```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity01;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity01;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity01;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

/*
 * Activity01
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates Hello, World by programmatically creating
 *  a layout. Code adapted from:
 * http://developer.android.com/resources/tutorials/hello-world.html
 */


public class Code1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Ohai!");
        setContentView(tv);

        // sample logging capability (check with logcat)
        Log.i("Activity01", "Completed onCreate()");
        Log.e("Activity01", "This is an error");
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity02;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity02;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity02;

import android.app.Activity;
import android.os.Bundle;
import net.cs76.lectures.activity02.R;

/*
 * Activity02
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates Hello, World layout using XML.
 * This is actually the default code and package created
 * by Eclipse when making a new Android project.
 */

public class Code2 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity03;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity03;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity03;

import net.cs76.lectures.activity03.R;
import android.app.Activity;
import android.os.Bundle;

/*
 * Activity03
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates LinearLayout and some of its options.
 */


public class Code3 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity04;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity04;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int MyButton=0x7f050000;
        public static final int MyButton2=0x7f050001;
        public static final int MyText=0x7f050002;
        public static final int MyText2=0x7f050003;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity04;

import android.app.Activity;
import android.os.Bundle;
import net.cs76.lectures.activity04.R;

/*
 * Activity04
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates layout options including layout_, gravity,
 * and weights.
 */

public class Code4 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity05;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity05;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
    public static final class style {
        public static final int styleName=0x7f050000;
    }
}
```

```java
package net.cs76.lectures.activity05;

import net.cs76.lectures.activity05.R;
import android.app.Activity;
import android.os.Bundle;

/*
 * Activity05
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates a TableLayout
 */

public class Code5 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity06;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity06;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int accept=0x7f050002;
        public static final int cancel=0x7f050003;
        public static final int myLabel=0x7f050000;
        public static final int numberStudents=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity06;

import net.cs76.lectures.activity06.R;
import android.app.Activity;
import android.os.Bundle;

/*
 * Activity06
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates a RelativeLayout
 * Code adapted from:
 * http://developer.android.com/resources/tutorials/views/hello-relativelayout.html
 */


public class Code6 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity07;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity07;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int accept=0x7f050002;
        public static final int cancel=0x7f050003;
        public static final int myLabel=0x7f050000;
        public static final int numberStudents=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity07;

import net.cs76.lectures.activity07.R;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

/*
 * Activity07
 * Dan Armendariz
 * Computer Science E-76
 *
 * Introducing event handlers to allow user interaction.
 */


public class Code7 extends Activity {

    // define an OnClickListener that is an anonymous function
    private OnClickListener respondToClick = new OnClickListener() {
        // the onClick() method is called when the OnClickListener
        // detects a click event.
        public void onClick(View v) {
            // get the current context to display a Toast (below)
            Context context = getApplicationContext();

            CharSequence text;

            // figure out which button was clicked by comparing
            // the View's ID with known IDs.
            switch(v.getId()) {
                case R.id.accept: text = "accept pushed!"; break;
                case R.id.cancel: text = "cancel pushed!"; break;
                default: text="Dunno what was pushed!";
            }

            // show a Toast for a short amount of time, displaying
            // which button was pushed.
            int duration = Toast.LENGTH_SHORT;
            Toast toast = Toast.makeText(context, text, duration);
            toast.show();
        }

    };
```

```
49.
50.     /** Called when the activity is first created. */
51.     @Override
52.     public void onCreate(Bundle savedInstanceState) {
53.         super.onCreate(savedInstanceState);
54.         setContentView(R.layout.main);
55.
56.         // find our button in the UI by its ID
57.         Button accept = (Button)findViewById(R.id.accept);
58.
59.         // tell the OnClickListener which method to call
60.         // when a click is detected.
61.         accept.setOnClickListener(respondToClick);
62.
63.
64.         // find our button in the UI by its ID
65.         Button cancel = (Button)findViewById(R.id.cancel);
66.
67.         // tell the OnClickListener which method to call
68.         // when a click is detected.
69.         cancel.setOnClickListener(respondToClick);
70.
71.     }
72.
73. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity08;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity08;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int accept=0x7f050002;
        public static final int cancel=0x7f050003;
        public static final int myLabel=0x7f050000;
        public static final int numberStudents=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity08;

import net.cs76.lectures.activity08.R;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnLongClickListener;
import android.widget.Button;
import android.widget.Toast;

/*
 * Activity08
 * Dan Armendariz
 * Computer Science E-76
 *
 * Another (better?) way to implement event handlers.
 */


public class Code8 extends Activity implements OnClickListener, OnLongClickListener {

    private Button accept, cancel;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find our button in the UI by its ID
        accept = (Button)findViewById(R.id.accept);

        // set listeners for the button. Since our current class
        // implements OnClickListener and OnLongClickListener interfaces
        // we can simply pass the current object ("this") into
        // the listeners. The appropriate methods will therefore
        // be called when the event fires.
        accept.setOnClickListener(this);
        accept.setOnLongClickListener(this);

        // find our button in the UI by its ID
        cancel = (Button)findViewById(R.id.cancel);

        // set listeners for the button
        cancel.setOnClickListener(this);
        cancel.setOnLongClickListener(this);
```

```
49.
50.     }
51.
52.     /* display a Toast with message text. */
53.     private void showMessage(CharSequence text) {
54.         Context context = getApplicationContext();
55.         int duration = Toast.LENGTH_SHORT;
56.         Toast toast = Toast.makeText(context, text, duration);
57.         toast.show();
58.     }
59.
60.     /* implement an event handler that responds to click events */
61.     public void onClick(View v) {
62.         CharSequence text;
63.
64.         // find out which button was pushed based on its ID
65.         switch(v.getId()) {
66.             case R.id.accept: text = "'accept' clicked!"; break;
67.             case R.id.cancel: text = "'cancel' clicked!"; break;
68.             default: text="Dunno what was pushed!";
69.         }
70.
71.         // notify the user which button was clicked
72.         showMessage(text);
73.     }
74.
75.     /* implement an event handler that responds to long click events */
76.     public boolean onLongClick(View v) {
77.         // we're only accepting LongClick events from buttons,
78.         // so we'll caste our View as a button to access its text.
79.         Button pushed = (Button)v;
80.
81.         // no need to figure out which button was clicked! We can
82.         // ask it by way of the Button's getText() method.
83.         showMessage(pushed.getText() + " LongClick'd!");
84.
85.         // we must return true to notify upstream processes that
86.         // we've handled the onLongClick event.
87.         return true;
88.     }
89.
90.
91.
92. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity09;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity09;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int accept=0x7f050002;
        public static final int cancel=0x7f050003;
        public static final int myLabel=0x7f050000;
        public static final int numberStudents=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity09;

import net.cs76.lectures.activity09.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

/*
 * Activity09
 * Dan Armendariz
 * Computer Science E-76
 *
 * A better dialog and accessing data from the text box.
 */


public class Code9 extends Activity implements OnClickListener {

    private Button accept, cancel;
    private EditText numberStudents;

    // define some nice names for our Dialog IDs
    static final int DIALOG_CANCEL_ID = 0;
    static final int DIALOG_ACCEPT_ID = 1;


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find our button in the UI by its ID and assign a listener
        accept = (Button)findViewById(R.id.accept);
        accept.setOnClickListener(this);

        // find our button in the UI by its ID and assign a listener
        cancel = (Button)findViewById(R.id.cancel);
        cancel.setOnClickListener(this);

        // find our text field by its ID
        numberStudents = (EditText)findViewById(R.id.numberStudents);
```

```
49.     }
50.
51.     /* onCreateDialog function that gets called by the system the first
52.      * time a dialog is requested.
53.      */
54.     protected Dialog onCreateDialog(int id) {
55.         // we must return a dialog, so create one in memory
56.         AlertDialog dialog = null;
57.
58.         // specifically, we'll build an "AlertDialog";
59.         // the most common type of dialog.
60.         AlertDialog.Builder builder = new AlertDialog.Builder(this);
61.
62.         // build different dialogs based on the dialog ID passed to us
63.         switch(id) {
64.
65.         // in this case, we're building a 'cancel' dialog
66.         case DIALOG_CANCEL_ID:
67.             // build a dialog with a "Are you sure" message and two buttons:
68.             // 'yes' will kill the app
69.             // 'no' will close the dialog
70.             builder.setMessage("Are you sure you want to cancel?")
71.                    .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
72.                        public void onClick(DialogInterface dialog, int id) {
73.                            Code9.this.finish();
74.                        }
75.                    })
76.                    .setNegativeButton("No", new DialogInterface.OnClickListener() {
77.                        public void onClick(DialogInterface dialog, int id) {
78.                            dialog.cancel();
79.                        }
80.                    });
81.             // generate the dialog object from the options
82.             // passed to the builder
83.             dialog = builder.create();
84.             break;
85.
86.         // we're asked to build a 'accept' dialog
87.         case DIALOG_ACCEPT_ID:
88.             // we've accepted some text, so might as well display it
89.             // with an option to close the dialog.
90.             builder.setMessage("You've entered the text: '"+numberStudents.getText()+"'")
91.                    .setCancelable(false) // disallow user to hit the 'back' button
92.                    .setNeutralButton("Cool!", new DialogInterface.OnClickListener() {
93.                        public void onClick(DialogInterface dialog, int id) {
94.                            dialog.dismiss();
95.                        }
96.                    });
```

```
            // generate the dialog object from the options
            // passed to the builder
            dialog = builder.create();
            break;

        // unknown id, so don't build a dialog
        default: dialog = null;
        }

        return dialog;
    }

    /* onPrepareDialog is called by the system every time
     * a dialog is requested.
     */
    protected void onPrepareDialog(int id, Dialog dialog) {
        if(id == DIALOG_ACCEPT_ID) {
            ((AlertDialog) dialog).setMessage("You've entered the text: '"+numberStudents.getText()+"'");
        }
    }

    public void onClick(View v) {
        switch(v.getId()) {
            case R.id.accept: showDialog(DIALOG_ACCEPT_ID); break;
            case R.id.cancel: showDialog(DIALOG_CANCEL_ID); break;
        }
    }


}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.activity10;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.activity10;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int custom_list_item=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.activity10;

import net.cs76.lectures.activity10.R;
import android.app.ListActivity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

/*
 * Activity10
 * Dan Armendariz
 * Computer Science E-76
 *
 * Displaying data in a List layout with AdapterView. Code adapted from:
 * http://developer.android.com/resources/tutorials/views/hello-listview.html
 */

public class Code10 extends ListActivity implements OnItemClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // no layout is loaded! There's no setContentView().

        // add a ListView to fill the entire screen of the
        // ListActivity, and pass into it an ArrayAdapter
        // that manages the array of list items.
        // Can also pass it an android-created list item:
        //   android.R.layout.simple_list_item_1
        setListAdapter(new ArrayAdapter<String>(this, R.layout.custom_list_item, COLORS));

        // obtain the ListView that was created by setListAdapter()
        ListView myList = getListView();

        // allow us to filter the list with keypresses
        myList.setTextFilterEnabled(true);

        // implement an onClick listener for when a user taps a color
        myList.setOnItemClickListener(this);

    }
```

```
49.
50.     /* display a Toast with message text. */
51.     private void showMessage(CharSequence text) {
52.         Context context = getApplicationContext();
53.         int duration = Toast.LENGTH_SHORT;
54.         Toast toast = Toast.makeText(context, text, duration);
55.         toast.show();
56.     }
57.
58.     /* this method is fired when an item is clicked */
59.     public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
60.         TextView item = (TextView)v;
61.         showMessage(item.getText());
62.     }
63.
64.
65.     // define a list of colors that the list will display
66.     static final String[] COLORS = new String[] {
67.         "Red",
68.         "Orange",
69.         "Yellow",
70.         "Green",
71.         "Blue",
72.         "Indigo",
73.         "Violet"
74.     };
75.
76.
77. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents01;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents01;

public final class R {
    public static final class attr {
    }
    public static final class color {
        public static final int red=0x7f040000;
        public static final int translucent_blue=0x7f040001;
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int firstHello=0x7f060000;
        public static final int secondHello=0x7f060001;
        public static final int thirdHello=0x7f060002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f050004;
        public static final int hello1=0x7f050000;
        public static final int hello2=0x7f050001;
        public static final int hello3=0x7f050002;
        public static final int hello4=0x7f050003;
        public static final int valid=0x7f050005;
        public static final int valid2=0x7f050006;
    }
}
```

```java
package net.cs76.lectures.intents01;

import net.cs76.lectures.intents01.R;
import android.app.Activity;
import android.os.Bundle;

/*
 * Intents01
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrating simple resource types.
 * Compare this code and layout to that of Activity03,
 * on which this project is based.
 */

public class Code1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents02;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents02;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.intents02;

import net.cs76.lectures.intents02.R;
import android.app.Activity;
import android.os.Bundle;

/*
 * Intents02
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrating Android's automatic localization via resources.
 */

public class Code2 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents03;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents03;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int img_1990=0x7f020001;
        public static final int img_1998=0x7f020002;
        public static final int img_2000=0x7f020003;
        public static final int img_2101=0x7f020004;
    }
    public static final class id {
        public static final int image=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.intents03;

import net.cs76.lectures.intents03.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;

/*
 * Intents03
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrating showing an image resource.
 */

public class Code3 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // connect the ImageView to an object
        ImageView img = (ImageView)findViewById(R.id.image);

        // set the image in the ImageView
        img.setImageResource(R.drawable.img_2000);

    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents04;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents04;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int citation=0x7f020000;
        public static final int icon=0x7f020001;
    }
    public static final class id {
        public static final int button=0x7f060000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class raw {
        public static final int ferrari250=0x7f040000;
    }
    public static final class string {
        public static final int app_name=0x7f050002;
        public static final int pause=0x7f050001;
        public static final int play=0x7f050000;
    }
}
```

```java
package net.cs76.lectures.intents04;

import net.cs76.lectures.intents04.R;
import android.app.Activity;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

/*
 * Intents04
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates playing a simple sound file resource, with
 * support for pausing, restarting, and responding to a
 * completed sound event.
 */


public class Code4 extends Activity implements OnClickListener, OnCompletionListener {

    MediaPlayer player;
    Button vroom;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // connect the button to an object
        vroom = (Button)findViewById(R.id.button);
        vroom.setOnClickListener(this);
    }

    // Consider why this instantiation is here and not in onCreate()!
    public void onResume() {
        super.onResume();

        // instantiate a MediaPlayer instance
        player = MediaPlayer.create(getApplicationContext(), R.raw.ferrari250);
        player.setLooping(false);
        player.setOnCompletionListener(this);
    }

    /* We should always release the MediaPlayer object when not in use. See:
```

```
49.       * http://developer.android.com/reference/android/media/MediaPlayer.html#release()
50.       */
51.      public void onPause() {
52.          super.onPause();
53.
54.          // release the MediaPlayer resource
55.          player.release();
56.          player = null;
57.      }
58.
59.      // when the button is clicked we want to hear a sound
60.      public void onClick(View v) {
61.
62.          // play or pause the sound, as appropriate
63.          if(player.isPlaying()) {
64.              player.pause();
65.
66.              // reset button text
67.              vroom.setText(R.string.play);
68.          } else {
69.              vroom.setText(R.string.pause);
70.              player.start();
71.          }
72.      }
73.
74.      // reset button text when the sound is done playing
75.      public void onCompletion(MediaPlayer mp) {
76.          vroom.setText(R.string.play);
77.      }
78.  }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents05;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents05;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int bmw3=0x7f020000;
        public static final int citation=0x7f020001;
        public static final int icon=0x7f020002;
    }
    public static final class id {
        public static final int button=0x7f070001;
        public static final int hare=0x7f070003;
        public static final int image=0x7f070000;
        public static final int tortoise=0x7f070002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class menu {
        public static final int options_menu=0x7f060000;
    }
    public static final class raw {
        public static final int earlycarengine=0x7f040000;
        public static final int ferrari250=0x7f040001;
    }
    public static final class string {
        public static final int app_name=0x7f050001;
        public static final int hello=0x7f050000;
        public static final int pause=0x7f050003;
        public static final int play=0x7f050002;
    }
}
```

```java
package net.cs76.lectures.intents05;

import net.cs76.lectures.intents05.R;
import android.app.Activity;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

/*
 * Intents05
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates creating an Options menu (accessible
 * by pushing the MENU key on the device).
 */

public class Code5 extends Activity implements OnClickListener, OnCompletionListener {

    MediaPlayer player;
    ImageView image;
    Button vroom;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find the image and button Views, and set the listener for the button
        image = (ImageView)findViewById(R.id.image);

        vroom = (Button)findViewById(R.id.button);
        vroom.setOnClickListener(this);

    }

    // cause a media player instantiation when the activity comes to the front
    public void onResume() {
        super.onResume();

        // by default, we'll show and play the sound for the Hare
```

```
49.         setCar(R.id.hare);
50.     }
51.
52.     // release the media player's resources when paused
53.     public void onPause() {
54.         super.onPause();
55.
56.         player.release();
57.         player = null;
58.     }
59.
60.     // initialize the media player with a sound and give the
61.     // ImageView an image, depending on the requested option
62.     public void setCar(int car) {
63.         // set different MediaPlayer or Image options based on the input
64.
65.         // if the player has already been initialized, we should reset it.
66.         // this will stop playing a sound (if one is playing)
67.         if(player != null) {
68.             player.reset();
69.             vroom.setText(R.string.play);
70.         }
71.
72.         switch(car) {
73.             case R.id.tortoise:
74.                 player = MediaPlayer.create(this, R.raw.earlycarengine);
75.                 image.setImageResource(R.drawable.bmw3);
76.                 break;
77.             case R.id.hare:
78.             default:
79.                 player = MediaPlayer.create(this, R.raw.ferrari250);
80.                 image.setImageResource(R.drawable.citation);
81.         }
82.
83.         // set some options for the player no matter which we picked
84.         player.setLooping(false);
85.         player.setOnCompletionListener(this);
86.
87.     }
88.
89.     // override the onCreateOptionsMenu that gets
90.     // called when the user requests a menu
91.     public boolean onCreateOptionsMenu(Menu menu) {
92.
93.         // can also use the add() method to programmatically create a menu
94.         MenuInflater inflater = getMenuInflater();
95.         inflater.inflate(R.menu.options_menu, menu);
96.         return true;
```

```
 97.     }
 98.
 99.     // when an item is selected in the menu, onOptionsItemSelected
100.     // is called and passes the selected 'item'
101.     public boolean onOptionsItemSelected(MenuItem item) {
102.         // change our car based on the selection
103.         setCar(item.getItemId());
104.
105.         return true;
106.     }
107.
108.
109.     // FROM Intents04
110.     // when the button is clicked we want to hear a sound
111.     public void onClick(View v) {
112.
113.         if(player.isPlaying()) {
114.             player.pause();
115.             vroom.setText(R.string.play);
116.         } else {
117.             vroom.setText(R.string.pause);
118.             player.start();
119.         }
120.
121.     }
122.
123.     // FROM Intents04
124.     // reset button text when the sound is done playing
125.     public void onCompletion(MediaPlayer mp) {
126.         vroom.setText(R.string.play);
127.     }
128.
129. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents06;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents06;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int thumbsup=0x7f020001;
    }
    public static final class id {
        public static final int OK=0x7f050003;
        public static final int image=0x7f050002;
        public static final int no=0x7f050001;
        public static final int yes=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
        public static final int thumbsup=0x7f030001;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.intents06;

import net.cs76.lectures.intents06.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

/*
 * Intents06
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrating intents and starting a second Activity.
 */

public class Code6 extends Activity implements OnClickListener{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find our buttons and set onClickListeners for both
        Button yes = (Button)findViewById(R.id.yes);
        Button no  = (Button)findViewById(R.id.no);
        yes.setOnClickListener(this);
        no.setOnClickListener(this);
    }

    public void onClick(View v) {

        // do something based on the button that was clicked
        switch(v.getId()) {
            /* if the "yes" button is clicked, let's start a new
             * activity by creating an Intent and passing it
             * into the startActivity() method.
             */

            case R.id.yes:
                    // create an intent indicating we want
                    // to start the ThumbsUp activity.
                    // Important! Make sure your activity is
                    // in the AndroidManifest.xml file!
                    Intent i = new Intent(this, ThumbsUp.class);

```

```
49.                     // start the activity based on the Intent
50.                     startActivity(i);
51.                     finish();
52.                 break;
53.
54.             // if "no" was selected, we'll quit the application.
55.             case R.id.no:
56.             default:
57.                 finish();
58.         }
59.
60.     }
61. }
```

```java
package net.cs76.lectures.intents06;

import net.cs76.lectures.intents06.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

/*
 * ThumbsUp.java
 * A second activity in Intents06.
 */

public class ThumbsUp extends Activity implements OnClickListener {

    // As with the Activities we've seen thus far, the system
    // calls the onCreate() method when an Activity starts up.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // call the parent's onCreate() method.
        super.onCreate(savedInstanceState);

        // set the ContentView to a layout we designed
        // for this activity.
        setContentView(R.layout.thumbsup);

        // Retrieve the button and set an onClickListener
        Button ok = (Button)findViewById(R.id.OK);
        ok.setOnClickListener(this);

    }

    // When a button is clicked, we'll run the finished()
    // method which ends this activity.
    public void onClick(View v) {
        finish();
    }

}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents07;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents07;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int email=0x7f050005;
        public static final int map=0x7f050002;
        public static final int sms=0x7f050004;
        public static final int stview=0x7f050003;
        public static final int tel=0x7f050001;
        public static final int url=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.intents07;

import net.cs76.lectures.intents07.R;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

/*
 * Intents07
 * Dan Armendariz
 * Computer Science E-76
 *
 * Using Intents to start Activities from other applications.
 */

public class Code7 extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find all buttons in the layout
        Button url   = (Button)findViewById(R.id.url);
        Button tel   = (Button)findViewById(R.id.tel);
        Button map   = (Button)findViewById(R.id.map);
        Button stvw  = (Button)findViewById(R.id.stview);
        Button sms   = (Button)findViewById(R.id.sms);
        Button email = (Button)findViewById(R.id.email);

        // set event handlers for each button
        url.setOnClickListener(this);
        tel.setOnClickListener(this);
        map.setOnClickListener(this);
        stvw.setOnClickListener(this);
        sms.setOnClickListener(this);
        email.setOnClickListener(this);

    }

    /* display a Toast with message text. */
    private void showMessage(CharSequence text) {
```

```
49.         Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
50.         toast.show();
51.     }
52.
53.     /* implement an even handler when a button is pushed */
54.     public void onClick(View v) {
55.
56.         // create an Intent object (we'll fill with data soon)
57.         Intent data = new Intent();
58.
59.         // switch, dependent on button that was pushed
60.         switch(v.getId()) {
61.
62.             case R.id.url:
63.                 // activities require an action to perform and data to act on
64.                 data.setAction(Intent.ACTION_VIEW);
65.                 data.setData(Uri.parse("http://www.cs76.net"));
66.                 break;
67.
68.             case R.id.tel:
69.                 data.setAction(Intent.ACTION_DIAL);
70.                 data.setData(Uri.parse("tel:(234) 567-8901"));
71.                 break;
72.
73.             case R.id.map:
74.                 data.setAction(Intent.ACTION_VIEW);
75.                 data.setData(Uri.parse("geo:0,0?q=el+paso,+tx"));
76.                 break;
77.
78.             case R.id.stview:
79.                 data.setAction(Intent.ACTION_VIEW);
80.                 /* Query fields for the URI:
81.                  * cbll=latitude,longitude
82.                  * cbp= street view window details
83.                  * See: http://mapki.com/index.php?title=Google_Map_Parameters
84.                  */
85.                 data.setData(Uri.parse("google.streetview:cbll=42.379069,-71.116564&cbp=12,60,,0,-1.21&mz=18"));
86.                 break;
87.
88.             case R.id.sms:
89.                 data.setAction(Intent.ACTION_VIEW);
90.                 data.setData(Uri.parse("sms:(234) 567-8901"));
91.                 data.putExtra(Intent.EXTRA_TEXT, "Hello from "+getResources().getString(R.string.app_name));
92.                 break;
93.
94.
95.             case R.id.email:
96.             default:
```

```
 97.                 data.setAction(Intent.ACTION_SEND);
 98.                 data.setType("text/plain");
 99.                 data.putExtra(Intent.EXTRA_EMAIL, new String[] {"danallan@mit.edu"});
100.                 data.putExtra(Intent.EXTRA_SUBJECT, "Hello!");
101.                 data.putExtra(Intent.EXTRA_TEXT, "Message body.\n--Dan");
102.                 break;
103.         }
104.
105.         // try to open the activity
106.         try {
107.             startActivity(data);
108.         } catch (ActivityNotFoundException e) {
109.             // do something if the activity appropriate for that intent/URI is not found
110.             showMessage("Activity not found!");
111.             return;
112.         }
113.
114.         // Alternatively, if we wanted to force the
115.         // user to pick an app to use for the intent:
116.         //startActivity(Intent.createChooser(data, "Send mail..."));
117.
118.         // OR, if you want to force a specific Activity from a specific
119.         // package, specify the ComponentName:
120.         // http://developer.android.com/reference/android/content/ComponentName.html
121.         // setComponent();
122.
123.     }
124. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.intents08;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.intents08;

public final class R {
    public static final class attr {
    }
    public static final class color {
        public static final int dark_grey=0x7f040001;
        public static final int translucent_grey=0x7f040000;
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int img_2851=0x7f020001;
        public static final int img_2944=0x7f020002;
        public static final int img_2989=0x7f020003;
        public static final int img_3005=0x7f020004;
        public static final int img_3012=0x7f020005;
        public static final int img_3034=0x7f020006;
        public static final int img_3047=0x7f020007;
        public static final int img_3092=0x7f020008;
        public static final int img_3110=0x7f020009;
        public static final int img_3113=0x7f02000a;
        public static final int img_3128=0x7f02000b;
        public static final int img_3160=0x7f02000c;
        public static final int img_3226=0x7f02000d;
        public static final int img_3228=0x7f02000e;
        public static final int img_3251=0x7f02000f;
        public static final int img_3268=0x7f020010;
        public static final int img_3275=0x7f020011;
        public static final int img_3346=0x7f020012;
        public static final int img_3365=0x7f020013;
        public static final int img_3374=0x7f020014;
        public static final int img_3385=0x7f020015;
        public static final int img_3392=0x7f020016;
        public static final int img_3397=0x7f020017;
        public static final int img_3398=0x7f020018;
        public static final int img_3403=0x7f020019;
        public static final int img_3424=0x7f02001a;
        public static final int img_3432=0x7f02001b;
        public static final int img_3448=0x7f02001c;
        public static final int img_3452=0x7f02001d;
        public static final int img_3484=0x7f02001e;
```

```
        public static final int img_3487=0x7f02001f;
        public static final int img_3494=0x7f020020;
        public static final int img_3576=0x7f020021;
        public static final int img_3597=0x7f020022;
        public static final int img_3599=0x7f020023;
        public static final int img_3610=0x7f020024;
    }
    public static final class id {
        public static final int RelativeLayout01=0x7f060000;
        public static final int copynotice=0x7f060002;
        public static final int gridview=0x7f060001;
        public static final int single_image=0x7f060003;
    }
    public static final class layout {
        public static final int main=0x7f030000;
        public static final int single_photo=0x7f030001;
    }
    public static final class string {
        public static final int app_name=0x7f050002;
        public static final int author=0x7f050001;
        public static final int photos_display=0x7f050000;
    }
}
```

```java
package net.cs76.lectures.intents08;

import net.cs76.lectures.intents08.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.AdapterView.OnItemClickListener;

/*
 * Intents08
 * Dan Armendariz
 * Computer Science E-76
 *
 * A sample photo gallery application, with one activity implementing
 * a grid view to show a series of images. Upon selection of an image
 * a second activity appears showing a larger version of that photo.
 * Includes an example on custom-building an Adapter, starting a second
 * activity, and passing data via Intents.
 */

public class Code8 extends Activity implements OnItemClickListener {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find our grid and assign our new ImageAdapter
        // class as the adapter for it, along with an
        // onItemClickListener.
        // See: ImageAdapter.java
        GridView grid = (GridView) findViewById(R.id.gridview);

        grid.setAdapter(new ImageAdapter(this));
        grid.setOnItemClickListener(this);
    }

    /* When an item has been clicked we want to show
     * that particular image in a new activity made
     * to just show one large image.
     */
    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {

        // create the Intent to open our ShowImage activity.
```

```
49.         Intent i = new Intent(this, ShowImage.class);
50.
51.         // pass a key:value pair into the 'extra' bundle for
52.         // the intent so the activity is made aware which
53.         // photo was selected.
54.         i.putExtra("imageToDisplay", id);
55.
56.         // start our activity
57.         startActivity(i);
58.     }
59. }
```

```java
package net.cs76.lectures.intents08;

import java.lang.reflect.Field;
import net.cs76.lectures.intents08.R;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter {
    // a list of resource IDs for the images we want to display
    private Integer[] images;

    // a context so we can later create a view within it
    private Context myContext;

    // store a cache of resized bitmaps
    // Note: we're not managing the cache size to ensure it doesn't
    // exceed any maximum memory usage requirements
    private Bitmap[] cache;

    // Constructor
    public ImageAdapter(Context c) {

        myContext = c;

        // Dynamically figure out which images we've imported
        // into the drawable folder, so we don't have to manually
        // type each image in to a fixed array.

        // obtain a list of all of the objects in the R.drawable class
        Field[] list = R.drawable.class.getFields();


        int count = 0, index = 0, j = list.length;

        // We first need to figure out how many of our images we have before
        // we can request the memory for an array of integers to hold their contents.

        // loop over all of the fields in the R.drawable class
        for(int i=0; i < j; i++)
            // if the name starts with img_ then we have one of our images!
            if(list[i].getName().startsWith("img_")) count++;

```

```
49.         // We now know how many images we have. Reserve the memory for an
50.         // array of integers with length 'count' and initialize our cache.
51.         images = new Integer[count];
52.         cache = new Bitmap[count];
53.
54.         // Next, (unsafely) try to get the values of each of those fields
55.         // into the images array.
56.         try {
57.             for(int i=0; i < j; i++)
58.                 if(list[i].getName().startsWith("img_"))
59.                     images[index++] = list[i].getInt(null);
60.         } catch(Exception e) {}
61.         // safer: catch IllegalArgumentException & IllegalAccessException
62.
63.     }
64.
65.     @Override
66.     // the number of items in the adapter
67.     public int getCount() {
68.         return images.length;
69.     }
70.
71.     @Override
72.     // not implemented, but normally would return
73.     // the object at the specified position
74.     public Object getItem(int position) {
75.         return null;
76.     }
77.
78.     @Override
79.     // return the resource ID of the item at the current position
80.     public long getItemId(int position) {
81.         return images[position];
82.     }
83.
84.     // create a new ImageView when requested
85.     @Override
86.     public View getView(int position, View convertView, ViewGroup parent) {
87.
88.         // we've been asked for an ImageView at a specific position. If
89.         // one doesn't already exist (ie, convertView is null) then we must create
90.         // one. Otherwise we can pass it convertView or a recycled view
91.         // that's been passed to us.
92.
93.         ImageView imgView;
94.
95.
96.         if(convertView == null) {
```

```
 97.
 98.             // create a new view
 99.             imgView = new ImageView(myContext);
100.             imgView.setLayoutParams(new GridView.LayoutParams(100,100));
101.
102.         } else {
103.
104.             // recycle an old view (it might have old thumbs in it!)
105.             imgView = (ImageView) convertView;
106.
107.         }
108.
109.         // see if we've stored a resized thumb in cache
110.         if(cache[position] == null) {
111.
112.             // create a new Bitmap that stores a resized
113.             // version of the image we want to display.
114.             BitmapFactory.Options options = new BitmapFactory.Options();
115.             options.inSampleSize = 4;
116.             Bitmap thumb = BitmapFactory.decodeResource(myContext.getResources(), images[position], options);
117.
118.             // store the resized thumb in a cache so we don't have to re-generate it
119.             cache[position] = thumb;
120.         }
121.
122.         // use the resized image we have in the cache
123.         imgView.setImageBitmap(cache[position]);
124.
125.
126.         // We might be tempted to do the below, but this is bad. The
127.         // images we've put in the drawable directory are quite large
128.         // and need to be scaled down to load all of them in memory to
129.         // display on screen. If we just use the raw images (as in the
130.         // below code) we would quickly get an OutOfMemory exception,
131.         // as the entire image would be loaded in memory and scaled
132.         // down live.
133.         //imgView.setImageResource(images[position]);
134.
135.         return imgView;
136.     }
137.
138.
139. }
```

```java
package net.cs76.lectures.intents08;

import net.cs76.lectures.intents08.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;

/* ShowImage.java
 * A new activity that accepts, via the Intent bundle,
 * an ID representing the image to display full-screen
 * to the user.
 */
public class ShowImage extends Activity implements OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.single_photo);

        // find our ImageView in the layout
        ImageView img = (ImageView)findViewById(R.id.single_image);

        // retrieve the set of data passed to us by the Intent
        Bundle extras = getIntent().getExtras();

        // and retrieve the imageToDisplay ID from the extras bundle
        int resource = (int)extras.getLong("imageToDisplay");

        // set the ImageView to display the specified resource ID
        img.setImageResource(resource);

        // close the Activity when a user taps/clicks on the image.
        img.setOnClickListener(this);
    }

    /*
     * finishes (closes) the activity when the user clicks on the image
     */
    public void onClick(View v) {
        finish();
    }
}
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.storage01;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.storage01;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int bmw3=0x7f020000;
        public static final int citation=0x7f020001;
        public static final int icon=0x7f020002;
    }
    public static final class id {
        public static final int button=0x7f070001;
        public static final int hare=0x7f070003;
        public static final int image=0x7f070000;
        public static final int tortoise=0x7f070002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class menu {
        public static final int options_menu=0x7f060000;
    }
    public static final class raw {
        public static final int earlycarengine=0x7f040000;
        public static final int ferrari250=0x7f040001;
    }
    public static final class string {
        public static final int app_name=0x7f050001;
        public static final int hello=0x7f050000;
        public static final int pause=0x7f050003;
        public static final int play=0x7f050002;
    }
}
```

```java
package net.cs76.lectures.storage01;

import net.cs76.lectures.storage01.R;
import android.app.Activity;
import android.content.SharedPreferences;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

/*
 * Code1
 * Dan Armendariz
 * Computer Science E-76
 *
 * Based on Intents05.
 * Demonstrates saving/restoring simple data via
 * the SharedPreferences object.
 */

public class Code1 extends Activity implements OnClickListener, OnCompletionListener {

    MediaPlayer player;
    ImageView image;
    Button vroom;

    // fields related to saving preferences
    int selection;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // find the image and button Views, and set the listener for the button
        image = (ImageView)findViewById(R.id.image);

        vroom = (Button)findViewById(R.id.button);
        vroom.setOnClickListener(this);
    }

    /** called when the Activity is resuming
     */
```

```
49.     public void onResume() {
50.         super.onResume();
51.
52.         // Restore preferences - get the saved preferences
53.         SharedPreferences prefs = getPreferences(MODE_PRIVATE);
54.
55.         // recall the saved car value, or use R.id.hare as the default if
56.         // the preference doesn't (yet?) exist.
57.         int savedCar = prefs.getInt("defaultCar", R.id.hare);
58.
59.         // display the saved car
60.         setCar(savedCar);
61.     }
62.
63.     /** called when the Activity is pausing (and might be killed, so save data)
64.       */
65.     public void onPause() {
66.         // call the parent's onPause() method
67.         super.onPause();
68.
69.         // free up the media player's resources
70.         player.release();
71.         player = null;
72.
73.         // build our preferences object with our data to save
74.         SharedPreferences prefs = getPreferences(MODE_PRIVATE);
75.         SharedPreferences.Editor editor = prefs.edit();
76.         editor.putInt("defaultCar", selection);
77.
78.         // we must commit the preferences or they won't be saved!
79.         editor.commit();
80.     }
81.
82.
83.     /** Initialize the media player with a sound and give the
84.       * ImageView an image, depending on the requested option.
85.       */
86.     public void setCar(int car) {
87.
88.         // remember our current car selection
89.         selection = car;
90.
91.         // set different MediaPlayer or Image options based on the input
92.
93.         // if the player has already been initialized, we should reset it.
94.         // this will stop playing a sound (if one is playing)
95.         if(player != null) {
96.             player.reset();
```

```
97.             vroom.setText(R.string.play);
98.         }
99.
100.        switch(car) {
101.            case R.id.tortoise:
102.                player = MediaPlayer.create(getApplicationContext(), R.raw.earlycarengine);
103.                image.setImageResource(R.drawable.bmw3);
104.                break;
105.            case R.id.hare:
106.            default:
107.                player = MediaPlayer.create(getApplicationContext(), R.raw.ferrari250);
108.                image.setImageResource(R.drawable.citation);
109.        }
110.
111.        // set some options for the player no matter which we picked
112.        player.setLooping(false);
113.        player.setOnCompletionListener(this);
114.
115.    }
116.
117.    /** override the onCreateOptionsMenu that gets
118.      * called when the user requests a menu
119.      */
120.    public boolean onCreateOptionsMenu(Menu menu) {
121.
122.        // can also use the add() method to programmatically create a menu
123.        MenuInflater inflater = getMenuInflater();
124.        inflater.inflate(R.menu.options_menu, menu);
125.        return true;
126.    }
127.
128.
129.    /** When an item is selected in the menu, onOptionsItemSelected
130.      * is called and passes the selected 'item'
131.      */
132.    public boolean onOptionsItemSelected(MenuItem item) {
133.        // change our car based on the selection
134.        setCar(item.getItemId());
135.
136.        return true;
137.    }
138.
139.
140.    /** When the button is clicked we want to hear a sound.
141.      */
142.    public void onClick(View v) {
143.
144.        if(player.isPlaying()) {
```

```
145.             player.pause();
146.             vroom.setText(R.string.play);
147.         } else {
148.             vroom.setText(R.string.pause);
149.             player.start();
150.         }
151.
152.     }
153.
154.     /** reset button text when the sound is done playing
155.       */
156.     public void onCompletion(MediaPlayer mp) {
157.         vroom.setText(R.string.play);
158.     }
159.
160. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.Storage02;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.Storage02;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int cancel=0x7f050004;
        public static final int checkBox=0x7f050003;
        public static final int save=0x7f050005;
        public static final int settings=0x7f050001;
        public static final int textView=0x7f050000;
        public static final int welcomeText=0x7f050002;
    }
    public static final class layout {
        public static final int main=0x7f030000;
        public static final int settings=0x7f030001;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.Storage02;

import net.cs76.lectures.Storage02.R;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

/*
 * Code2
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates saving and restoring data between
 * multiple Activities in the same application via
 * the SharedPreferences object.
 */

public class Code2 extends Activity implements OnClickListener {

    // define a "file name", of sorts, in which to store the preferences
    private final String PREFS_NAME = "Code2Prefs";

    // declare the textview that will show our text
    TextView display;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // connect to the UI object
        display = (TextView)findViewById(R.id.textView);

        // make sure to start the Settings activity when a user
        // clicks on the "Settings" button
        Button settings = (Button)findViewById(R.id.settings);
        settings.setOnClickListener(this);
    }


    /** Restore our preferences.
      * Why might we want to restore onResume(),
      * rather than onCreate()?
```

```
49.       */
50.     public void onResume() {
51.         // as with all overridden activity methods, we should call the
52.         // same method on the parent or risk a crash.
53.         super.onResume();
54.
55.         // get our SharedPreferences object
56.         SharedPreferences prefs = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
57.
58.         // load data from it or use some defaults if values don't (yet?) exist
59.         boolean boldedText = prefs.getBoolean("boldedText", false);
60.         String welcomeText = prefs.getString("welcomeText", getString(R.string.hello));
61.
62.         // we'll set the TextView to bold, but only if the boldedText setting is true
63.         if(boldedText)
64.             display.setTypeface(Typeface.DEFAULT_BOLD);
65.         else
66.             display.setTypeface(Typeface.DEFAULT);
67.
68.         // change the text of the TextView to the saved value
69.         display.setText(welcomeText);
70.     }
71.
72.     /** Open the "Settings" activity on request.
73.      */
74.     public void onClick(View v) {
75.         Intent i = new Intent(this, Settings.class);
76.         startActivity(i);
77.     }
78.
79.
80. }
```

```java
package net.cs76.lectures.Storage02;

import net.cs76.lectures.Storage02.R;
import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;

/*
 * Settings
 * Dan Armendariz
 * Computer Science E-76
 *
 * Allows the user to change some settings for the text that
 * was displayed in the initial Activity, and save them via
 * the SharedPreferences object.
 */

public class Settings extends Activity implements OnClickListener {

    // fields for each UI element
    Button save, cancel;
    CheckBox bold;
    EditText text;

    // define a "file name", of sorts, in which to store the preferences
    private final String PREFS_NAME = "Code2Prefs";


    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings);

        // connect to View objects
        save =   (Button)findViewById(R.id.save);
        cancel = (Button)findViewById(R.id.cancel);
        bold =   (CheckBox)findViewById(R.id.checkBox);
        text =   (EditText)findViewById(R.id.welcomeText);

        // make sure our buttons do something
        save.setOnClickListener(this);
        cancel.setOnClickListener(this);

        // load saved preferences
```

```
49.         SharedPreferences prefs = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
50.
51.         // check the checkbox if the stored preference is true
52.         boolean boldedText = prefs.getBoolean("boldedText", false);
53.         bold.setChecked(boldedText);
54.
55.         // set the EditText view to the current string
56.         String welcomeText = prefs.getString("welcomeText", getString(R.string.hello));
57.         text.setText(welcomeText);
58.
59.     }
60.
61.
62.     /** Save the settings to PREFS_NAME via SharedPreferences
63.      * object when requested.
64.      */
65.     private void savePrefs() {
66.         // get our SharedPreferences object and create an editor for it
67.         SharedPreferences prefs = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
68.         SharedPreferences.Editor editor = prefs.edit();
69.
70.         // place our settings in the object
71.         editor.putBoolean("boldedText", bold.isChecked());
72.         editor.putString("welcomeText", text.getText().toString());
73.
74.         // we must commit the preferences or they won't be saved!
75.         editor.commit();
76.
77.     }
78.
79.
80.     /** Save the data, if requested, and quit this activity
81.       */
82.     public void onClick(View v) {
83.         // save the preferences if the "Save" button was pushed
84.         if(v.getId() == R.id.save) savePrefs();
85.
86.         // in either case, we'll close this activity
87.         finish();
88.     }
89.
90.
91. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.Storage03;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.Storage03;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int auth=0x7f050004;
        public static final int passLabel=0x7f050003;
        public static final int password=0x7f050002;
        public static final int save=0x7f050005;
        public static final int userLabel=0x7f050000;
        public static final int username=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.Storage03;

import net.cs76.lectures.Storage03.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

/*
 * Code3
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrates saving and querying arbitrary data from a
 * SQLite database.
 */

public class Code3 extends Activity implements OnClickListener {

    DBAdapter db;
    EditText user, pass;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // instantiate a DBAdapter
        db = new DBAdapter(this);

        // open a connection to the DB, and create
        // a table if one does not yet exist.
        db.open();

        // connect to UI elements
        Button auth = (Button)findViewById(R.id.auth);
        Button save = (Button)findViewById(R.id.save);
        user = (EditText)findViewById(R.id.username);
        pass = (EditText)findViewById(R.id.password);

        // allow our buttons to do something
        auth.setOnClickListener(this);
        save.setOnClickListener(this);

    }

```

```
49.
50.     /** Explicitly close our database connection when our
51.      * application is done with it and we're about to quit.
52.      */
53.     public void onDestroy() {
54.         super.onDestroy();
55.         db.close();
56.     }
57.
58.
59.     /** Perform the requested button action: Save will add a
60.      * username and password pair to the SQLite table, and
61.      * Authenticate will query the database to see if the
62.      * current user/pass combination are present.
63.      */
64.     public void onClick(View v) {
65.
66.         // find the username and password that were entered
67.         String username = user.getText().toString();
68.         String password = pass.getText().toString();
69.
70.         // if the "Save" button was pushed, we'll save the user/pass into the DB.
71.         // otherwise we'll try to 'authenticate' by verifying the user/pass exist in the db
72.         switch(v.getId()) {
73.             case R.id.save:
74.
75.                 // insertUser() method will insert a user and return a row ID
76.                 long id = db.insertUser(username, password);
77.
78.                 // if the row ID is -1 there was some error, otherwise it was successful
79.                 if (id != -1)
80.                     displayMessage(username + " inserted!");
81.                 else
82.                     displayMessage(username + " wasn't inserted?");
83.
84.                 break;
85.             case R.id.auth: default:
86.
87.                 // attempt to authenticate a user. It will return true
88.                 // if authenticated or false otherwise.
89.                 if(db.authenticateUser(username, password)) {
90.                     displayMessage(username + " authenticated!");
91.                 } else {
92.                     displayMessage("Authentication failed for "+username + "!");
93.                 }
94.         }
95.     }
96.
```

```
 97.
 98.     /** Display a long Toast as feedback for this Activity.
 99.      *
100.      * @param msg is the string to display
101.      */
102.     private void displayMessage(String msg) {
103.         Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
104.     }
105.
106. }
```

```java
package net.cs76.lectures.Storage03;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/*
 * DBAdapter
 * Dan Armendariz
 * Computer Science E-76
 *
 * Provides an interface through which we can perform queries
 * against the SQLite database.
 */

public class DBAdapter {

    // define the layout of our table in fields
    // "_id" is used by Android for Content Providers and should
    // generally be an auto-incrementing key in every table.
    public static final String KEY_ROWID = "_id";
    public static final String KEY_USER = "user";
    public static final String KEY_PASS = "pass";

    // define some SQLite database fields
    // Take a look at your DB on the emulator with:
    //  adb shell
    //  sqlite3 /data/data/<pkg_name>/databases/<DB_NAME>
    private static final String DB_NAME = "db_example";
    private static final String DB_TABLE = "users";
    private static final int    DB_VER = 1;

    // a SQL statement to create a new table
    private static final String DB_CREATE =
        "CREATE TABLE users (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
         "user TEXT NOT NULL, pass TEXT NOT NULL);";


    // define an extension of the SQLiteOpenHelper to handle the
    // creation and upgrade of a table
    private static class DatabaseHelper extends SQLiteOpenHelper {

        // Class constructor
        DatabaseHelper(Context c) {
            // instantiate a SQLiteOpenHelper by passing it
```

```
49.             // the context, the database's name, a CursorFactory
50.             // (null by default), and the database version.
51.             super(c, DB_NAME, null, DB_VER);
52.         }
53.
54.         // called by the parent class when a DB doesn't exist
55.         public void onCreate(SQLiteDatabase db) {
56.             // Execute our DB_CREATE statement
57.             db.execSQL(DB_CREATE);
58.         }
59.
60.         // called by the parent when a DB needs to be upgraded
61.         public void onUpgrade(SQLiteDatabase db, int oldVer, int newVer) {
62.             // remove the old version and create a new one.
63.             // If we were really upgrading we'd try to move data over
64.             db.execSQL("DROP TABLE IF EXISTS "+DB_TABLE);
65.             onCreate(db);
66.         }
67.     }
68.
69.
70.     // useful fields in the class
71.     private final Context context;
72.     private DatabaseHelper helper;
73.     private SQLiteDatabase db;
74.
75.     // DBAdapter class constructor
76.     public DBAdapter(Context c) {
77.         this.context = c;
78.     }
79.
80.     /** Open the DB, or throw a SQLException if we cannot open
81.       * or create a new DB.
82.       */
83.     public DBAdapter open() throws SQLException {
84.         // instantiate a DatabaseHelper class (see above)
85.         helper = new DatabaseHelper(context);
86.
87.         // the SQLiteOpenHelper class (a parent of DatabaseHelper)
88.         // has a "getWritableDatabase" method that returns an
89.         // object of type SQLiteDatabase that represents an open
90.         // connection to the database we've opened (or created).
91.         db = helper.getWritableDatabase();
92.
93.         return this;
94.     }
95.
96.     /** Close the DB
```

```
97.       */
98.      public void close() {
99.          helper.close();
100.     }
101.
102.     /** Insert a user and password into the db
103.       *
104.       * @param user username (string)
105.       * @param pass user's password (string)
106.       * @return the row id, or -1 on failure
107.      */
108.     public long insertUser(String user, String pass) {
109.         ContentValues vals = new ContentValues();
110.         vals.put(KEY_USER, user);
111.         vals.put(KEY_PASS, pass);
112.         return db.insert(DB_TABLE, null, vals);
113.     }
114.
115.     /** Authenticate a user by querying the table to see
116.       * if that user and password exist. We expect only one row
117.       * to be returned if that combination exists, and if so, we
118.       * have successfully authenticated.
119.       *
120.       * @param user username (string)
121.       * @param pass user's password (string)
122.       * @return true if authenticated, false otherwise
123.      */
124.     public boolean authenticateUser(String user, String pass) {
125.         // Perform a database query
126.         Cursor cursor = db.query(
127.                 DB_TABLE, // table to perform the query
128.                 new String[] { KEY_USER }, //resultset columns/fields
129.                 KEY_USER+"=? AND "+KEY_PASS+"=?", //condition or selection
130.                 new String[] { user, pass },  //selection arguments (fills in '?' above)
131.                 null,  //groupBy
132.                 null,  //having
133.                 null   //orderBy
134.             );
135.
136.         // if a Cursor object was returned by the query and
137.         // that query returns exactly 1 row, then we've authenticated
138.         if(cursor != null && cursor.getCount() == 1) {
139.             return true;
140.         }
141.
142.         // The query returned no results or the incorrect
143.         // number of rows
144.         return false;
```

```
145.     }
146.
147. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.Threads01;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.Threads01;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int smiley=0x7f020001;
        public static final int smiley_1=0x7f020002;
        public static final int smiley_2=0x7f020003;
        public static final int smiley_3=0x7f020004;
    }
    public static final class id {
        public static final int anim=0x7f050002;
        public static final int button_runnable=0x7f050004;
        public static final int button_sleep=0x7f050003;
        public static final int check_sleep=0x7f050005;
        public static final int layout=0x7f050000;
        public static final int txt=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.Threads01;

import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.Toast;

/*
 * Code1
 * Dan Armendariz
 * Computer Science E-76
 *
 * Demonstrating Android's single-threaded model,
 * Handler/Runnable adherence to that model, and the
 * dreaded Application Not Responding (ANR) dialog.
 */

public class Code1 extends Activity implements OnClickListener {

    // UI objects
    private Button sleep, runnable;
    private CheckBox moreSleep;
    private ImageView img;

    // animation object
    private AnimationDrawable anim;

    // the amount of time (in seconds) we should sleep when asked
    private static final int S = 10;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // connect to UI objects
        sleep    = (Button)    findViewById(R.id.button_sleep);
        runnable = (Button)    findViewById(R.id.button_runnable);
        img      = (ImageView) findViewById(R.id.anim);
        moreSleep= (CheckBox)  findViewById(R.id.check_sleep);
```

```
49.
50.         // set click listeners
51.         sleep.setOnClickListener(this);
52.         runnable.setOnClickListener(this);
53.         img.setOnClickListener(this);
54.     }
55.
56.     /**
57.      * Display a message to a user via a toast
58.      */
59.     public void showMessage(CharSequence txt) {
60.         (Toast.makeText(this, txt, Toast.LENGTH_SHORT)).show();
61.     }
62.
63.
64.     /**
65.      * Handle onClick events to UI elements.
66.      */
67.     @Override
68.     public void onClick(View v) {
69.         switch(v.getId()) {
70.
71.         case R.id.anim:
72.             // start animating when the image is clicked
73.
74.             // set the image resource to use our animation drawable
75.             img.setImageResource(R.drawable.smiley);
76.
77.             // cast the drawable to an animation and start it
78.             anim = (AnimationDrawable) img.getDrawable();
79.             anim.start();
80.
81.             break;
82.
83.         case R.id.button_sleep:
84.             // sleep for S seconds
85.             SystemClock.sleep(S * 1000);
86.
87.             // Bonus! Try tapping on the screen as soon as the thread starts sleeping.
88.
89.             // more info on time keeping:
90.             // http://developer.android.com/reference/android/os/SystemClock.html
91.
92.             showMessage("Waited " + S + " seconds.");
93.             break;
94.
95.         case R.id.button_runnable:
96.             // create a new handler
```

```
 97.             h = new Handler();
 98.
 99.             // have the handler post the runnable to the message queue in S seconds
100.             h.postDelayed(waitForS, S*1000);
101.
102.             // notify the user that the runnable was posted
103.             showMessage("Runnable posted");
104.             break;
105.         }
106.     }
107.
108.
109.     // define a handler that will post our runnable to the queue
110.     private Handler h;
111.
112.     // the class that contains code to be run
113.     private Runnable waitForS = new Runnable() {
114.
115.         /**
116.          * The method to be run when the queue gets to this runnable
117.          */
118.         public void run() {
119.             // if requested, we'll sleep some additional time
120.             if(moreSleep.isChecked()) SystemClock.sleep(S * 1000);
121.
122.             showMessage("Waited approx. " + S + " seconds.");
123.         }
124.     };
125.
126.
127. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.Threads02;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.Threads02;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int smiley=0x7f020001;
        public static final int smiley_1=0x7f020002;
        public static final int smiley_2=0x7f020003;
        public static final int smiley_3=0x7f020004;
    }
    public static final class id {
        public static final int anim=0x7f050002;
        public static final int button_async=0x7f050003;
        public static final int layout=0x7f050000;
        public static final int txt=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```java
package net.cs76.lectures.Threads02;

import net.cs76.lectures.Threads02.R;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

/*
 * Code2
 * Dan Armendariz
 * Computer Science E-76
 *
 * Fixing ANR issues by creating a background task.
 */

public class Code2 extends Activity implements OnClickListener {

    // UI objects
    private Button async;
    private ImageView img;

    // animation object
    private AnimationDrawable anim;

    // the amount of time (in seconds) we should sleep when asked
    private static final int S = 10;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // connect to UI objects
        async = (Button)    findViewById(R.id.button_async);
        img   = (ImageView) findViewById(R.id.anim);

        // set click listeners
        async.setOnClickListener(this);
        img.setOnClickListener(this);

```

```
49.     }
50.
51.     /**
52.      * Display a message to a user via a toast
53.      */
54.     public void showMessage(CharSequence txt) {
55.         (Toast.makeText(this, txt, Toast.LENGTH_SHORT)).show();
56.     }
57.
58.
59.     /**
60.      * Handle onClick events to UI elements.
61.      */
62.     @Override
63.     public void onClick(View v) {
64.         switch(v.getId()) {
65.
66.         case R.id.anim:
67.             // start animating when the image is clicked
68.
69.             // set the image resource to use our animation drawable
70.             img.setImageResource(R.drawable.smiley);
71.
72.             // cast the drawable to an animation and start it
73.             anim = (AnimationDrawable) img.getDrawable();
74.             anim.start();
75.
76.             break;
77.
78.         case R.id.button_async:
79.
80.             // do our task in the background!
81.             new DoSomeTask().execute();
82.
83.             break;
84.
85.         }
86.     }
87.
88.
89.     // a class that will spawn a background thread to perform a task
90.     // see: http://developer.android.com/reference/android/os/AsyncTask.html
91.     private class DoSomeTask extends AsyncTask<Void, Void, Void> {
92.
93.         // this is the method with the task that will be run in
94.         // the background thread
95.         @Override
96.         protected Void doInBackground(Void... params) {
```

```
 97.             SystemClock.sleep(S*1000);
 98.             return null;
 99.         }
100.
101.         // once the background thread completes, this method
102.         // will be called in the UI thread
103.         protected void onPostExecute(Void params) {
104.             showMessage("Waited " + S + " seconds!");
105.         }
106.     }
107.
108.
109. }
```

```java
/** Automatically generated file. DO NOT MODIFY */
package net.cs76.lectures.Threads03;

public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

```java
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package net.cs76.lectures.Threads03;

public final class R {
    public static final class attr {
    }
    public static final class color {
        public static final int dark_grey=0x7f040001;
        public static final int translucent_grey=0x7f040000;
    }
    public static final class drawable {
        public static final int blank=0x7f020000;
        public static final int icon=0x7f020001;
    }
    public static final class id {
        public static final int RelativeLayout01=0x7f060000;
        public static final int copynotice=0x7f060002;
        public static final int gridview=0x7f060001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f050002;
        public static final int author=0x7f050001;
        public static final int photos_display=0x7f050000;
    }
}
```

```java
package net.cs76.lectures.Threads03;

import net.cs76.lectures.Threads03.URLImageAdapter;
import net.cs76.lectures.Threads03.R;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.TextView;

/*
 * Threads03
 * Dan Armendariz
 * Computer Science E-76
 *
 * A sample photo gallery application, based on loosely on
 * Intents08, that now pulls images from the Internet and
 * supports lazy loading via a background thread.
 */
public class Code3 extends Activity implements OnItemClickListener, OnClickListener {

    private URLImageAdapter adapter;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // get data generated before a config change, if it exists
        final Object data = getLastNonConfigurationInstance();

        // instantiate our adapter, sending in any data
        adapter = new URLImageAdapter(this, data);

        // attach to the grid layout in the UI
        GridView grid = (GridView) findViewById(R.id.gridview);
        grid.setAdapter(adapter);
        grid.setOnItemClickListener(this);

        // enable the little textview at the bottom to open a link
```

```
49.         TextView copy = (TextView) findViewById(R.id.copynotice);
50.         copy.setOnClickListener(this);
51.     }
52.
53.
54.     /**
55.      * Open new activity to show the selected image
56.      * full-screen in a new Activity.
57.      */
58.     public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
59.
60.         openPage((String) adapter.getItem(position));
61.     }
62.
63.
64.     /**
65.      * Open a webpage when the textview is clicked.
66.      */
67.     @Override
68.     public void onClick(View v) {
69.         openPage("http://danallan.net");
70.     }
71.
72.
73.     /**
74.      * Start a browser activity to display the url passed.
75.      */
76.     public void openPage(String url) {
77.         // create an intent
78.         Intent data = new Intent();
79.
80.         // specify the intent's action and url
81.         data.setAction(Intent.ACTION_VIEW);
82.         data.setData(Uri.parse(url));
83.
84.         try {
85.             startActivity(data);
86.         } catch (ActivityNotFoundException e) {
87.             Log.e("Threads03", "Cannot find an activity to start URL " + url);
88.         }
89.
90.     }
91.
92.
93.     /**
94.      * Preserve adapter data between orientation changes
95.      * See: http://developer.android.com/reference/android/app/Activity.html#onRetainNonConfigurationInstance()
96.      */
```

```
 97.     @Override
 98.     public Object onRetainNonConfigurationInstance() {
 99.         return adapter.getData();
100.     }
101. }
```

```java
package net.cs76.lectures.Threads03;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import net.cs76.lectures.Threads03.R;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.ImageView.ScaleType;

public class URLImageAdapter extends BaseAdapter {

    // an object we'll use to keep our cache data together
    private class Image {
        String url;
        Bitmap thumb;
    }

    // an array of resources we want to display
    private Image[] images;

    // a context so we can later create a view within it
    private Context myContext;

    // the background task object
    private LoadThumbsTask thumbnailGen;


    // Constructor
    public URLImageAdapter(Context c, Object previousList) {

        myContext = c;

        // get our thumbnail generation task ready to execute
        thumbnailGen = new LoadThumbsTask();

        // we'll want to use pre-existing data, if it exists
```

```
49.         if(previousList != null) {
50.             images = (Image[]) previousList;
51.
52.             // continue processing remaining thumbs in the background
53.             thumbnailGen.execute(images);
54.
55.             // no more setup required in this constructor
56.             return;
57.         }
58.
59.         // if no pre-existing data, we need to generate it from scratch.
60.
61.         // initialize array
62.         images = new Image[imageURLs.length];
63.
64.         for(int i = 0, j = imageURLs.length; i < j; i++) {
65.             images[i] = new Image();
66.             images[i].url = imageURLs[i];
67.         }
68.
69.         // start the background task to generate thumbs
70.         thumbnailGen.execute(images);
71.     }
72.
73.
74.     @Override
75.     /**
76.      * Getter: number of items in the adapter's data set
77.      */
78.     public int getCount() {
79.         return images.length;
80.     }
81.
82.
83.     @Override
84.     /**
85.      * Getter: return URL at specified position
86.      */
87.     public Object getItem(int position) {
88.         return images[position].url;
89.     }
90.
91.
92.     @Override
93.     /**
94.      * Getter: return resource ID of the item at the current position
95.      */
96.     public long getItemId(int position) {
```

```
 97.         return position;
 98.     }
 99.
100.
101.     /**
102.      * Getter: return generated data
103.      * @return array of Image
104.      */
105.     public Object getData() {
106.         // stop the task if it isn't finished
107.         if(thumbnailGen != null && thumbnailGen.getStatus() != AsyncTask.Status.FINISHED) {
108.             // cancel the task
109.             thumbnailGen.cancel(true);
110.
111.         }
112.
113.         // return generated thumbs
114.         return images;
115.     }
116.
117.
118.     /**
119.      * Create a new ImageView when requested, filling it with a
120.      * thumbnail or a blank image if no thumb is ready yet.
121.      */
122.     @Override
123.     public View getView(int position, View convertView, ViewGroup parent) {
124.
125.         ImageView imgView;
126.
127.         // pull the cached data for the image assigned to this position
128.         Image cached = images[position];
129.
130.         // can we recycle an old view?
131.         if(convertView == null) {
132.
133.             // no view to recycle; create a new view
134.             imgView = new ImageView(myContext);
135.             imgView.setLayoutParams(new GridView.LayoutParams(100,100));
136.
137.         } else {
138.
139.             // recycle an old view (it might have old thumbs in it!)
140.             imgView = (ImageView) convertView;
141.
142.         }
143.
144.         // do we have a thumb stored in cache?
```

```
145.         if(cached.thumb == null) {
146.
147.             // no cached thumb, so let's set the view as blank
148.             imgView.setImageResource(R.drawable.blank);
149.             imgView.setScaleType(ScaleType.CENTER);
150.
151.         } else {
152.
153.             // yes, cached thumb! use that image
154.             imgView.setScaleType(ScaleType.FIT_CENTER);
155.             imgView.setImageBitmap(cached.thumb);
156.
157.         }
158.
159.
160.         return imgView;
161.     }
162.
163.
164.     /**
165.      * Notify the adapter that our data has changed so it can
166.      * refresh the views & display any newly-generated thumbs
167.      */
168.     private void cacheUpdated() {
169.         this.notifyDataSetChanged();
170.     }
171.
172.
173.     /**
174.      * Download and return a thumb specified by url, subsampling
175.      * it to a smaller size.
176.      */
177.     private Bitmap loadThumb(String url) {
178.
179.         // the downloaded thumb (none for now!)
180.         Bitmap thumb = null;
181.
182.         // sub-sampling options
183.         BitmapFactory.Options opts = new BitmapFactory.Options();
184.         opts.inSampleSize = 4;
185.
186.         try {
187.
188.             // open a connection to the URL
189.             // Note: pay attention to permissions in the Manifest file!
190.             URL u = new URL(url);
191.             URLConnection c = u.openConnection();
192.             c.connect();
```

```
193.
194.             // read data
195.             BufferedInputStream stream = new BufferedInputStream(c.getInputStream());
196.
197.             // decode the data, subsampling along the way
198.             thumb = BitmapFactory.decodeStream(stream, null, opts);
199.
200.             // close the stream
201.             stream.close();
202.
203.         } catch (MalformedURLException e) {
204.             Log.e("Threads03", "malformed url: " + url);
205.         } catch (IOException e) {
206.             Log.e("Threads03", "An error has occurred downloading the image: " + url);
207.         }
208.
209.         // return the fetched thumb (or null, if error)
210.         return thumb;
211.     }
212.
213.     // the class that will create a background thread and generate thumbs
214.     private class LoadThumbsTask extends AsyncTask<Image, Void, Void> {
215.
216.         /**
217.          * Generate thumbs for each of the Image objects in the array
218.          * passed to this method. This method is run in a background task.
219.          */
220.         @Override
221.         protected Void doInBackground(Image... cache) {
222.
223.
224.             // define the options for our bitmap subsampling
225.             BitmapFactory.Options opts = new BitmapFactory.Options();
226.             opts.inSampleSize = 4;
227.
228.             // iterate over all images ...
229.             for (Image i : cache) {
230.
231.                 // if our task has been cancelled then let's stop processing
232.                 if(isCancelled()) return null;
233.
234.                 // skip a thumb if it's already been generated
235.                 if(i.thumb != null) continue;
236.
237.                 // artificially cause latency!
238.                 SystemClock.sleep(500);
239.
240.                 // download and generate a thumb for this image
```

```
241.                 i.thumb = loadThumb(i.url);
242.
243.                 // some unit of work has been completed, update the UI
244.                 publishProgress();
245.             }
246.
247.             return null;
248.         }
249.
250.
251.         /**
252.          * Update the UI thread when requested by publishProgress()
253.          */
254.         @Override
255.         protected void onProgressUpdate(Void... param) {
256.             cacheUpdated();
257.         }
258.     }
259.
260.
261.     // all images from: http://danallan.net
262.     private String[] imageURLs = {
263.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_2851.jpg",
264.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_2944.jpg",
265.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_2989.jpg",
266.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3005.jpg",
267.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3012.jpg",
268.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3034.jpg",
269.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3047.jpg",
270.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3092.jpg",
271.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3110.jpg",
272.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3113.jpg",
273.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3128.jpg",
274.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3160.jpg",
275.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3226.jpg",
276.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3228.jpg",
277.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3251.jpg",
278.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3268.jpg",
279.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3275.jpg",
280.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3346.jpg",
281.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3365.jpg",
282.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3374.jpg",
283.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3385.jpg",
284.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3392.jpg",
285.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3397.jpg",
286.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3398.jpg",
287.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3403.jpg",
288.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3424.jpg",
```

```
289.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3432.jpg",
290.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3448.jpg",
291.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3452.jpg",
292.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3484.jpg",
293.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3487.jpg",
294.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3494.jpg",
295.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3576.jpg",
296.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3597.jpg",
297.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3599.jpg",
298.             "http://cdn.cs76.net/2011/spring/lectures/6/imgs/img_3610.jpg"
299.     };
300. }
```