Android Boot Camp for Developers Using Java, 3E

Chapter 4: Explore!
Icons and Decision-Making Controls

Objectives

In this chapter, you learn to:

- Create an Android project with a custom icon
- Change the text color in controls using hexadecimal colors
- Align controls using the gravity properties
- Determine layout with the layout:margin properties
- Place a RadioGroup and RadioButtons in Android applications
- Write code for a RadioGroup control

Objectives (continued)

- Make decisions using an If statement
- Make decisions using an If Else statement
- Make decisions using logical operators
- Display an Android toast notification
- Test the isChecked property
- Make decisions using nested if statements

The Medical Calculator App

- We will be creating an app to convert pounds to kilograms and kilograms to pounds
  - Formulas needed:
    - Kilograms = pounds * 2.2
    - Pounds = kilograms / 2.2
- App is designed to be used in a hospital setting to administer medication to patients based on patient weight
  - Hospital scales register pounds
  - Meds (based on patient weight) dispensed in kilograms

The Medical Calculator App (continued)

Figure 4-1 Opening screen of the Medical Calculator

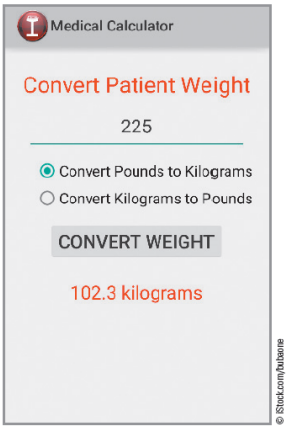

Figure 4-2 Results screen of the Medical Calculator

The Medical Calculator App (continued)

Steps to complete the App

1. Create a customized launcher icon.
2. Add the icon using code to display in the ActionBar.
3. Define a TextField for the data entry of the weight of the patient.
4. Define a RadioGroup to select pounds to kilograms or kilograms to pounds.
5. Display a Toast message for data validation.
6. Convert data so it can be used for arithmetic operations.
7. Perform arithmetic operations on data the user enters.
8. Display formatted results.

The Launcher Icon

- **The Launcher Icon** allows you to view which apps are available
  - High-quality launcher icons can influence users to purchase your app
  - Icons can establish brand identity
  - Simple images with clear visual cues have a memorable impact
  - Icon dimensions are 72 X 72 pixels for the high-density screen
  - Vector graphics as best for icon design because images are easily resized

Figure 4-4 Launcher icon for the Medical Calculator app

The Launcher Icon (continued)

Resolution	Dots per Inch (dpi)	Size (px)
ldpi (low-density screen)	120	36 × 36
mdpi (medium-density screen)	160	48 × 48
hdpi (high-density screen)	240	72 × 72
xhdpi (extra high-density screen)*	320	96 × 96
xxhdpi (extra extra high-density screen)*	440	144 × 144

Table 4-1 Launcher icon sizes

* Used by some tablets

- When you publish an app to the Android Market, you must provide a 512 × 512 pixel, high-resolution application icon in the developer console as you upload your program. This icon is displayed in the Android Market to provide a description of the app and does not replace your launcher icon.

The Launcher Icon (continued)

Customizing a Launcher Icon

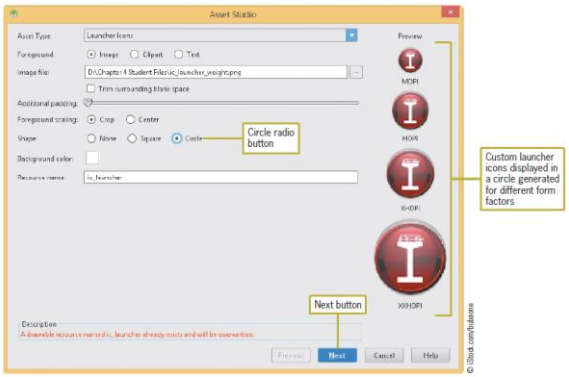

Figure 4-8 Custom launcher icons

The Launcher Icon (continued)

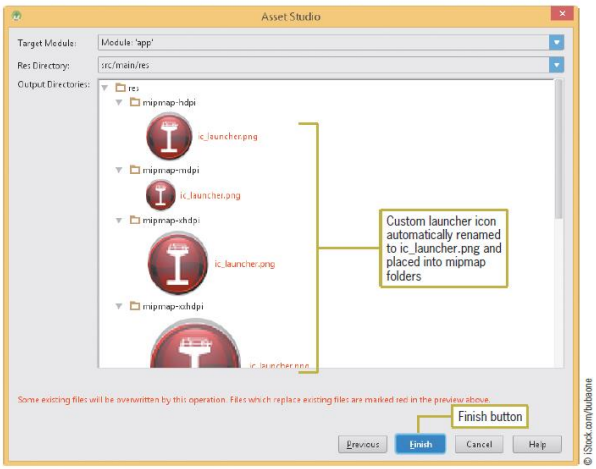

Figure 4-9 Custom icons displayed in res/mipmap folders

The Launcher Icon (continued)

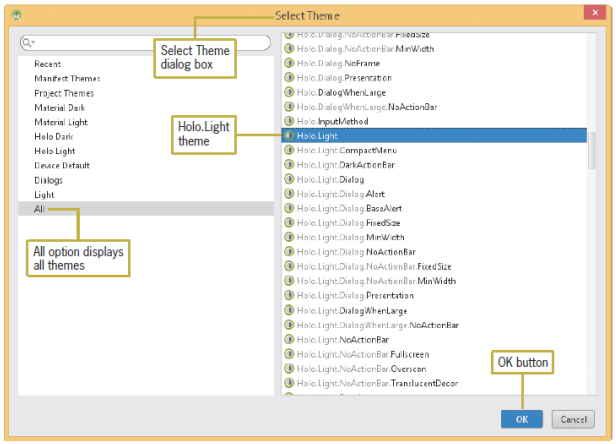

Figure 4-10 Selecting a theme for the application

The Launcher Icon (continued)

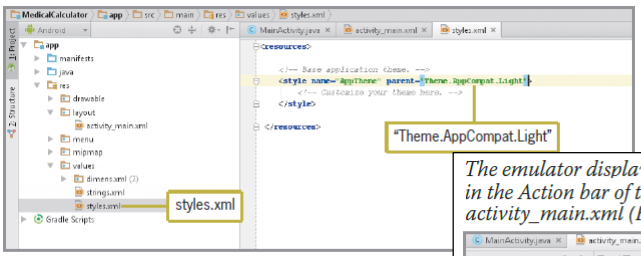

Figure 4-11 styles.xml with updated theme

The theme is updated in styles.xml to display a white background and a gray title bar when the application is executed (Figure 4-11). If you execute the app at this point, the icon launcher appears on the Home screen, but does not appear in the Action bar.

An Action bar icon as shown in Figure 4-12 is considered a **logo** that represents what the program's function is in a single glance; for example, the medical scale conveys the purpose and identity of the app.

The emulator displays the icon launcher in the Action bar of the Holo.Light theme in activity_main.xml (Figure 4-12).

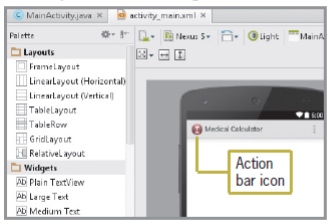

Figure 4-12 Action bar displays icon launcher in activity_main.xml

Difference between holo.light and appcompat.light

Holo.Light

- Can only be used with Activity class
- Not recommended
- Cannot support backward compatibility
- Appearance looks slightly different
- Used in older devices (<API 11)

Appcompat.Light

- Can only be used with AppCompatActivity class
- Recommended
- Can support backward and forward compatibility
- Appearance looks slightly different
- Backward compatibility ensures that either newer or older theme can be applied

Displaying the Action Bar Icon using Code

MainActivity.java has three new statements to display the logo named ic_launcher that was placed in the mipmap folder (Figure 4-13). Line 16 displays the logo image of the scale near the line number.

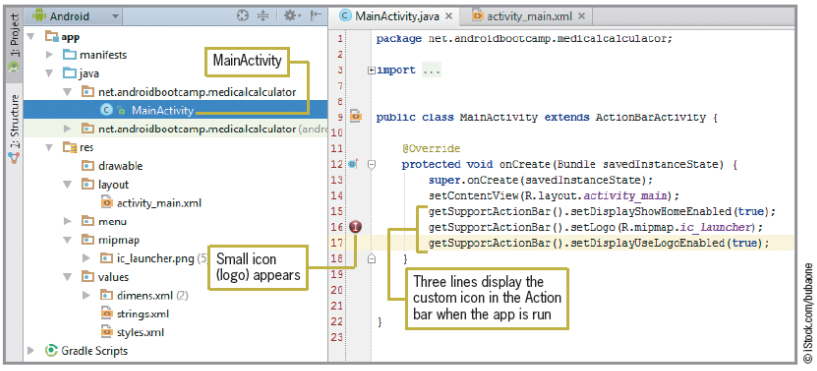

Figure 4-13 Code to display logo in finished app

String Table

- String resources are stored within the /res/values/strings.xml file
- Any strings you add to the strings.xml file are accessible within your application

String Name	String Value
hint	Weight of Patient
radLbToKilo	Convert Pounds to Kilograms
radKiloToLb	Convert Kilograms to Pounds
btnConvert	Convert Weight

Table 4-2 String names and values to add

String Table (continued)

The strings.xml file is populated with the text used in the app (Figure 4-14).

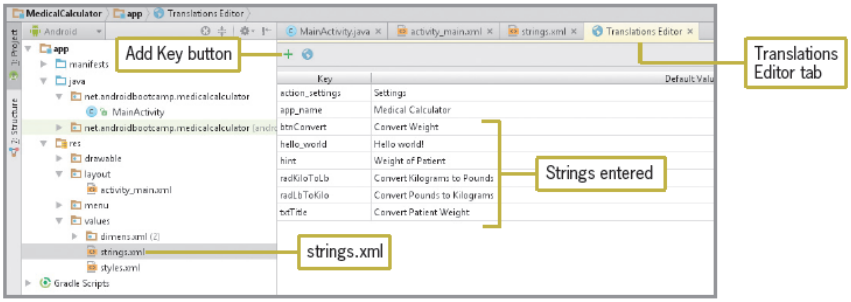

Figure 4-14 Translations Editor with new strings

RadioButton and RadioGroup Controls

- A **RadioButton** control selects or deselects an option
  - Can be arranged horizontally or vertically
  - Have a label defined by the text property
  - Can be initially set to checked or unchecked
  - Typically used together in a **RadioGroup**
    - Only one RadioButton in the group can be selected at a time
  - Good to offer a default selection (checked = true) for the option that is used most

Changing the Text Color of Android Controls

- Use **hexadecimal color codes** to represent RGB (Red, Green, Blue) values
- Codes range from 00 to FF (00 = none, FF = full)
- Codes are identified by a pound sign, followed by the RGB values
  - #FF0000 is all RED
  - #00FF00 is all GREEN
  - #0000FF is all BLUE
  - #FFFF00 is YELLOW (RED and GREEN = YELLOW)

Changing the Text Color of Android Controls

(continued)

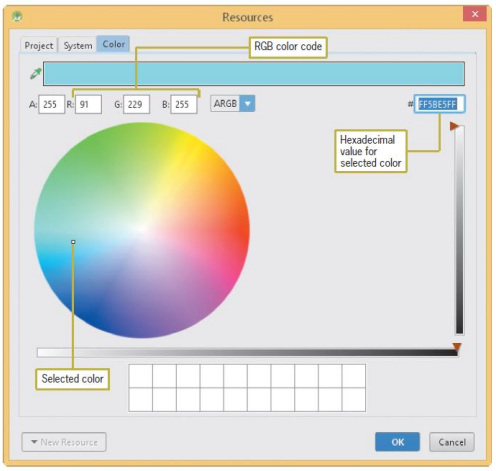

Figure 4-15 Color tab in the Resources dialog box

Changing Margins and Layout Gravity

- **Changing the Margins**
  - **Layout:margin** allows for more flexibility in controlling your layout
  - Set independent pixel values instead of "eyeballing" to create equal spaces around controls
  - Using the same specified margins creates a symmetrical layout
- **Changing the Layout Gravity**
  - Linear layout is the default setting on the emulator for older android

    OS but constraint layout is default in new android OS
  - The **Gravity** tool changes the alignment
    - Works like the left, center, right, top or bottom buttons on the Microsoft Office ribbon

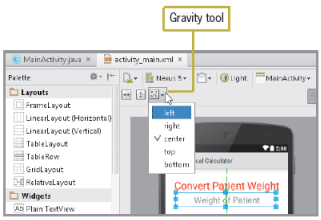

Figure 4-16 Gravity tool options

Adding the RadioButton Group

- Use the prefix rad to name the control

The layout:margin top property is displayed with 15dp entered. The Plain TextView control is added to the form with the text, size, and text color changed (Figure 4-17).

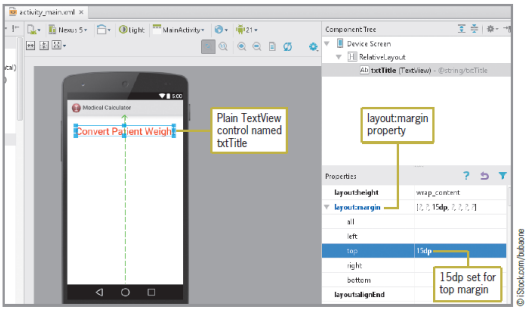

Figure 4-17 Plain TextView control and layout:margin property

Adding the RadioButton Group (continued)

A Number Text Field control is placed on the emulator with the id, text size, text hint, gravity, and margins changed (Figure 4-18).

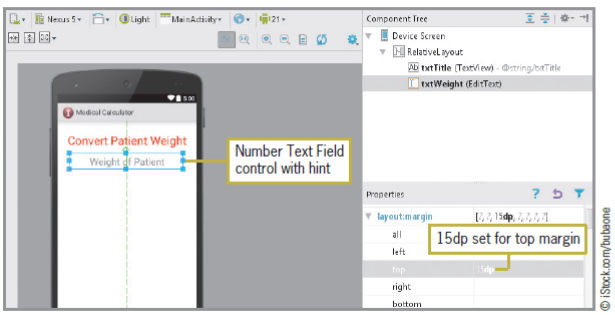

Figure 4-18 Number Text Field control

Adding the RadioButton Group (continued)

The RadioGroup object is placed on the emulator with the id, text, size, and margin properties changed (Figure 4-19).

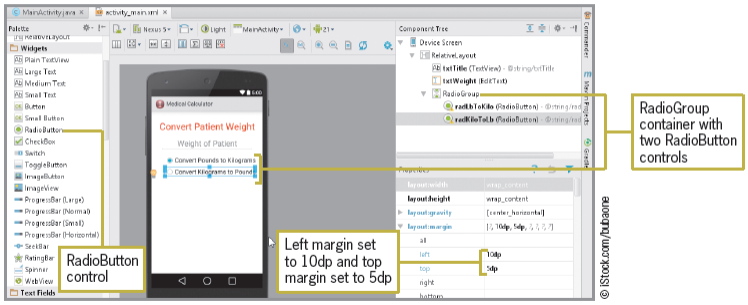

Figure 4-19 RadioGroup control with two RadioButton controls

Completing the User Interface

Figure 4-20 Button control

Figure 4-21 Plain TextView control to display the results

Coding a RadioButton Control

```
final RadioButton lbsToKilo = (RadioButton) findViewById(R.id.radLbToKilo);
final RadioButton kiloToLbs = (RadioButton) findViewById(R.id.radKiloToLb);
```

Three variables are declared in the Java code (Figure 4-22).

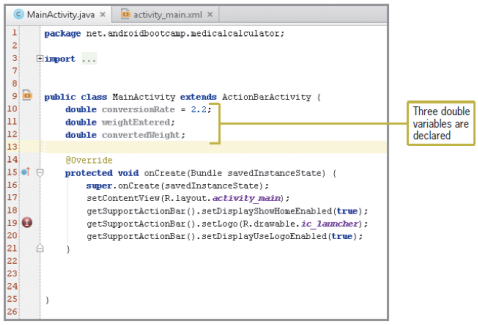

Figure 4-22 Variables declared

Coding a RadioButton Control (continued)

The EditText class extracts the value from the user's input for the patient weight and the RadioButton class extracts the checked value from the radio buttons (Figure 4-23).

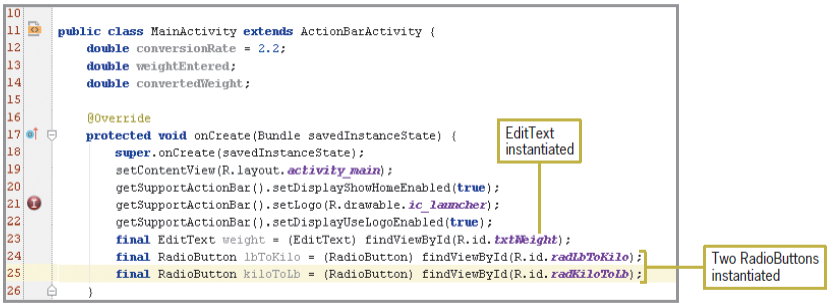

Figure 4-23 EditText and RadioButtons referenced

Coding the Button Control

```
12  public class MainActivity extends ActionBarActivity {
13      double conversionRate = 2.2;
14      double weightEntered;
15      double convertedWeight;
16
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21          getSupportActionBar().setDisplayShowHomeEnabled(true);
22          getSupportActionBar().setLogo(R.drawable.ic_launcher);
23          getSupportActionBar().setDisplayUseLogoEnabled(true);
24          final EditText weight = (EditText) findViewById(R.id.txtWeight);
25          final RadioButton lbToKilo = (RadioButton) findViewById(R.id.radLbToKilo);
26          final RadioButton kiloToLb = (RadioButton) findViewById(R.id.radKiloToLb);
27          final TextView result = (TextView) findViewById(R.id.txtResult);
28      }
```

TextView instantiated and assigned to the variable result

Figure 4-24 TextView control referenced

Coding the Button Control (continued)

```
14  public class MainActivity extends ActionBarActivity {
15      double conversionRate = 2.2;
16      double weightEntered;
17      double convertedWeight;
18
19      @Override
20      protected void onCreate(Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22          setContentView(R.layout.activity_main);
23          getSupportActionBar().setDisplayShowHomeEnabled(true);
24          getSupportActionBar().setLogo(R.drawable.ic_launcher);
25          getSupportActionBar().setDisplayUseLogoEnabled(true);
26          final EditText weight = (EditText) findViewById(R.id.txtWeight);
27          final RadioButton lbToKilo = (RadioButton) findViewById(R.id.radLbToKilo);
28          final RadioButton kiloToLb = (RadioButton) findViewById(R.id.radKiloToLb);
29          final TextView result = (TextView) findViewById(R.id.txtResult);
30          Button convert = (Button) findViewById(R.id.btnConvert);
31
32          convert.setOnClickListener(new View.OnClickListener() {
33              @Override
34              public void onClick(View v) {
35
36              }
37          });
38      }
```

Button instantiated

OnClickListener

Figure 4-25 OnClickListener for the Button control

Making Decisions with Conditional Statements

Decision structures are used to test conditions

- Using an **If Statement**

```
if (condition) {
     //Statements completed if true
}
```

  - Statements between the opening and closing braces are executed if the condition is true

Using If Else Statements

Code Syntax

```
if (weightEntered <= 500) {
     convertedWeight = weightEntered / conversionRate;
} else {
     //Statements completed if condition is false
}
```

  - One set of statements are executed if the condition is true and a different set of statements are executed if the condition is false

Relational Operators

Java strings are compared with the **equals method** (==) of the string class

Relational Operator	Meaning	Example	Resulting Condition
= =	Equal to	6 = = 6	True
! =	Not equal to	4! = 7	False
>	Greater than	3 > 2	True
<	Less than	8 < 1	False
>=	Greater than or equal to	5 >= 5	True
<=	Less than or equal to	9 <= 6	False

Table 4-3 Relational operators

Relational Operators (continued)

```
String name1 = "Sara";
String name2 = "Shawna";
String name3 = "Ryan";
```

If Statement	Comparison	Resulting Condition
if (name1.equals(name2))	Strings are not equal	False
if (name1.compareTo(name1) = = 0)	Strings are equal	True
if (name1.compareTo(name3) = = 0)	Strings are not equal	False
if (name1.compareTo(name2) > 0)	The first string precedes the second string; returns a negative number	False
if (name1.compareTo(name3) < 0)	The first string follows the third string; returns a negative number	True
If (name3.compareTo(name2) > 0)	The first string follows the second string; returns a positive number	True
If (name3.compareTo(name2) > 0)	The first string follows the second string; returns a positive number	True

Table 4-4 Examples of the equals and compareTo methods

Logical Operators

When more than one condition is tested the conditions are called a **compound condition**

Logical Operator	Meaning	Example
&&	And—all conditions must be true	if (flight < 400 && hotel < 120)
||	Or—at least one condition must be true	if (stamp < 0.49 11 rate = = 2)
!	Not—reverses the meaning of a condition	if (! (grade > 70))

Table 4-5 Common logical operators

Data Validation and Toast Notifications

- **Data Validation**
  - User entries must be checked for reasonable values
- **Toast Notification**
  - A **toast notification** communicates messages to the user (message slides upward into view like toast popping out of a toaster)

Code Syntax

```
Toast toast = Toast.makeText(context, text, duration).show( );
```

Code Syntax

```
Toast.makeText(MainActivity.this,"Pounds must be less than 500",
Toast.LENGTH_LONG).show( );
```

Using the isChecked() Method of RadioButton Controls

- The **isChecked() method** determines if the RadioButton object has been selected

Code Syntax

```
if (lbToKilo.isChecked) {
     //Statements completed if condition is true
} else {
     //Statements completed if condition is false
}
```


Coding the Button Event

The syntax **Double.parseDouble** converts input to a Double data type and Integer.parseInt converts input to an Integer data type

The weight entered by the user is converted to a Double data type (Figure 4-26).

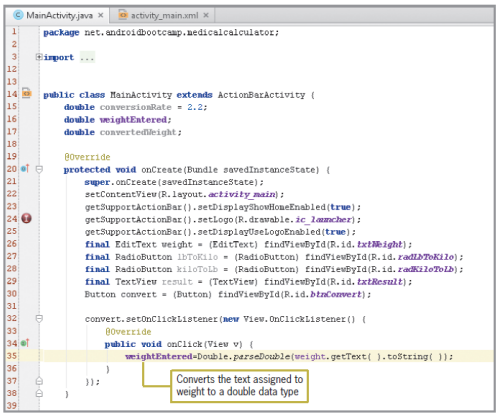

Figure 4-26 Weight converted to a double data type

Coding the Button Event (continued)

The DecimalFormat code rounds off to the nearest tenth (Figure 4-27).

```
34          convert.setOnClickListener(new View.OnClickListener() {
35              @Override
36              public void onClick(View v) {
37                  weightEntered=Double.parseDouble(weight.getText( ).toString( ));
38                  DecimalFormat tenth = new DecimalFormat("#.#");
39              }
40          });
41      }
```

DecimalFormat rounds to one place past the decimal point

Figure 4-27 DecimalFormat—Rounding to one decimal place

Coding the Nested If Statements

- If statements are **nested** when one if statement is inside of another if statement

An If statement determines if the lbToKilo RadioButton control is checked (Figure 4-28).

```
33
34          convert.setOnClickListener(new View.OnClickListener() {
35              @Override
36              public void onClick(View v) {
37                  weightEntered=Double.parseDouble(weight.getText( ).toString( ));
38                  DecimalFormat tenth = new DecimalFormat("#.#");
39                  if(lbToKilo.isChecked( )) {
40
41                  }
42                  }
43          });
44      }
45
46
47
48  }
49
```

If statement determines if the first RadioButton is checked

Figure 4-28 If statement to test if the first radio button is checked

Coding the Nested If Statements (continued)

A nested If Else statement determines if the number of pounds entered is valid (Figure 4-29).

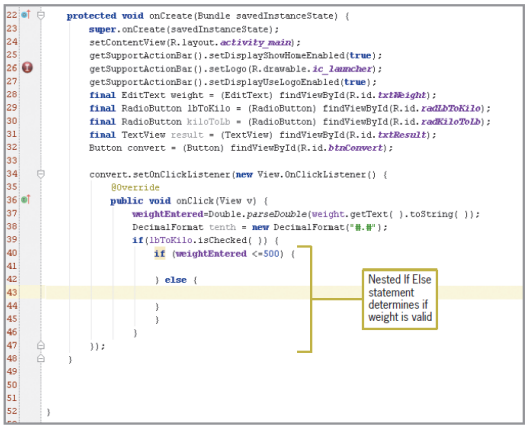

Figure 4-29 Nested If Else statement

Coding the Nested If Statements (continued)

The number of pounds is converted to kilograms and displayed in the result Plain TextView control (Figure 4-30).

Figure 4-30 Equation for weight conversion and displayed results

Coding the Nested If Statements (continued)

A toast message displays a reminder to enter a valid weight (Figure 4-31).

```
34
35          convert.setOnClickListener(new View.OnClickListener() {
36              @Override
37              public void onClick(View v) {
38                  weightEntered=Double.parseDouble(weight.getText( ).toString( ));
39                  DecimalFormat tenth = new DecimalFormat("#.#");
40                  if(lbToKilo.isChecked( )) {
41                      if (weightEntered <=500) {
42                          convertedWeight = weightEntered / conversionRate;
43                          result.setText(tenth.format(convertedWeight) + " kilograms");
44                      } else {
45                          Toast.makeText(MainActivity.this,"Pounds must be less than 500", Toast.LENGTH_LONG).show( );
46                      }
47                      }
48                  }
49          });
50      }
51
52
53
54  }
```

Toast message

Figure 4-31 Toast message

Coding the Nested If Statements (continued)

The nested If statement is executed if the second RadioButton control is selected (Figure 4-32).

MainActivity.java × activity_main.xml ×

```
1   package net.androidbootcamp.medicalcalculator;
2
3   import ...
15
16
17  public class MainActivity extends ActionBarActivity {
18      double conversionRate = 2.2;
19      double weightEntered;
20      double convertedWeight;
21
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26          getSupportActionBar().setDisplayShowHomeEnabled(true);
27          getSupportActionBar().setLogo(R.drawable.ic_launcher);
28          getSupportActionBar().setDisplayUseLogoEnabled(true);
29          final EditText weight = (EditText) findViewById(R.id.txtWeight);
30          final RadioButton lbToKilo = (RadioButton) findViewById(R.id.radLbToKilo);
31          final RadioButton kiloToLb = (RadioButton) findViewById(R.id.radKiloToLb);
32          final TextView result = (TextView) findViewById(R.id.txtResult);
33          Button convert = (Button) findViewById(R.id.btnConvert);
34
35          convert.setOnClickListener(new View.OnClickListener() {
36              @Override
37              public void onClick(View v) {
38                  weightEntered = Double.parseDouble(weight.getText().toString());
39                  DecimalFormat tenth = new DecimalFormat("#.#");
40                  if (lbToKilo.isChecked()) {
41                      if (weightEntered <= 500) {
42                          convertedWeight = weightEntered / conversionRate;
43                          result.setText(tenth.format(convertedWeight) + " kilograms");
44                      } else {
```

Figure 4-32 Completed code *(continues)*

Coding the Nested If Statements (continued)

(continued)

```
45                      Toast.makeText(MainActivity.this, "Pounds must be less than 500", Toast.LENGTH_LONG).show();
46                  }
47              }
48              if (kiloToLb.isChecked()) {
49                  if (weightEntered <= 225) {
50                      convertedWeight = weightEntered * conversionRate;
51                      result.setText(tench.format(convertedWeight) + " pounds");
52                  } else {
53                      Toast.makeText(MainActivity.this, "Kilos must be less than 225", Toast.LENGTH_LONG).show();
54                  }
55              }
56          }
57      ));
58  }
59
60
61
62  }
63
```

Figure 4-32 Completed code

Summary

- To display a custom launcher icon instead of the default icon on the home screen of an Android device, tap or click Image Asset on the New menu to open the Asset Studio dialog box
- Include RadioButton controls to allow users to select or deselect options – only one button can be selected at a time
- Android apps use hexadecimal color codes
- Use the layout:margin property to change the spacing between objects

Summary (continued)

- If statements execute statements if a condition is true
- If Else statements execute one group of statements if a condition is true and different group of statements if the condition is false
- Relational operators are used within the conditional statement
- Compound conditions must use logical operators such as && (And)

Summary (continued)

- Toast notifications display a brief message to a user
- Use nested If statements to test a second condition only after determining that a first condition is true or false

Practice Demo

- Currency Conversion App
- The opening screen requests the amount of U.S. dollars to be converted.
- The user selects euros, Mexican pesos, or Canadian dollars.
- The conversion of U.S. dollars to the selected currency is displayed.
- The program only converts values below $100,000 U.S. dollars.
- Use a customized launcher icon – use licenced for reuse images from google.
- Use http://xe.com to locate current conversion rates.