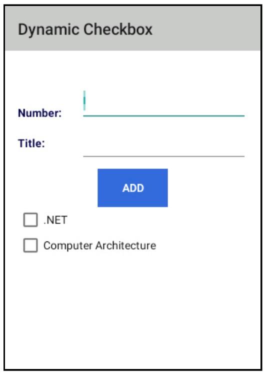
Slide 9.14: Dynamic checkboxes: layout file
Slide 9.16: Dynamic checkboxes: Java source code (cont.)
Home

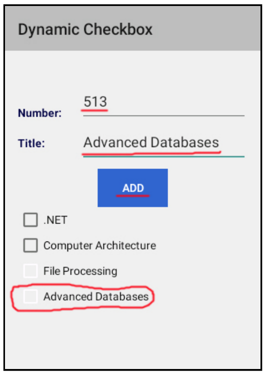
PRINT VERSION

Dynamic Checkboxes: Java Source Code



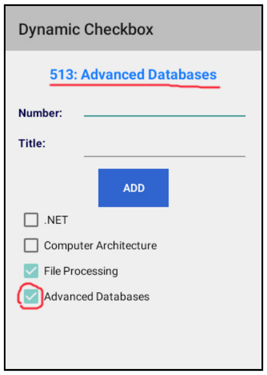
Add 350
& 513

⇒



Check 350
& 513

⇒



Line 03: import java.util.Iterator;

An interface is used to iterate over a `collection`.

Line 04: import java.util.LinkedHashMap;

It is the class of hash table and linked list implementation of the `Map` interface, with predictable iteration order.

Line 05: import java.util.Map;

An object maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.

Line 06: import java.util.Set;

A collection contains no duplicate elements. More formally, sets contain no pair of elements e_1 and e_2 such that e_1`.equals(`e_2`)`, and at most one null element.

Line 13: import android.widget.LinearLayout;

A linear layout arranges its children in a single column or a single row. The direction of the row can be set by calling `setOrientation`. The default orientation is horizontal. Gravity, which specifies the alignment of all the child elements, can be specified by `setGravity`.

Line 16: import android.graphics.Color;

The `Color` class defines methods for creating and converting color ints. Colors are represented as packed ints, made up of 4 bytes: alpha, red, green, blue.

DynamicCheckbox/app/src/main/java/com/ecs/wenchen/dynamiccheckbox/MainActivity.java

```
01 package com.ecs.wenchen.dynamiccheckbox;
02 
03 import java.util.Iterator;
04 import java.util.LinkedHashMap;
05 import java.util.Map;
06 import java.util.Set;
07 
08 import android.app.Activity;
09 import android.os.Bundle;
10 import android.view.View;
11 import android.widget.Button;
12 import android.widget.CheckBox;
13 import android.widget.LinearLayout;
14 import android.widget.TextView;
15 import android.widget.EditText;
16 import android.graphics.Color;
17 
18 public class MainActivity extends Activity {
19   LinearLayout linearBox;
20   CheckBox     checkBox, checkBox1;
21 
22   @Override
23   protected void onCreate( Bundle savedInstanceState ) {
24     super.onCreate( savedInstanceState );
25     setContentView( R.layout.activity_main );
26 
27     linearBox = (LinearLayout) findViewById( R.id.linearBox );
28     final EditText number = (EditText) findViewById( R.id.number );
29     final EditText title  = (EditText) findViewById( R.id.title );
30     final Button   button = (Button)   findViewById( R.id.add );
31 
32     button.setOnClickListener( new View.OnClickListener( ) {
33       public void onClick( View v ) {
34         checkBox1 = new CheckBox( getApplicationContext( ) );
35         checkBox1.setId( Integer.parseInt( number.getText( ).toString( ) ) );
36         checkBox1.setText( title.getText( ).toString( ) );
37         checkBox1.setTextColor( Color.BLACK );
38         checkBox1.setOnClickListener( getOnClickDoSomething( checkBox1 ) );
39         linearBox.addView( checkBox1 );
40       }
41     } );
42 
43     final LinkedHashMap<String, String>
44       alphabet = new LinkedHashMap<String, String>( );
45     alphabet.put( "260", ".NET" );
46     alphabet.put( "370", "Computer Architecture" );
47 
48     Set<?> set = alphabet.entrySet( );
49     // Get an iterator.
50     Iterator<?> i = set.iterator( );
51     // Display elements.
52     while ( i.hasNext( ) ) {
53       @SuppressWarnings( "rawtypes" )
```

```java
      Map.Entry me = ( Map.Entry ) i.next( );
      checkBox     = new CheckBox( this );
      checkBox.setId( Integer.parseInt( me.getKey( ).toString( ) ) );
      checkBox.setText( me.getValue( ).toString( ) );
      checkBox.setOnClickListener( getOnClickDoSomething( checkBox ) );
      linearBox.addView( checkBox );
    }
  }

  View.OnClickListener getOnClickDoSomething( final Button button ) {
    return new View.OnClickListener( ) {
      public void onClick( View v ) {
        final TextView tvView = (TextView) findViewById( R.id.textView3 );
        tvView.setText( button.getId( ) + ": " + button.getText( ).toString( ) );
      }
    };
  }
}
```

Slide 9.14: Dynamic checkboxes: layout file
Slide 9.16: Dynamic checkboxes: Java source code (cont.)
Home

PRINT VERSION