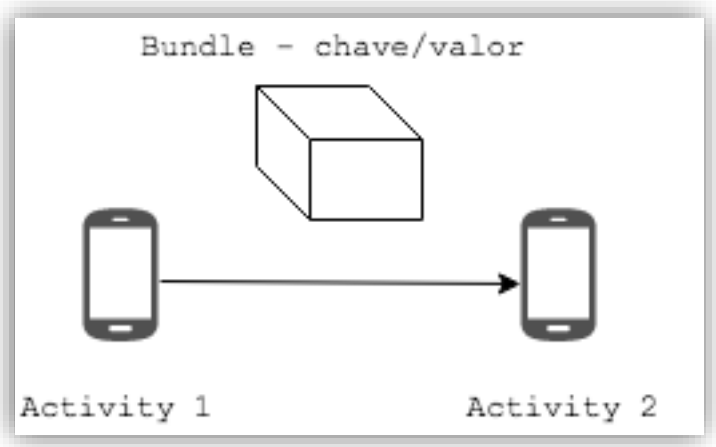
```
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

Quando startActivity() é chamada, o sistema avalia todos os aplicativos que possam processar e realizar a ação que está sendo chamada. Caso o usuário possua somente um aplicativo que possa realizar aquela ação, ele será aberto automaticamente. Caso contrário, o usuário pode optar por qual aplicativo ele deseja executar a ação desejada.

O resolveActivity() é necessário para verificarmos se existe uma aplicação que atenda a nossa chamada da *activity.*

6. Bundle

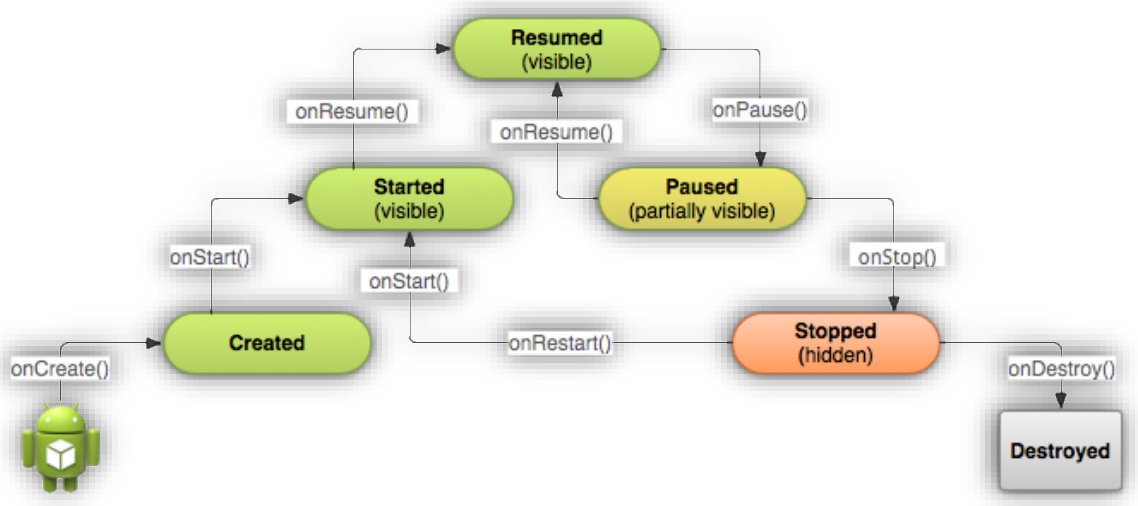
Como opção, nós podemos passar dados de uma *activity* para outra com o *putExtra()*. Nós passamos dois parâmetros: chave e valor. Além disso, podemos criar um objeto do tipo *Bundle* com os dados que queremos passar e adicionamos o *Bundle* em nossa *Intent* com o *putExtras()*.



O componente que recebe a *intent* pode usar a chamada do método *getIntent().getExtras()* para os dados extras que foram adicionados na tela anterior.

7. Ciclo de Vida

À medida que o usuário navega no aplicativo, as instâncias de *Activity* no aplicativo transitam entre diferentes estados no ciclo de vida. Por exemplo, quando uma activity é iniciada pela primeira vez, ou quando o usuário vai para outra atividade, o sistema realiza um conjunto de métodos de ciclo de vida, que podem ser diferentes.



onCreate() – É a primeira função a ser executada em uma activity. Carrega os dados XML e outras operações de inicialização. Executada apenas uma vez.

onStart() – Após o onCreate(), quando a activity volta a ter o foco. Não está mais visível e volta a ter foco.

onResume() – Retomada de foco.

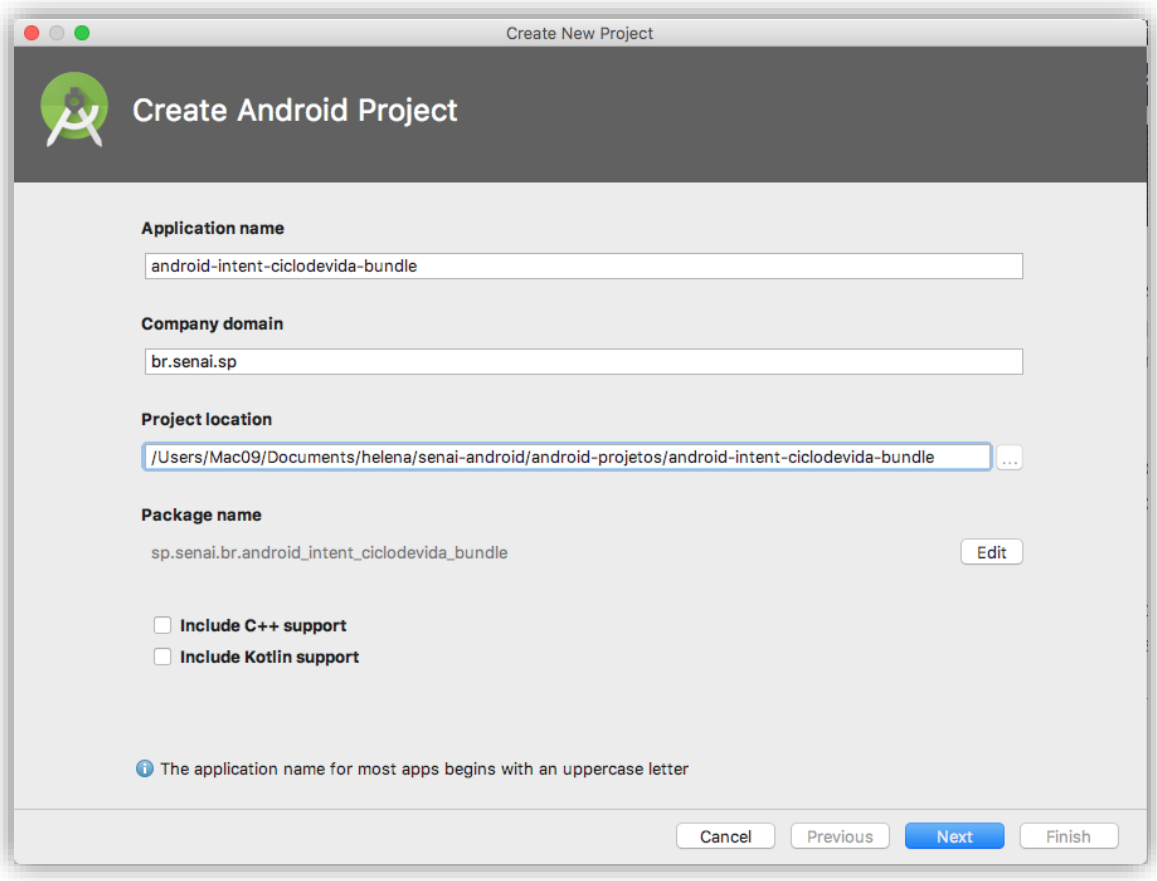
onPause() – Perde o foco (quando uma nova activity é iniciada).

onStop() – Só é chamada quando a activity fica completamente encoberta por outra activity.

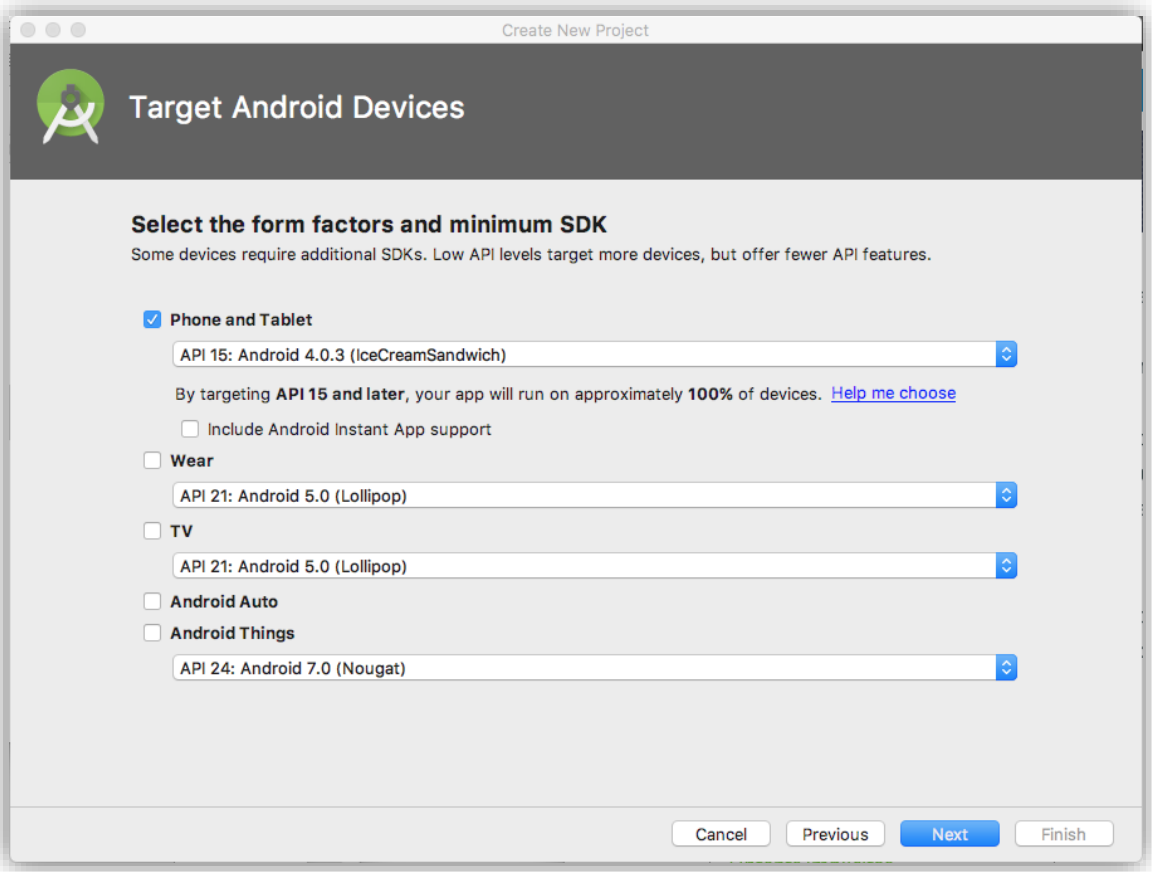
onDestroy() – quando ela é destruída. Ao ser iniciada novamente, ela deve ser recriada.

onRestart() – volta a ter o foco depois de estar em background.

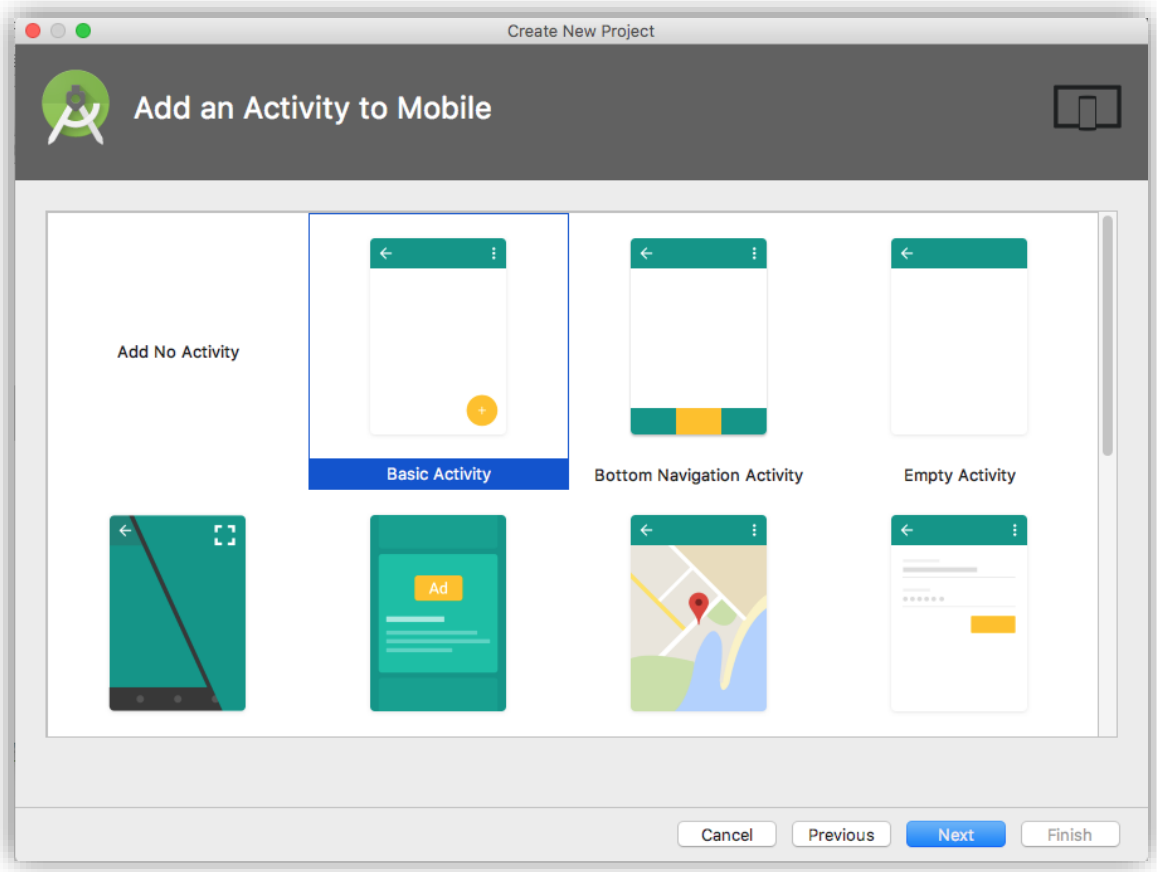
8. Novo Projeto



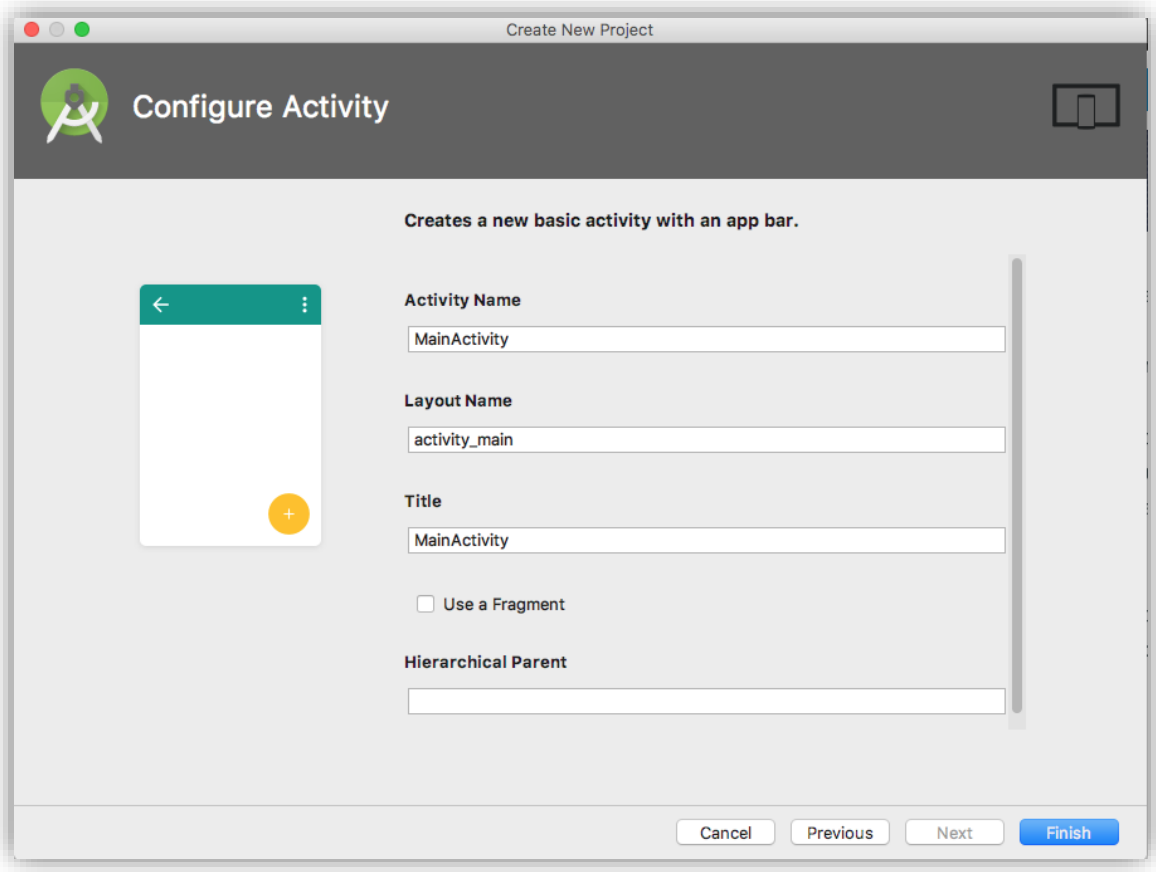
Clicar em "*Next*".



Clicar em "*Next*".



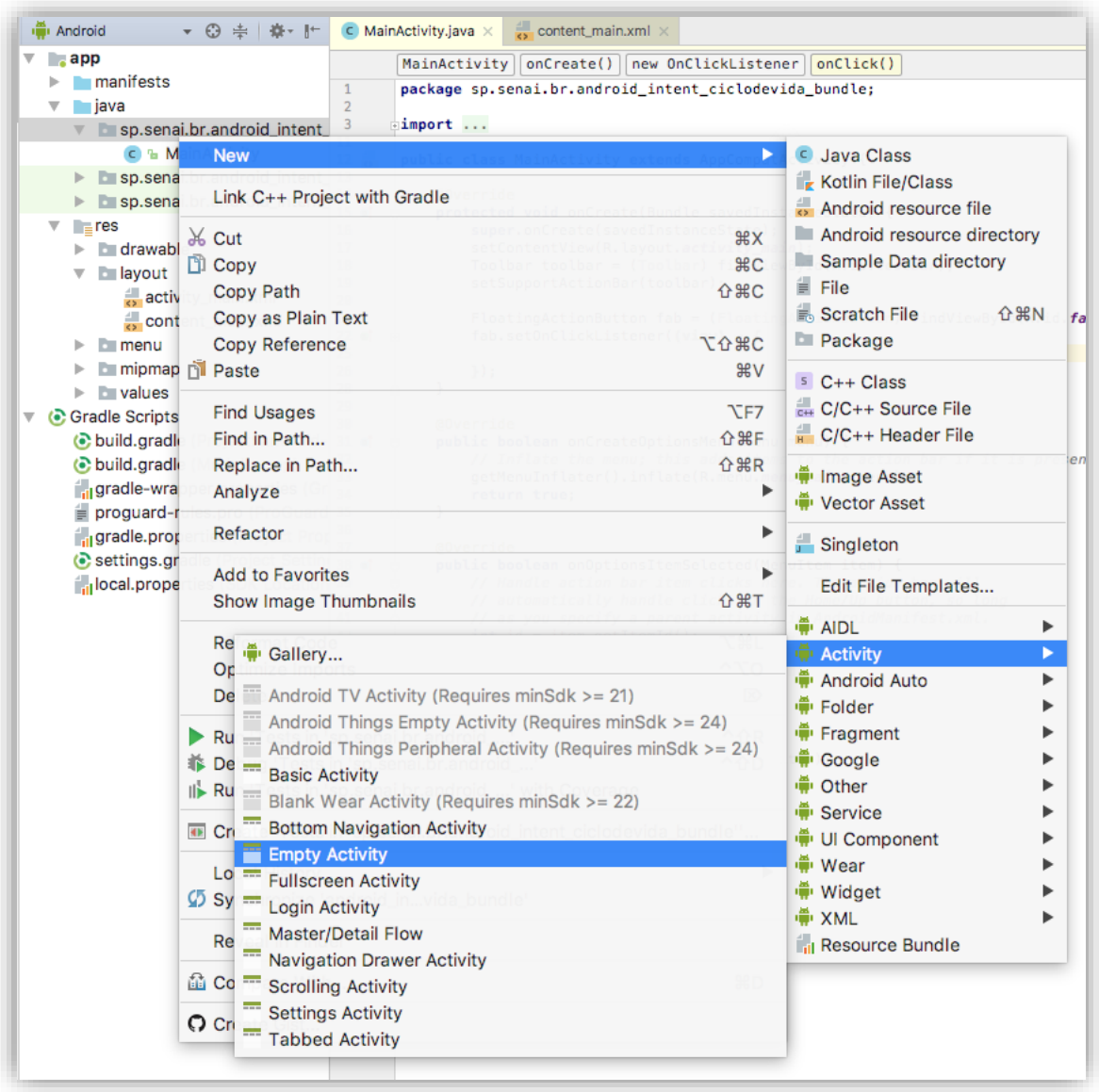
Clicar em "*Next*".



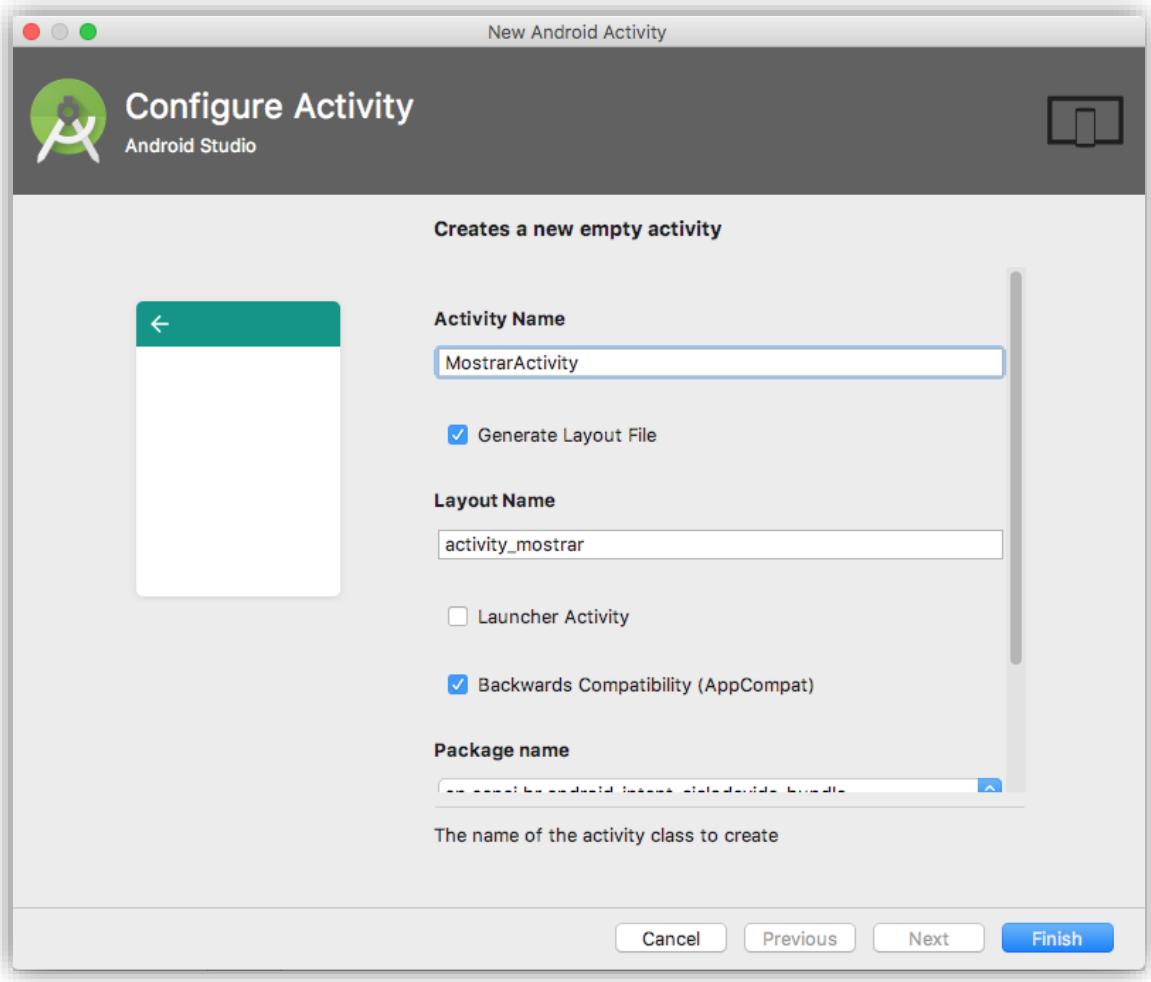
Clicar em "*Finish*".

8.1. Intent

Vamos criar uma nova *activity* em nosso projeto. A primeira coisa que iremos fazer é navegarmos entre as telas. Na etapa seguinte, vamos colocar informações na *intent* e por último vamos mostrar como o ciclo de vida funciona.



Criar uma nova *activity* – Empty Activity chamada “MostrarActivity”.



Clicar em "*Finish*".

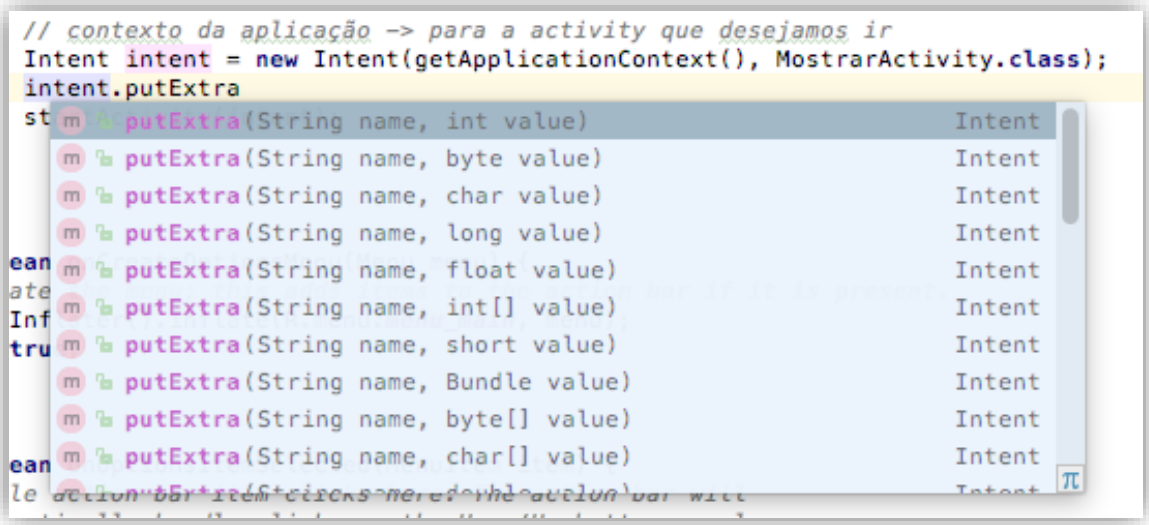
Na nossa MainActivity, no nosso floating button, vamos realizar a nossa primeira *intenção.* Queremos sair da nossa primeira tela e irmos para a nossa segunda tela criada.

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {

            // contexto de onde estamos, para onde queremos ir
            Intent intent = new Intent(getApplicationContext(), MostrarActivity.class);
            startActivity(intent);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

8.1.1. putExtra()

Vamos passar primeiro uma informação simples de uma tela para outra, colocando essa informação primeiramente na *intent.*

Ao começarmos a escrever que queremos colocar um valor na nossa *intent,* vemos que o método já especifica que devemos colocar um par de chave/valor.




```java
public class MainActivity extends AppCompatActivity {

    private String nome;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        nome = "World";

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {

                // contexto da aplicação -> para a activity que desejamos ir
                Intent intent = new Intent(getApplicationContext(), MostrarActivity.class);
                intent.putExtra( name: "hello", nome);
                startActivity(intent);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Vamos inserir um *TextView* no nosso xml de "*activity_mostrar.xml*" para mostrar a informação que estamos trazendo da *activity* anterior.

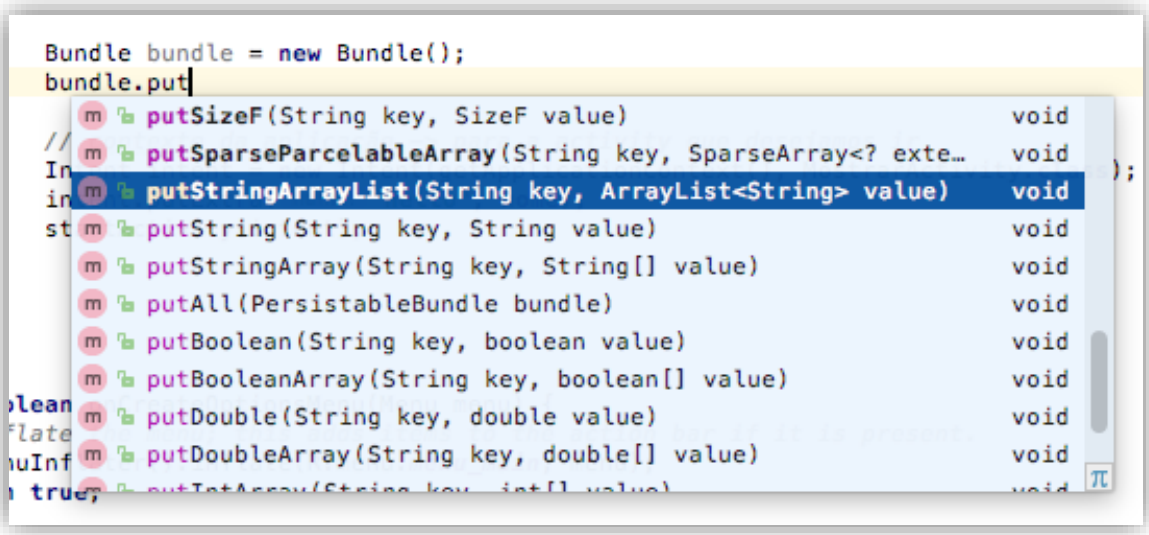
```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sp.senai.br.android_intent_ciclodevida_bundle.MostrarActivity">

    <RelativeLayout
        android:id="@+id/container3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/tvNome"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true"
            android:fontFamily="sans-serif"
            android:letterSpacing="0.03"
            android:text="Another text"
            android:textSize="12sp"
            android:textStyle="normal" />

    </RelativeLayout>

</android.support.constraint.ConstraintLayout>
```

Na *MostrarActivity.java* iremos buscar os valores que estamos passando da activity anterior e mostrarmos na tela.

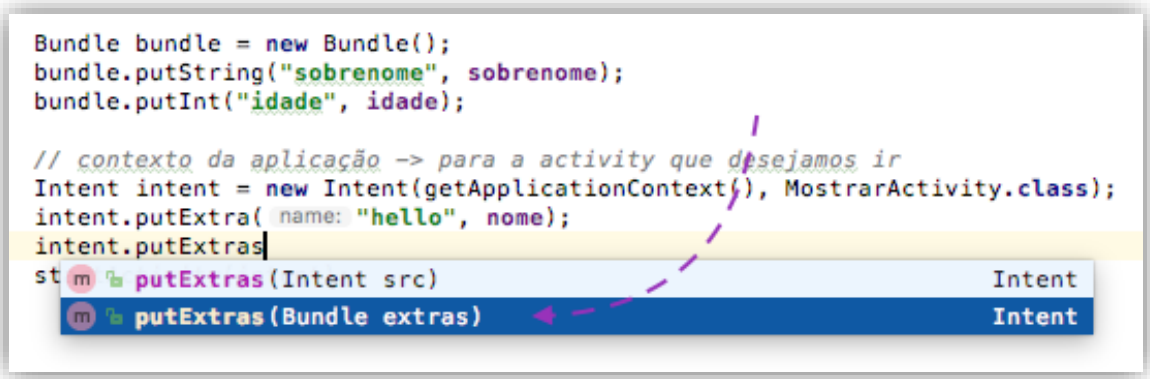
```java
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MostrarActivity extends AppCompatActivity {

    private TextView tvNome;
    private String resultado;

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

        Intent intent = getIntent();

        resultado = intent.getStringExtra( name: "hello");

        tvNome = findViewById(R.id.tvNome);
        tvNome.setText(resultado);

    }
}
```

8.2. Bundle

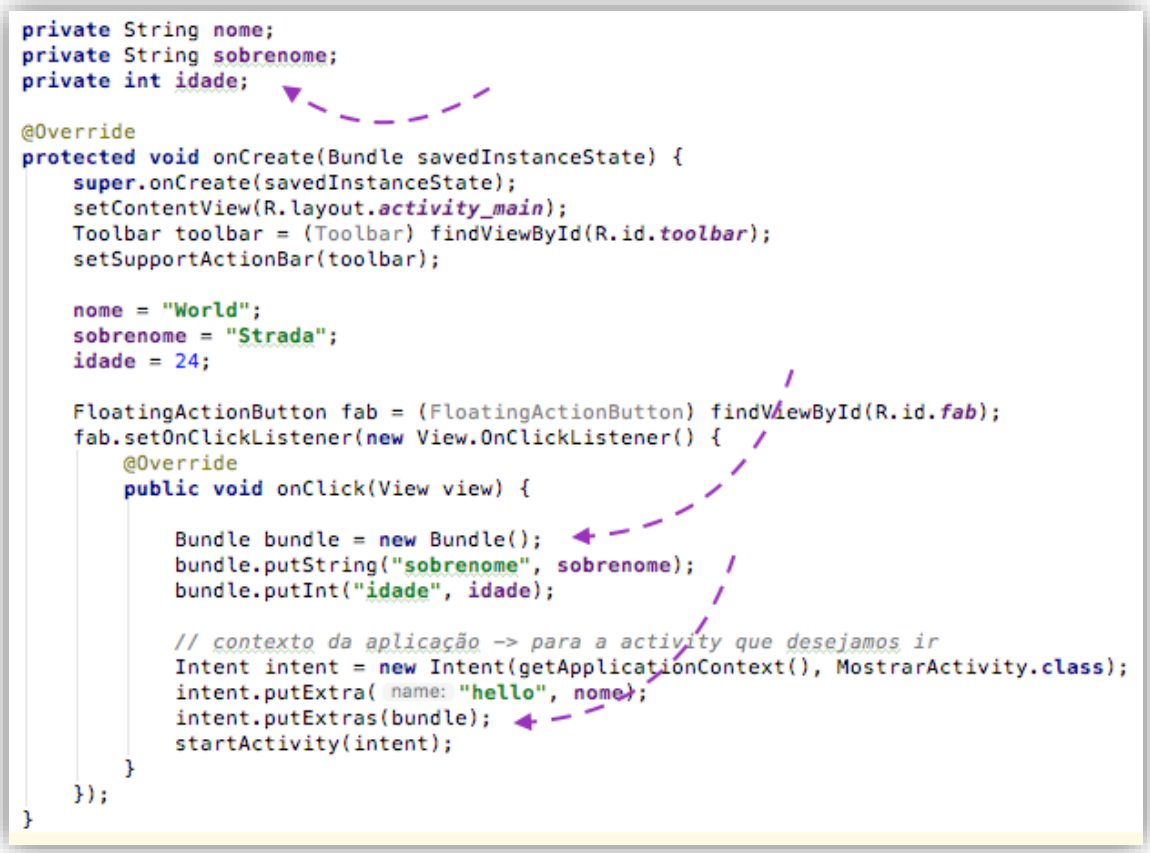
Vamos realizar agora, em cima deste mesmo projeto, um exemplo com *Bundle*.



Com o *bundle* nós conseguimos colocar um par de chave/valor com o tipo de dado que queremos inserir/definir. Por exemplo, além de encaminharmos o nosso sobrenome, vamos também encaminhar a nossa idade.



Vamos colocar na nossa *intent*, o *bundle* que definimos previamente. Código completo da nossa *MainActivity.java*.

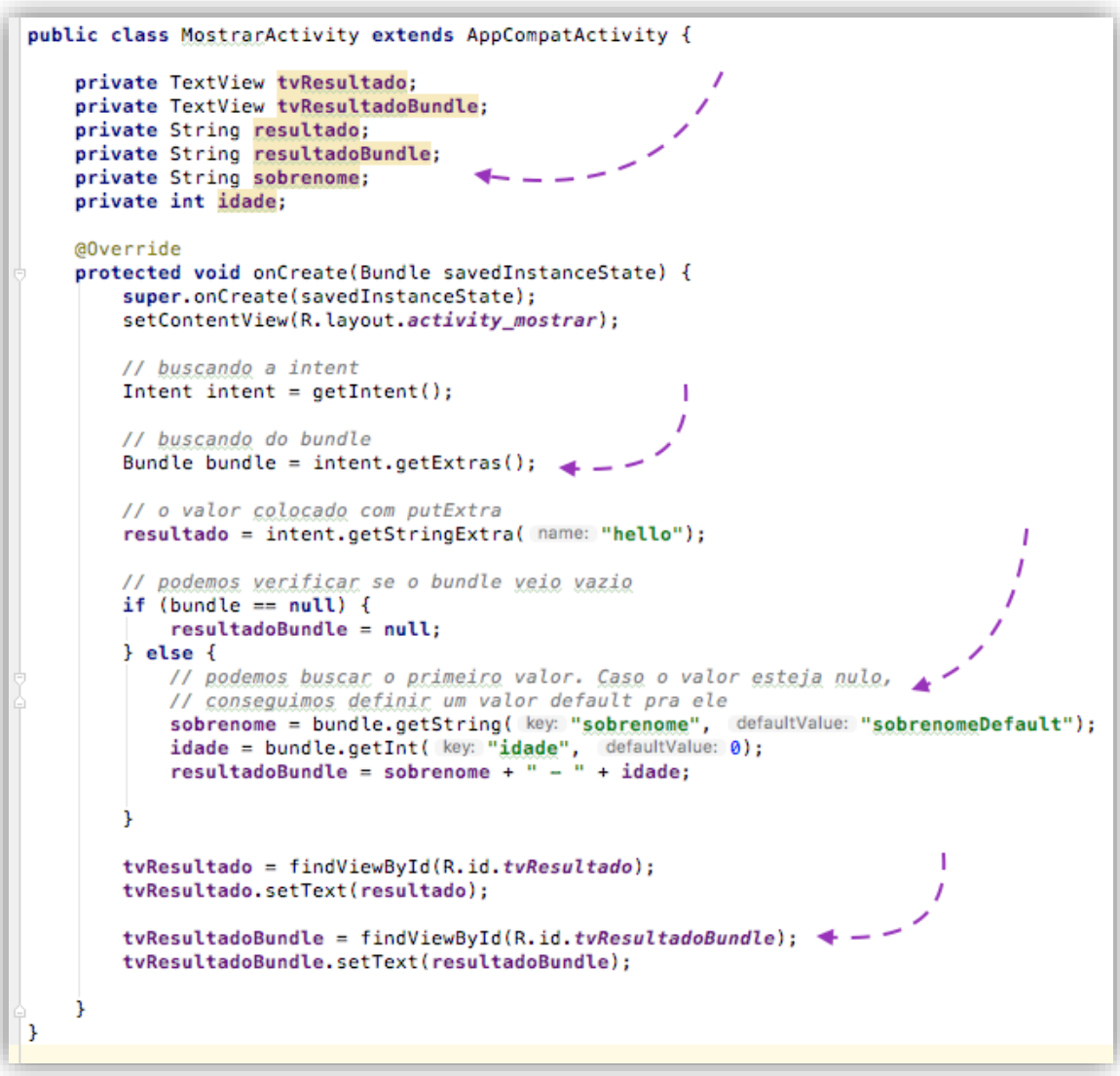


```
private String nome;
private String sobrenome;
private int idade;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    nome = "World";
    sobrenome = "Strada";
    idade = 24;

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            Bundle bundle = new Bundle();
            bundle.putString("sobrenome", sobrenome);
            bundle.putInt("idade", idade);

            // contexto da aplicação -> para a activity que desejamos ir
            Intent intent = new Intent(getApplicationContext(), MostrarActivity.class);
            intent.putExtra( name: "hello", nome);
            intent.putExtras(bundle);
            startActivity(intent);
        }
    });
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
    tools:context="sp.senai.br.android_intent_ciclodevida_bundle.MostrarActivity">

    <LinearLayout
        android:id="@+id/container3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:gravity="center_horizontal"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvResultado"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true"
            android:fontFamily="sans-serif"
            android:letterSpacing="0.03"
            android:text="Another text"
            android:textSize="12sp"
            android:textStyle="normal" />

        <TextView
            android:id="@+id/tvResultadoBundle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true"
            android:fontFamily="sans-serif"
            android:letterSpacing="0.03"
            android:text="Another text"
            android:textSize="12sp"
            android:textStyle="normal" />

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

Vamos fazer uma alteração no nosso layout para mostrarmos as informações que precisamos na nossa aplicação. Vamos deixar como LinearLayout e vamos alinhar como vertical.

Agora que criamos mais um campo *TextView* para mostrarmos o resultado que estaremos recebendo do *bundle*, vamos alterar a nossa *MostrarActivity.java.*



```
public class MostrarActivity extends AppCompatActivity {

    private TextView tvResultado;
    private TextView tvResultadoBundle;
    private String resultado;
    private String resultadoBundle;
    private String sobrenome;
    private int idade;

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

        // buscando a intent
        Intent intent = getIntent();

        // buscando do bundle
        Bundle bundle = intent.getExtras();

        // o valor colocado com putExtra
        resultado = intent.getStringExtra( name: "hello");

        // podemos verificar se o bundle veio vazio
        if (bundle == null) {
            resultadoBundle = null;
        } else {
            // podemos buscar o primeiro valor. Caso o valor esteja nulo,
            // conseguimos definir um valor default pra ele
            sobrenome = bundle.getString( key: "sobrenome",  defaultValue: "sobrenomeDefault");
            idade = bundle.getInt( key: "idade",  defaultValue: 0);
            resultadoBundle = sobrenome + " - " + idade;

        }

        tvResultado = findViewById(R.id.tvResultado);
        tvResultado.setText(resultado);

        tvResultadoBundle = findViewById(R.id.tvResultadoBundle);
        tvResultadoBundle.setText(resultadoBundle);

    }
}
```

8.3. Ciclo de Vida

Na nossa *MainActivity.java* vamos colocar o seguinte conteúdo.

```java
@Override
protected void onStart() {
    super.onStart();
    Log.d( tag: "onStart",  msg: "onStart");
}

@Override
protected void onResume() {
    super.onResume();
    Log.d( tag: "onResume",  msg: "onResume");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.d( tag: "onRestart",  msg: "onRestart");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d( tag: "onDestroy",  msg: "onDestroy");

}

@Override
protected void onPause() {
    super.onPause();
    Log.d( tag: "onPause",  msg: "onPause");
}

@Override
protected void onStop() {
    super.onStop();
    Log.d( tag: "onStop",  msg: "onStop");
}
```

Sempre que precisarmos realizarmos alguma ação quando o nosso ciclo de vida for alterado, nós podemos chamar método correspondente para determinada ação.

9. Resumo

Nesta apostila, mostramos como irmos de uma tela para outra, através das nossas intents, bem como passarmos valores entre as nossas Activities. Além disso, como o sistema trabalha com ciclo de vida de uma aplicação Android.

10. Referências

https://developer.android.com/guide/components/intents-filters.html?hl=pt-br

https://developer.android.com/guide/components/intents-common.html

https://www.androidpro.com.br/intents/

https://developer.android.com/training/basics/activity-lifecycle/starting.html?hl=pt-br

https://developer.android.com/guide/components/activities.html?hl=pt-br#StartingAnActivity