```
        }

        @Override
        public void onFinish() {
            progressDialog.dismiss();
        }
    }.start();
```

After the three seconds have elapsed, you dismiss the dialog by calling the `dismiss()` method.

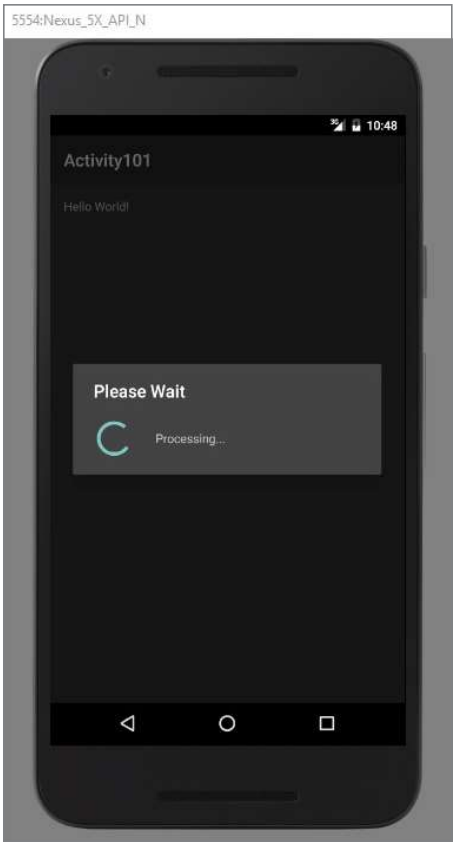

FIGURE 3-7

The next section explains using Intents, which help you navigate between multiple Activities.

LINKING ACTIVITIES USING INTENTS

An Android application can contain zero or more activities. When your application has more than one activity, you often need to navigate from one to another. In Android, you navigate between activities through what is known as an intent.

The best way to understand this very important but somewhat abstract concept is to experience it firsthand and see what it helps you achieve. The following Try It Out shows how to add another activity to an existing project and then navigate between the two activities.

TRY IT OUT Linking Activities with Intents (UsingIntent.zip)

1. Using Android Studio, create a new Android project with an empty Activity named `MainActivity`; name the project **UsingIntent**.
2. Right-click your package name under the `app>>app>>src>>main>>java` folder in the Project Files windows and select New ➪ Java Class
3. Name the new class **SecondActivity** and click OK.
4. Add the bolded statements from the following code to the `AndroidManifest.xml` file. Be sure to change all instances of `"com.jfdimarzio"` to whatever package name your project is using.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.jfdimarzio.usingintent">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity" >
         <intent-filter >
             <action android:name="com.jfdimarzio.usingintent.SecondActivity" />
             <category android:name="android.intent.category.DEFAULT" />
         </intent-filter>
     </activity>
    </application>

</manifest>
```

5. Make a copy of the `activity_main.xml` file (in the `res/layout` folder) by right-clicking it and selecting Copy. Then right-click the `res/layout` folder and select Paste. Name the file activity_second.xml.
6. Modify the `activity_second.xml` file as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.usingintent.SecondActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the Second Activity!" />
</RelativeLayout>
```

7. In the `SecondActivity.java` file, add the bolded statements from the following code:

```
package com.jfdimarzio.usingintent;

import android.app.Activity;
import android.os.Bundle;

public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }

}
```

8. Add the bolded lines in the following code to the `activity_main.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.usingintent.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activity!"
        android:id="@+id/textView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Display second activity"
        android:onClick="onClick"
        android:id="@+id/button"
        android:layout_below="@+id/textView"
        android:layout_alignParentStart="true"
        android:layout_marginTop="56dp" />
</RelativeLayout>
```

9. Modify the `MainActivity.java` file as shown in the bolded lines in the following code:

```
package com.jfdimarzio.usingintent;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        startActivity(new Intent("com.jfdimarzio.usingintent.SecondActivity"));
    }

}
```

10. Press Shift+F9 to debug the application on the Android emulator. When the first activity is loaded, click the button and the second activity also loads (see Figures 3-8 and 3-9).

How It Works

As previously described, an activity is made up of a UI component (for example, `activity_main.xml`) and a class component (for example, `MainActivity.java`). If you want to add another activity to a project, you need to create these two components.

Specifically, you need to add the following to the `AndroidManifest.xml` file:

```
</activity>
        <activity android:name=".SecondActivity" >
        <intent-filter >
            <action android:name="com.jfdimarzio.usingintent.SecondActivity" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
</activity>
```

When you add a new activity to the application, be sure to note the following:

- The name (class) of the new activity is `SecondActivity`.
- The intent filter name for the new activity is `<Your Package Name>.SecondActivity`. Other activities that want to call this activity invoke it via this name. Ideally, you should use the reverse domain name of your company as the intent filter name to reduce the chances of another application having the same intent filter name.
- The category for the intent filter is `android.intent.category.DEFAULT`. You need to add this to the intent filter so that this activity can be started by another activity using the `startActivity()` method (more on this shortly).

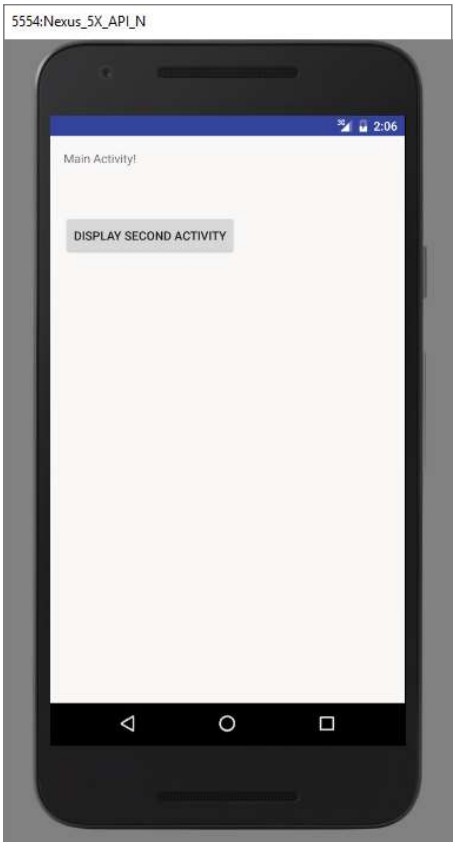

FIGURE 3-8

FIGURE 3-9

When the Display Second Activity button is clicked, you use the `startActivity()` method to display `SecondActivity` by creating an instance of the `Intent` class and passing it the intent filter name of `SecondActivity` (`net.learn2develop.SecondActivity`):

```
public void onClick(View view) {
    startActivity(new Intent("net.learn2develop.SecondActivity"));
}
```

Activities in Android can be invoked by any application running on the device. For example, you can create a new Android project and then display `SecondActivity` by using its `net.learn2develop.SecondActivity` intent filter. This is one of the fundamental concepts in Android that enables an application to easily invoke another application.

If the activity you want to invoke is defined within the same project, you can rewrite the preceding statement like this:

```
startActivity(new Intent(this, SecondActivity.class));
```

However, this approach is applicable only when the activity you want to display is within the same project as the current activity.

Returning Results from an Intent

The `startActivity()` method invokes another activity but does not return a result to the current activity. For example, you might have an activity that prompts the user for username and password. The information entered by the user in that activity needs to be passed back to the calling activity for further processing. If you need to pass data back from an activity, you should instead use the `startActivityForResult()` method. The following Try It Out demonstrates this.

TRY IT OUT Obtaining a Result from an Activity

1. Using the same project from the previous section, modify the `secondactivity.xml` file to look like the following code. Please be sure to change all references from `"com.jfdimarzio"` to whatever package name your project is using:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.usingintent.SecondActivity">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the Second Activity!"
        android:id="@+id/textView2" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Please enter your name"
        android:id="@+id/textView3" />

    <EditText
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/txtUsername" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK"
        android:onClick="onClick"
        android:id="@+id/button2" />
</LinearLayout>
```

2. Add the bolded statements in the following code to `SecondActivity.java`:

```
package com.jfdimarzio.usingintent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
    public void onClick(View view) {
         Intent data = new Intent();
         //---get the EditText view---
         EditText txt_username = (EditText)findViewById(R.id.txtUsername);
         //---set the data to pass back---
         data.setData(Uri.parse( txt_username.getText().toString()));
         setResult(RESULT_OK, data);
         //---closes the activity---
         finish();
    }
}
```

3. Add the bolded statements in the following code to the `MainActivity.java` file:

```
package com.jfdimarzio.usingintent;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {
    int request_Code = 1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        startActivityForResult(new Intent("com.jfdimarzio.usingintent.
          SecondActivity"),request_Code);
    }
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == request_Code) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(this,data.getData().toString(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    }

}
```

4. Press Shift+F9 to debug the application on the Android emulator. When the first activity is loaded, click the button to load `SecondActivity`. Enter your name (see Figures 3-10, 3-11, and 3-12) and click the OK button. The first activity displays the name you have entered using the `Toast` class.

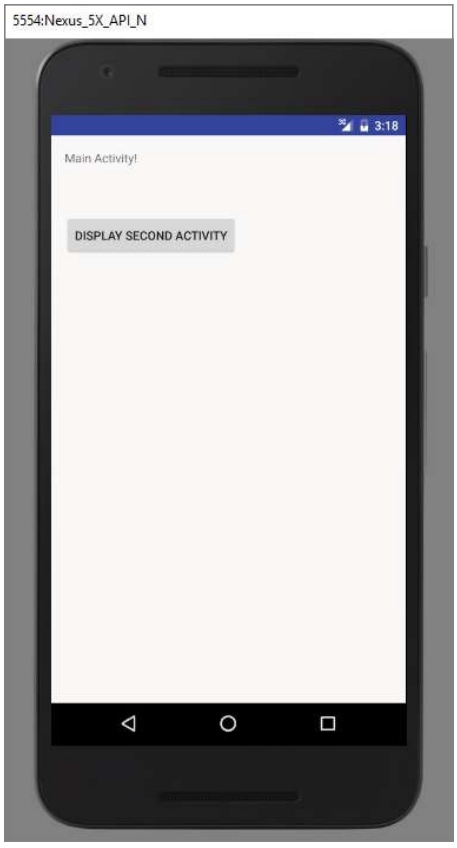

FIGURE 3-10

How It Works

To call an activity and wait for a result to be returned from it, you need to use the `startActivityForResult()` method, like this:

```
startActivityForResult(new Intent("com.jfdimarzio.usingintent.
  SecondActivity"),request_Code);
```

In addition to passing in an `Intent` object, you need to pass in a request code as well. The request code is simply an integer value that identifies an activity you are calling. This is needed because when an activity returns a value, you must have a way to identify it. For example, you might be calling multiple activities at the same time, though some activities might not return immediately (for example, waiting for a reply from a server). When an activity returns, you need this request code to determine which activity is actually returned.

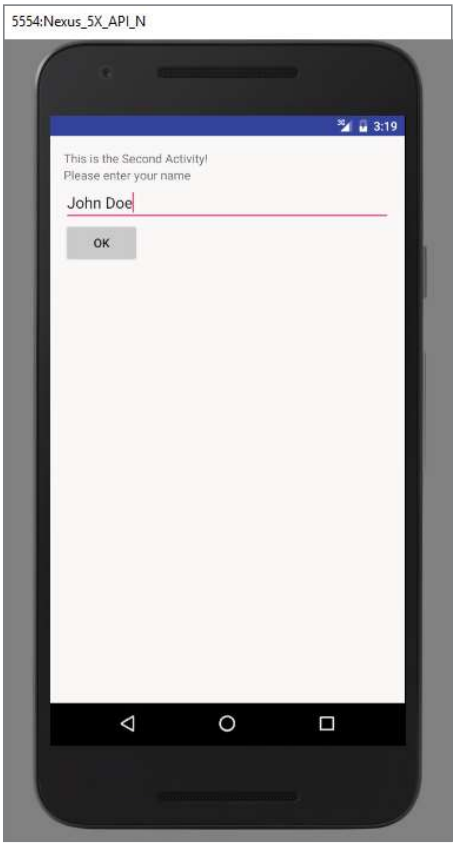

FIGURE 3-11

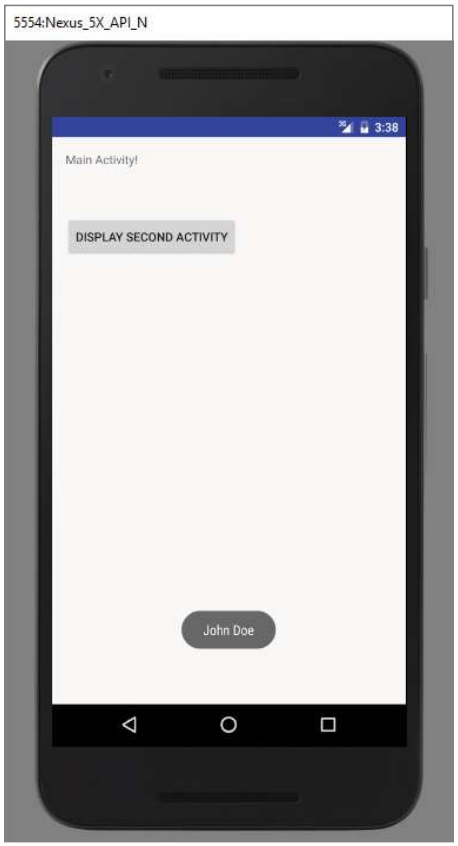

FIGURE 3-12

> ***NOTE*** *If the request code is set to –1 then calling it using the* `startActivityForResult()` *method is equivalent to calling it using the* `startActivity()` *method. That is, no result is returned.*

In order for an activity to return a value to the calling activity, you use an `Intent` object to send data back via the `setData()` method:

```
Intent data = new Intent();
//---get the EditText view---
EditText txt_username =
    (EditText) findViewById(R.id.txt_username);
//---set the data to pass back---
data.setData(Uri.parse(
    txt_username.getText().toString()));
setResult(RESULT_OK, data);
//---closes the activity---
finish();
```

The `setResult()` method sets a result code (either `RESULT_OK` or `RESULT_CANCELLED`) and the data (an `Intent` object) to be returned back to the calling activity. The `finish()` method closes the activity and returns control to the calling activity.

In the calling activity, you need to implement the `onActivityResult()` method, which is called whenever an activity returns:

```
public void onActivityResult(int requestCode, int resultCode,
Intent data)
{
    if (requestCode == request_Code) {
        if (resultCode == RESULT_OK) {
            Toast.makeText(this,data.getData().toString(),
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Here, you check for the appropriate request and result codes and display the result that is returned. The returned result is passed in via the `data` argument; and you obtain its details through the `getData()` method.

Passing Data Using an Intent Object

Besides returning data from an activity, it is also common to pass data to an activity. For example, in the previous example, you might want to set some default text in the `EditText` view before the activity is displayed. In this case, you can use the `Intent` object to pass the data to the target activity.

The following Try It Out shows you the various ways in which you can pass data between activities.

TRY IT OUT Passing Data to the Target Activity

1. Using Eclipse, create a new Android project and name it **PassingData**.
2. Add the bolded statements in the following code to the `activity_main.xml` file. Be sure to change all instances of `"com.jfdimarzio"` to whatever package name your project is using.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.passingdata.MainActivity">


    <Button
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Click to go to Second Activity"
         android:id="@+id/button"
         android:onClick="onClick"/>
</LinearLayout>
```

3. Add a new XML file to the `res/layout` folder and name it **activity_second.xml**. Populate it as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.passingdata.MainActivity">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to the Second Activity"
        android:id="@+id/textView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click to go to Main Activity"
        android:id="@+id/button"
        android:onClick="onClick"/>
</LinearLayout>
```

4. Add a new `Class` file to the package and name it **SecondActivity**. Populate the `SecondActivity.java` file as follows:

```
package com.jfdimarzio.passingdata;
```

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class SecondActvity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        //---get the data passed in using getStringExtra()---
        Toast.makeText(this,getIntent().getStringExtra("str1"),
                Toast.LENGTH_SHORT).show();
        //---get the data passed in using getIntExtra()---
        Toast.makeText(this,Integer.toString(
                getIntent().getIntExtra("age1", 0)),
                Toast.LENGTH_SHORT).show();
        //---get the Bundle object passed in---
        Bundle bundle = getIntent().getExtras();
        //---get the data using the getString()---
        Toast.makeText(this, bundle.getString("str2"),
                Toast.LENGTH_SHORT).show();
        //---get the data using the getInt() method---
        Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
                Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        //---use an Intent object to return data---
        Intent i = new Intent();
        //---use the putExtra() method to return some
        // value---
        i.putExtra("age3", 45);
        //---use the setData() method to return some value---
        i.setData(Uri.parse("Something passed back to main activity"));
        //---set the result with OK and the Intent object---
        setResult(RESULT_OK, i);
        //---destroy the current activity---
        finish();
    }

}
```

5. Add the bolded statements from the following code to the `AndroidManifest.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.jfdimarzio.passingdata">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
```

```
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActvity" >
         <intent-filter >
             <action android:name="com.jfdimarzio.passingdata.SecondActivity" />
              <category android:name="android.intent.category.DEFAULT" />
         </intent-filter>

    </activity>
    </application>

</manifest>
```

6. Add the bolded statements from the following code to the `MainActivity.java` file:

```
package com.jfdimarzio.passingdata;

import android.content.Intent;
import android.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
         Intent i = new
                 Intent("com.jfdimarzio.passingdata.SecondActivity");
         //---use putExtra() to add new name/value pairs---
         i.putExtra("str1", "This is a string");
         i.putExtra("age1", 25);
         //---use a Bundle object to add new name/values
         // pairs---
         Bundle extras = new Bundle();
         extras.putString("str2", "This is another string");
         extras.putInt("age2", 35);
         //---attach the Bundle object to the Intent object---
         i.putExtras(extras);
         //---start the activity to get a result back---
         startActivityForResult(i, 1);
    }

    public void onActivityResult(int requestCode,
                                 int resultCode, Intent data)
```

```
        {
            //---check if the request code is 1---
            if (requestCode == 1) {
                //---if the result is OK---
                if (resultCode == RESULT_OK) {
                    //---get the result using getIntExtra()---
                    Toast.makeText(this, Integer.toString(
                            data.getIntExtra("age3", 0)),
                            Toast.LENGTH_SHORT).show();
                    //---get the result using getData()---
                    Toast.makeText(this, data.getData().toString(),
                            Toast.LENGTH_SHORT).show();
                }
            }
        }

    }
```

7. Press Shift+F9 to debug the application on the Android emulator. Click the button on each activity and observe the values displayed.

How It Works

While this application is not visually exciting, it does illustrate some important ways to pass data between activities.

First, you can use the `putExtra()` method of an `Intent` object to add a name/value pair:

```
//---use putExtra() to add new name/value pairs---
i.putExtra("str1", "This is a string");
i.putExtra("age1", 25);
```

The preceding statements add two name/value pairs to the `Intent` object: one of type `string` and one of type `integer`.

Besides using the `putExtra()` method, you can also create a `Bundle` object and then attach it using the `putExtras()` method. Think of a `Bundle` object as a dictionary object—it contains a set of name/value pairs. The following statements create a `Bundle` object and then add two name/value pairs to it. The `Bundle` object is then attached to the `Intent` object:

```
//---use a Bundle object to add new name/values pairs---
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);
//---attach the Bundle object to the Intent object---
i.putExtras(extras);
```

To obtain the data sent using the `Intent` object, you first obtain the `Intent` object using the `getIntent()` method. Then, call its `getStringExtra()` method to get the string value set using the `putExtra()` method:

```
//---get the data passed in using getStringExtra()---
Toast.makeText(this,getIntent().getStringExtra("str1"),
        Toast.LENGTH_SHORT).show();
```

In this case, you have to call the appropriate method to extract the name/value pair based on the type of data set. For the integer value, use the `getIntExtra()` method (the second argument is the default value in case no value is stored in the specified name):

```
//---get the data passed in using getIntExtra()---
Toast.makeText(this,Integer.toString(
        getIntent().getIntExtra("age1", 0)),
        Toast.LENGTH_SHORT).show();
```

To retrieve the `Bundle` object, use the `getExtras()` method:

```
//---get the Bundle object passed in---
Bundle bundle = getIntent().getExtras();
```

To get the individual name/value pairs, use the appropriate method. For the string value, use the `getString()` method:

```
//---get the data using the getString()---
Toast.makeText(this, bundle.getString("str2"),
        Toast.LENGTH_SHORT).show();
```

Likewise, use the `getInt()` method to retrieve an integer value:

```
//---get the data using the getInt() method---
Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
        Toast.LENGTH_SHORT).show();
```

Another way to pass data to an activity is to use the `setData()` method (as used in the previous section), like this:

```
//---use the setData() method to return some value---
i.setData(Uri.parse(
        "Something passed back to main activity"));
```

Usually, you use the `setData()` method to set the data on which an `Intent` object is going to operate, such as passing a URL to an `Intent` object so that it can invoke a web browser to view a web page. (For more examples, see the section "Calling Built-In Applications Using Intents," later in this chapter.)

To retrieve the data set using the `setData()` method, use the `getData()` method (in this example `data` is an `Intent` object):

```
//---get the result using getData()---
Toast.makeText(this, data.getData().toString(),
    Toast.LENGTH_SHORT).show();
```

FRAGMENTS

In the previous section, you learned what an activity is and how to use it. In a small-screen device (such as a smartphone), an activity typically fills the entire screen, displaying the various views that make up the user interface of an application. The activity is essentially a container for views. However, when an activity is displayed in a large-screen device, such as on a tablet, it is somewhat