JÖNKÖPING UNIVERSITY

School of Engineering

ANDROID RUNTIME CONFIGURATION CHANGES

Peter Larsson-Green

Jönköping University

Spring 2020

JÖNKÖPING UNIVERSITY
School of Engineering

WHAT ARE "CONFIGURATIONS"?

Things about Android/the device we don't know in advance.

- Screen size
- Preferred language
- Landscape/Portrait mode
- Preferred font size
- Pixel density
- ...

RUNTIME CONFIGURATION CHANGE

Occurs when a Configuration setting changes.

Android will:

1. Destroy your activity instances:
   - `onPause()`, `onStop()` & `onDestroy()` are all called (in that order).
2. Create new instances:
   - `onCreate()`, `onStart()` & `onResume()` are all called (in that order).

Why?

- To load the right resources (e.g. the string resources in a new language).

Consequence: the state of the activity is lost.

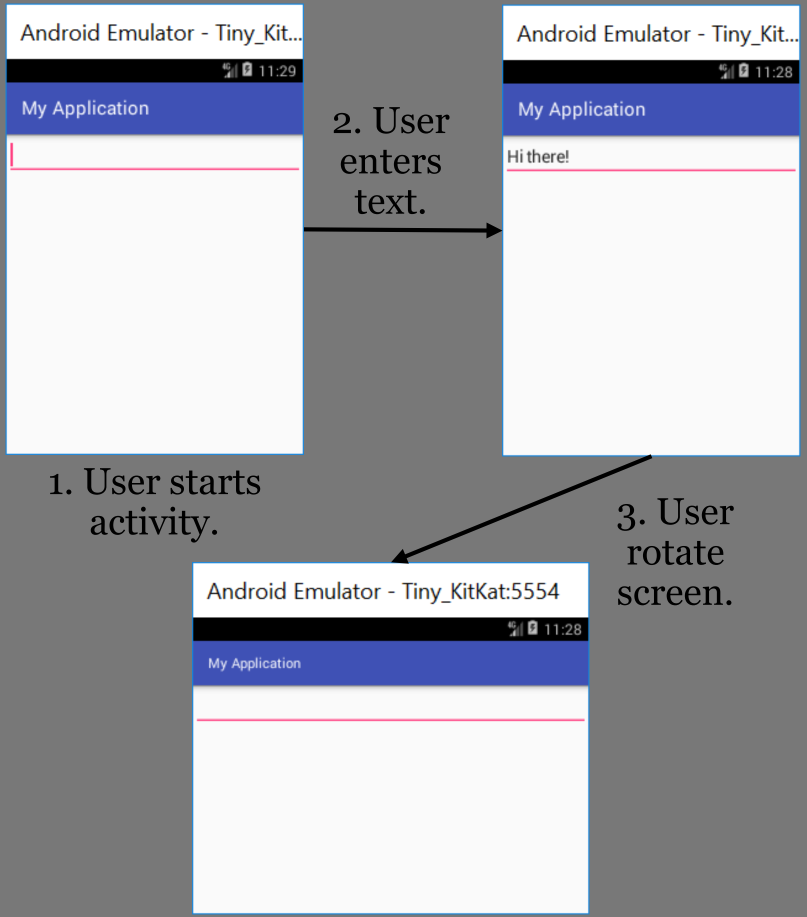
- Needs to be retained.

EXAMPLE: NOT RETAINING THE STATE

```
public class MainActivity extends Activity{
  @Override
  protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
}
```

```
<EditText
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
```

res/layout/activity_main.xml


Android Emulator - Tiny_Kit...
11:29
My Application
Android Emulator - Tiny_Kit...
11:28
My Application
Hi there!
Android Emulator - Tiny_KitKat:5554
11:28
My Application

1. User starts activity.

2. User enters text.

3. User rotate screen.

4. Entered text gone.

JÖNKÖPING UNIVERSITY
School of Engineering

REMEMBERING THE STATE

The method `onSaveInstanceState` is called on configuration changes.

```
@Override
protected void onSaveInstanceState(Bundle outState){
  super.onSaveInstanceState(outState);
  EditText theEditText = (EditText) findViewById(R.id.theEditText);
  String enteredText = theEditText.getText().toString();
  outState.putString("enteredText", enteredText);
}
```

RESTORING THE STATE

Restore the state in `onCreate()`:

Is `null` if not being re-created.

```
@Override
protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  if(savedInstanceState != null){
    String enteredText = savedInstanceState.getString("enteredText");
    EditText theEditText = (EditText) findViewById(R.id.theEditText);
    theEditText.setText(enteredText);
  }
}
```

RESTORING THE STATE

Or restore the state in `onRestoreInstanceState()`:

- Will be called after `onCreate()`.

```
@Override
protected void onRestoreInstanceState(Bundle savedState){
  super.onRestoreInstanceState(savedInstanceState);
  String enteredText = savedInstanceState.getString("enteredText");
  EditText theEditText = (EditText) findViewById(R.id.theEditText);
  theEditText.setText(enteredText);
}
```

ACTUALLY...

The default implementation takes care of most of the UI per-instance state for you by calling onSaveInstanceState() on each view in the hierarchy that has an id, and by saving the id of the currently focused view (all of which is restored by the default implementation of onRestoreInstanceState(Bundle)).

https://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)

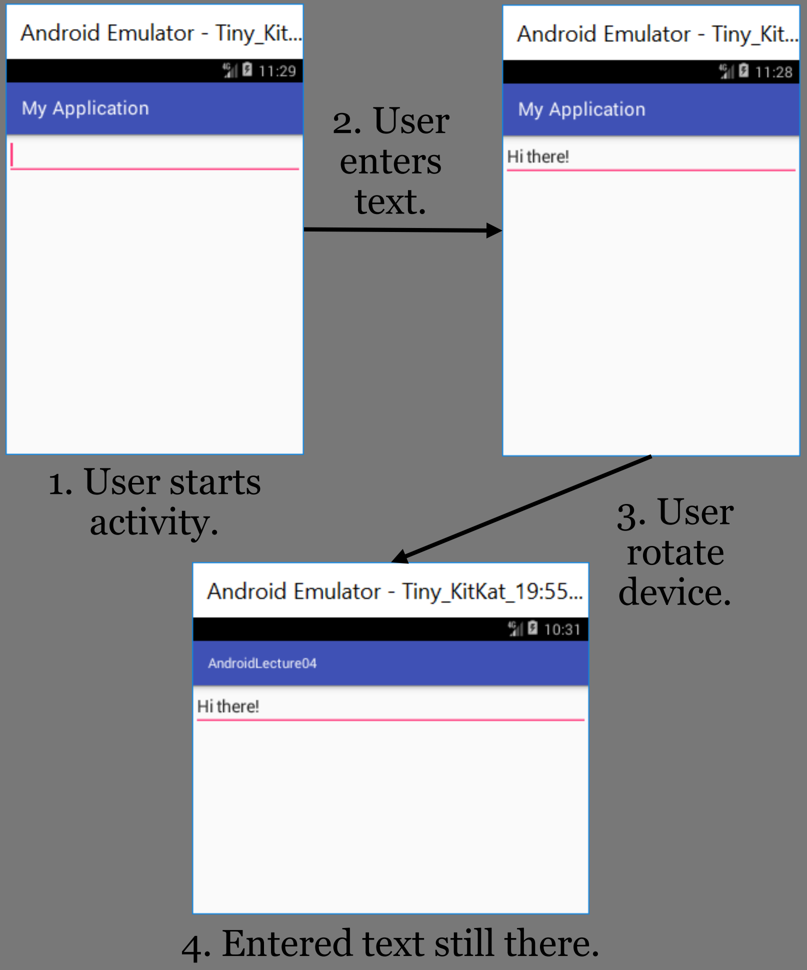
`Views` with their own state are retained by default if you give them an id. $\Rightarrow$ We only need to worry about the state of our data.

EXAMPLE: RETAINING THE STATE

```java
public class MainActivity extends Activity{
  @Override
  protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
}
```

```xml
<EditText
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/theEditText"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
```

res/layout/activity_main.xml



JÖNKÖPING UNIVERSITY
School of Engineering

BAD EXAMPLE

```
public class MainActivity extends Activity{
  private int counter = 0;
  @Override
  protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void inc(View view){
    counter += 1;
    ((Button)
     findViewById(R.id.theButton)
    ).setText(""+counter);
  }
}
```

In `onCreate`, `this.counter` needs to be restored, and we need to change the button text.

`res/layout/activity_main.xml`

```
<Button
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="wrap_parent"
 android:layout_height="wrap_content"
 android:text="0"
 android:onClick="inc"
 android:id="@+id/theButton" />
```

REMEMBERING THE STATE

`Bundle` can store most primitive data types.

`Bundle` can store objects implementing the interface `Parcelable`.

- Many classes you will use do not implement `Parcelable`.

Old solution:

- Return object from `onRetainNonConfigurationInstance()`.
  - Called on the old activity being destroyed.
- Receive object using `getLastNonConfigurationInstance()`.
  - You call it in the activity.
- Deprecated in API level 11 (Recommendation: use model fragments instead).
- Re-introduced in API level 22, but named `onRetainCustomNonConfigurationInstance()` & `getLastCustomNonConfigurationInstance()`.
  - Deprecated again when we got Android Architecture Components (`ViewModel`).