Android Boot Camp for Developers Using Java, 3E

Chapter 3: Engage! Android User Input, Variables, and Operations

Objectives

In this chapter, you learn to:

- Use an Android theme
- Add a theme to the Android Manifest file
- Add text to the String table
- Develop a user interface using Text Fields
- Describe the role of different Text Fields
- Display a hint using the Hint property
- Develop the user interface using a Spinner control
- Add a prompt to a Spinner control

Objectives

- Declare variables to hold data
- Code the GetText() method
- Understand arithmetic operations
- Convert numeric data
- Format numeric data
- Code the SetText() method
- Run the completed app in an emulator

Android Themes

- Engaging the user by requesting input customizes the user experience each time the application is executed
- A **theme** is a style applied to an Activity or an entire application
  - Themes are Android's mechanism for applying a consistent style to an app or Activity
- The default for Nexus 5 shows the title bar displaying the app name with a white background when running the app

Previewing a Theme

- Check the activity_main.xml file in the emulator to see what your screen looks like

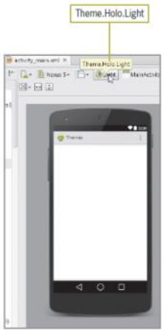


Figure 3-3 Holographic theme

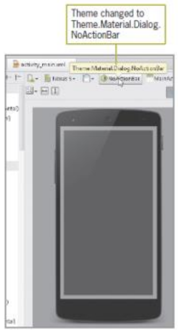


Figure 3-4 Material theme with no action bar

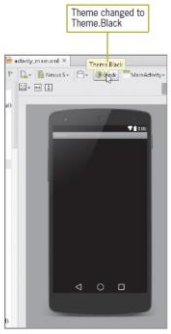
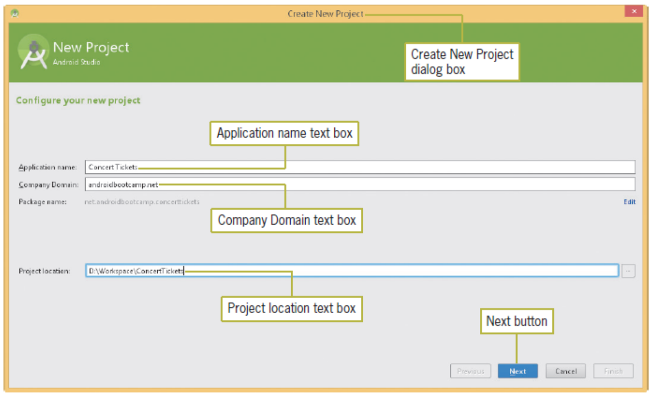


Figure 3-5 Black theme

Previewing a Theme (continued)

A new application named Concert Tickets is configured to be saved on the D:\Workspace USB drive (Figure 3-6).



Configure your new project

Application name: Concert Tickets
Company Domain: androidbootcamp.net
Package name: net.androidbootcamp.concerttickets
Project location: D:\Workspace\ConcertTickets

Figure 3-6 Configuring the Concert Tickets project

Previewing a Theme (continued)

The activity_main.xml file is displayed on the Design tab and the Hello world TextView widget is deleted (Figure 3-7). The emulator displays the default AppTheme.

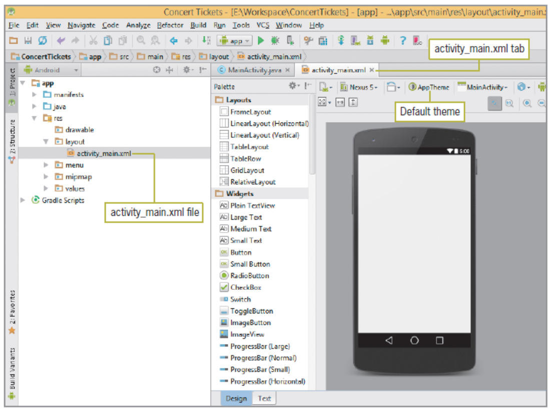


Figure 3-7 activity_main.xml for the Concert Tickets project

Previewing a Theme (continued)

A list of themes with the text "black" in the name is displayed in the Select Theme dialog box (Figure 3-8).

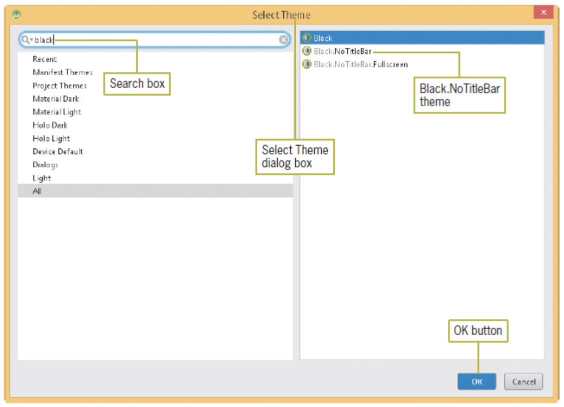


Figure 3-8 Select Theme dialog box

Previewing a Theme (continued)

The theme changes to Black.NoTitleBar. Android Studio removes the title bar in the emulator and displays the background as black (Figure 3-9).

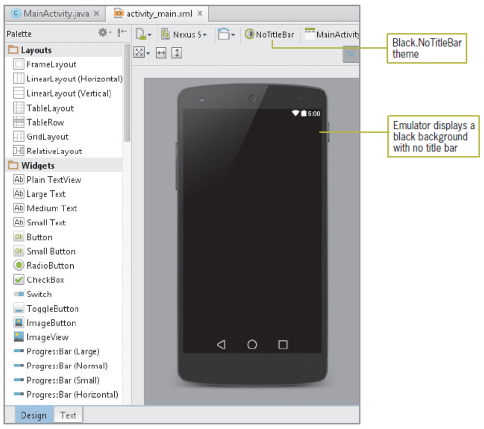
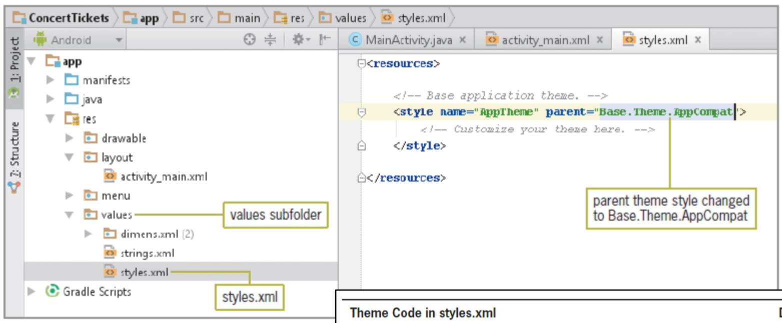


Figure 3-9 New theme applied

Updating the Theme in the styles.xml File

The Android theme is updated within the Activity in the styles.xml file (Figure 3-10).



```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Base.Theme.AppCompat">
        <!-- Customize your theme here. -->
    </style>

</resources>
```

Figure 3-10 Updated theme coded in the styles.xml file

Theme Code in styles.xml	Description
<style name="AppTheme" parent="Base.Theme.AppCompat">	Black background, grey title bar
<style name="AppTheme" parent=" Base.Theme.AppCompat.Light ">	White background, grey title bar
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">	White background, no title bar

Table 3-1 XML code for common themes

Simplifying User Input

- The onscreen keyboard is called a **soft keyboard**
  - Input can be in the form of tapping or gestures (using two fingers to pan, rotate, or zoom)
  - Primary design challenge is to simplify user experiences
  - Use legible fonts, simplify input, and optimize each device's capabilities to maximize user experience

Simplifying User Input

Using Android Text Fields

- Text Fields are the most common type of mobile input
  - Can be free-form plain text
  - Numbers
  - A person's name, password, email, phone number
  - A date
  - Multiline text

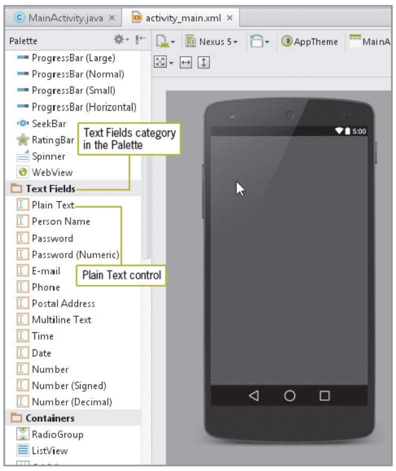


Figure 3-11 Text Fields category

Simplifying User Input

Adding a Text Field

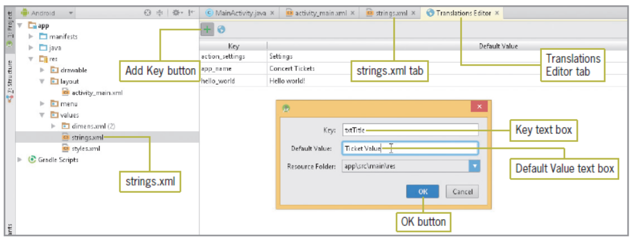
- Use the Id property in the Properties pane to enter a name that begins with the prefix txt
- Use descriptive names like txtTickets instead of txtText1

Using the String Table

- Uses the file strings.xml
- A **string** is a series of alphanumeric characters that can include spaces
- **Localization** is the use of the String table to change text based on the user's preferred language

Simplifying User Input

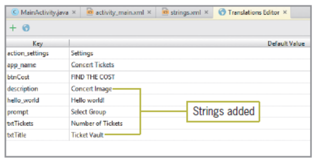
The Key and Default Value of the TextView control are entered into the strings.xml Translations Editor within a dialog box (Figure 3-12).



Key	Default Value
action_settings	Settings
app_name	Concert Tickets
hello_world	Hello world!

Key: txtTitle
Default Value: Ticket Value
Resource Folder: app\src\main\res

Figure 3-12 Adding strings



Key	Default Value
action_settings	Settings
app_name	Concert Tickets
btnCost	FIND THE COST
description	Concert Image
hello_world	Hello world!
prompt	Select Group
txtTickets	Number of Tickets
txtTitle	Ticket Vault

Figure 3-13 Multiple strings added to the Translations Editor

Key	Default Value
txtTickets	Number of Tickets
prompt	Select Group
description	Concert Image
btnCost	FIND THE COST

Table 3-2 String table text

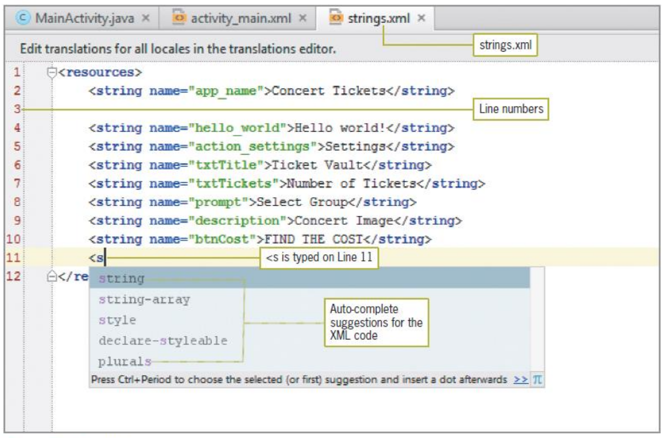
Simplifying User Input

Adding a String Array

- A **string array** defines a string resource of related items in a central location within strings.xml
- An **item** defines an individual entry within a string array
  - As you type the String array XML code, the Android Studio editor offers suggestions in a panel that can complete the statement

Simplifying User Input (continued)

Auto-completion suggestions are displayed in the strings.xml code window (Figure 3-14).


MainActivity.java
activity_main.xml
strings.xml
strings.xml
Edit translations for all locales in the translations editor.
Line numbers
<s is typed on Line 11
Auto-complete suggestions for the XML code

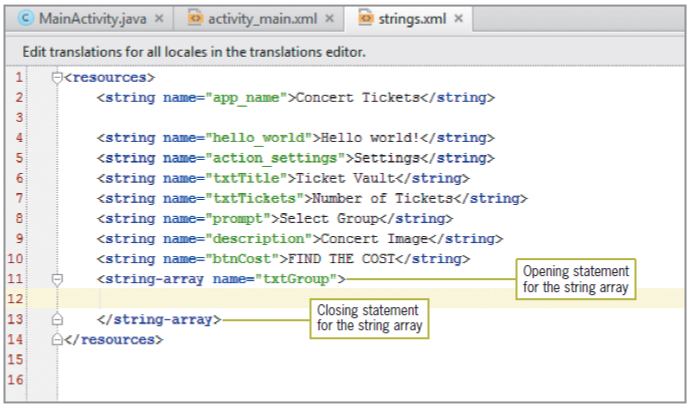
```
1   <resources>
2       <string name="app_name">Concert Tickets</string>
3
4       <string name="hello_world">Hello world!</string>
5       <string name="action_settings">Settings</string>
6       <string name="txtTitle">Ticket Vault</string>
7       <string name="txtTickets">Number of Tickets</string>
8       <string name="prompt">Select Group</string>
9       <string name="description">Concert Image</string>
10      <string name="btnCost">FIND THE COST</string>
11      <s
12  </re
```

```
string
string-array
style
declare-styleable
plurals
Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >> π
```

Figure 3-14 XML code auto-completion suggestions

Simplifying User Input (continued)

The string-array XML code named txtGroup is added. The closing statement </string-array> is added automatically by the editor in Android Studio (Figure 3-15).

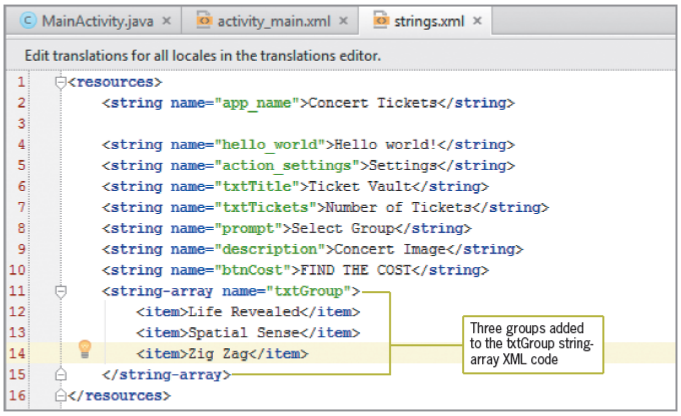

MainActivity.java
activity_main.xml
strings.xml
Edit translations for all locales in the translations editor.
Opening statement for the string array
Closing statement for the string array

```
<resources>
    <string name="app_name">Concert Tickets</string>

    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="txtTitle">Ticket Vault</string>
    <string name="txtTickets">Number of Tickets</string>
    <string name="prompt">Select Group</string>
    <string name="description">Concert Image</string>
    <string name="btnCost">FIND THE COST</string>
    <string-array name="txtGroup">

    </string-array>
</resources>
```

Figure 3-15 Opening and closing XML statements of the string array

Simplifying User Input (continued)

Three items are added to the txtGroup String array (Figure 3-16).


MainActivity.java
activity_main.xml
strings.xml
Edit translations for all locales in the translations editor.
Three groups added to the txtGroup string-array XML code

```
<resources>
    <string name="app_name">Concert Tickets</string>

    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="txtTitle">Ticket Vault</string>
    <string name="txtTickets">Number of Tickets</string>
    <string name="prompt">Select Group</string>
    <string name="description">Concert Image</string>
    <string name="btnCost">FIND THE COST</string>
    <string-array name="txtGroup">
        <item>Life Revealed</item>
        <item>Spatial Sense</item>
        <item>Zig Zag</item>
    </string-array>
</resources>
```

Figure 3-16 string-array with three items added

Simplifying User Input (continued)

The text resource txtTitle from the strings.xml file is selected and the text Ticket Vault is displayed in the size of 48 scalable pixels (Figure 3-17).

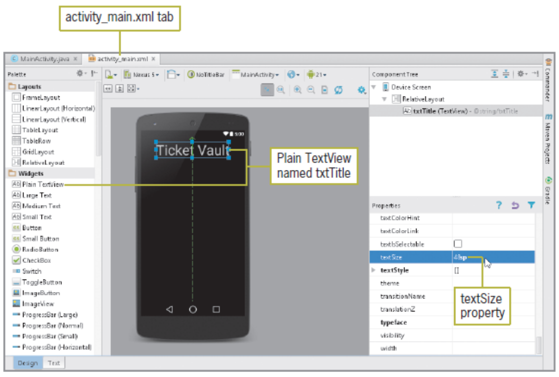


Figure 3-17 Title added to emulator

Simplifying User Input (continued)

A Number Text Field control named txtTickets with the size of 36sp is added to the emulator to allow the user to enter the number of tickets (Figure 3-18).

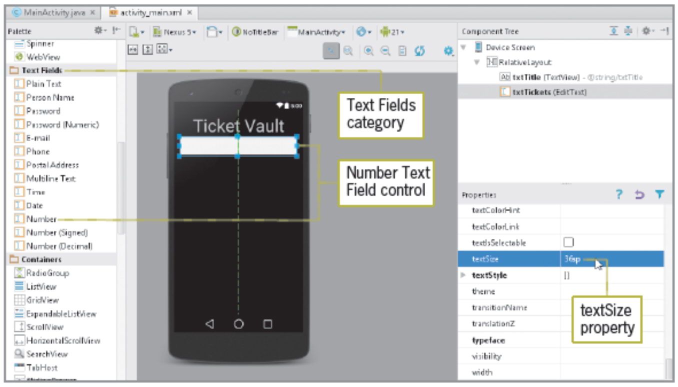


Figure 3-18 String Array

Simplifying User Input (continued)

Setting the Hint Property for the Text Field

- A **hint** is a short description of a field visible as light-colored text (called a watermark)

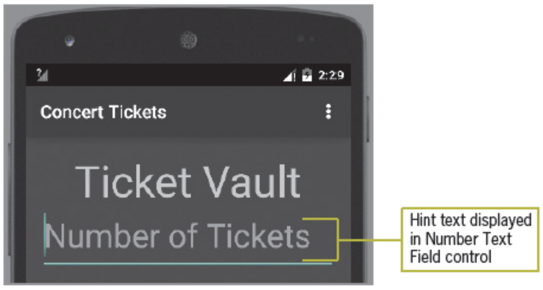
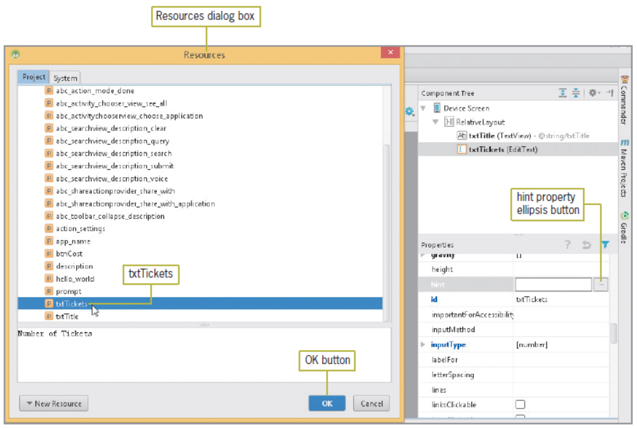


Figure 3-19 Hint in Text Field control

Simplifying User Input (continued)

The hint text txtTickets is selected in the Resources dialog box (Figure 3-20).



```
abc_action_mode_done
abc_activity_chooser_view_see_all
abc_activitychooserview_choose_application
abc_searchview_description_clear
abc_searchview_description_query
abc_searchview_description_search
abc_searchview_description_submit
abc_searchview_description_voice
abc_shareactionprovider_share_with
abc_shareactionprovider_share_with_application
abc_toolbar_collapse_description
action_settings
app_name
btnCost
description
hello_world
prompt
txtTickets
txtTitle

Number of Tickets
```

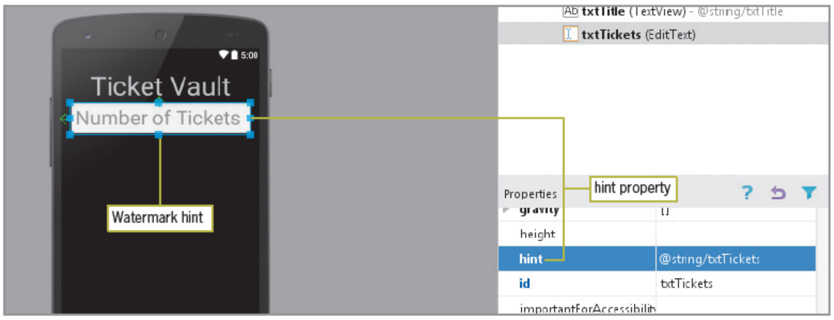
```
Device Screen
  RelativeLayout
    txtTitle (TextView) - @string/txtTitle
    txtTickets (EditText)
```

gravity	[]
height	
hint	
id	txtTickets
importantForAccessibility	
inputMethod	
inputType	[number]
labelFor	
letterSpacing	
lines	
linksClickable	☐

Figure 3-20 Resources dialog box for hint property of the Text Field control

Simplifying User Input (continued)

A watermark hint requests the number of tickets as input in the Text Field control (Figure 3-21).



gravity	[]
height	
hint	@string/txtTickets
id	txtTickets
importantForAccessibility	

Figure 3-21 Hint added to Text Field control

Simplifying User Input (continued)

Using the Android Spinner Control

- A **Spinner control** is similar to a drop-down list for selecting a single item from a fixed list
- The spinner control displays a list of strings called **items** in a pop-up window
- A **prompt**, which can be used to display instructions at the top of the Spinner control, also is stored in strings.xml and is named prompt
- The Spinner property called **Entries** connects the String Array to the Spinner control for display in the application

Simplifying User Input (continued)

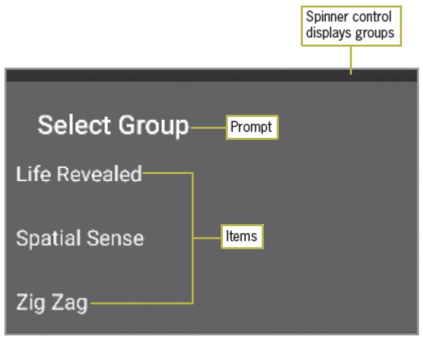


Figure 3-22 Spinner control

The spinnerMode property displays the options for the Spinner control named txtGroup (Figure 3-23).

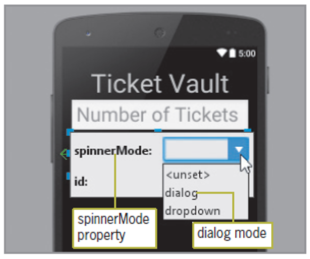
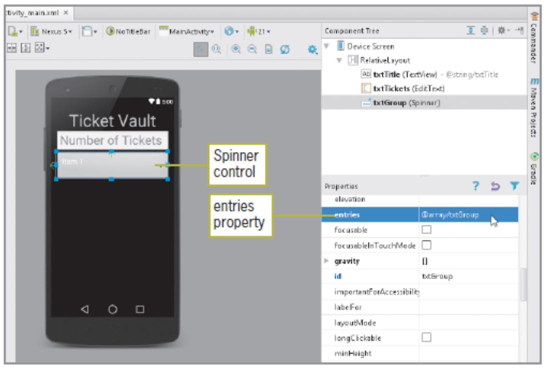


Figure 3-23 spinnerMode property options

Simplifying User Input (continued)

The prompt property connects to the resource named @string/prompt. The entries property connects to the resources of the String Array @array/txtGroup. The actual groups are displayed in the Spinner when the app is executed in the emulator (Figure 3-24).



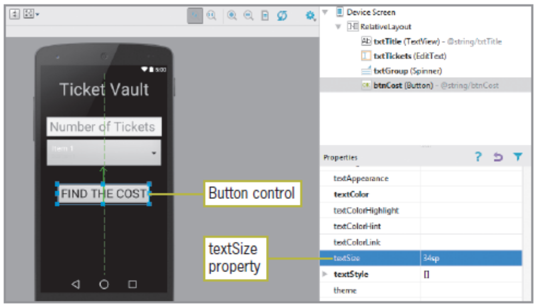
```
Component Tree
Device Screen
  RelativeLayout
    txtTitle (TextView) - @string/txtTitle
    txtTickets (EditText)
    txtGroup (Spinner)

Properties
elevation
entries                      @array/txtGroup
focusable                    ☐
focusableInTouchMode         ☐
gravity                      []
id                           txtGroup
importantForAccessibility
labelFor
layoutMode
longClickable                ☐
minHeight
```

Figure 3-24 Spinner control prompt and entries properties

Simplifying User Input (continued)

Adding the Button, TextView, and ImageView Controls

The Button control named btnCost displays the text FIND THE COST from the btnCost String and the size is changed to 34sp (Figure 3-25).



Device Screen
RelativeLayout
txtTitle (TextView) - @string/txtTitle
txtTickets (EditText)
txtGroup (Spinner)
btnCost (Button) - @string/btnCost

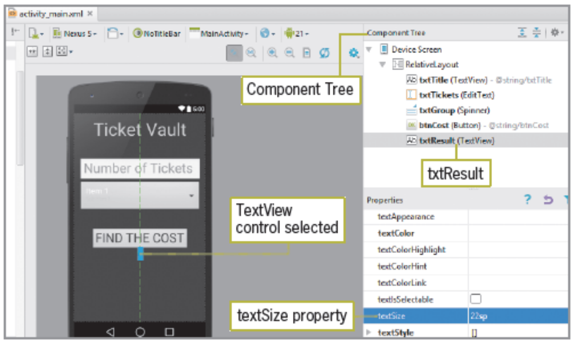
Properties

textAppearance	
textColor	
textColorHighlight	
textColorHint	
textColorLink	
textSize	34sp
textStyle	[]
theme	

Figure 3-25 Adding a Button control

Simplifying User Input (continued)

The txtResult TextView control is added to the emulator with the Text property blank (Figure 3-26).



Component Tree
Device Screen
RelativeLayout
txtTitle (TextView) - @string/txtTitle
txtTickets (EditText)
txtGroup (Spinner)
btnCost (Button) - @string/btnCost
txtResult (TextView)

Properties

textAppearance	
textColor	
textColorHighlight	
textColorHint	
textColorLink	
textIsSelectable	☐
textSize	22sp
textStyle	[]

Figure 3-26 Adding a TextView control to display results

Simplifying User Input (continued)

The concert image is displayed at the bottom of the emulator with a content description for accessibility purposes (Figure 3-27).

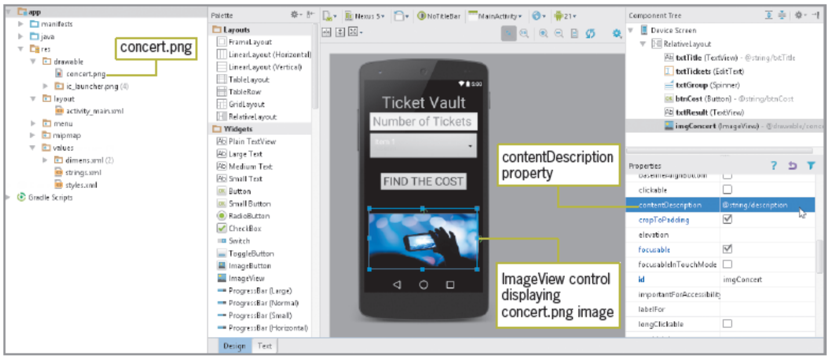

concert.png
contentDescription property
ImageView control displaying concert.png image
Ticket Vault
Number of Tickets
FIND THE COST

Figure 3-27 Adding an ImageView control

Simplifying User Input (continued)

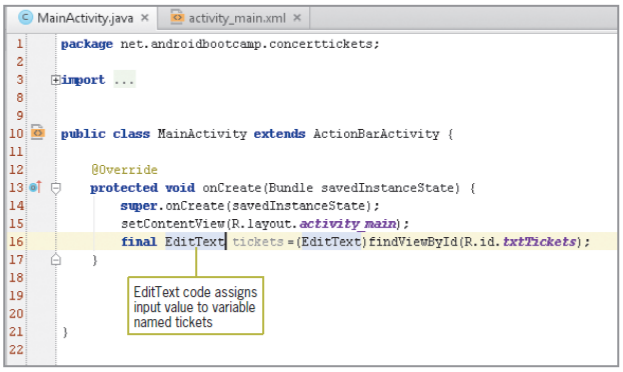
- **Coding the EditText Class for the Text Field**
  - A **variable** is used in programming to contain data that changes during the execution of a program
  - **Final** variables can be initialized but cannot be changed
  - Insert this code to create a variable:

Code Syntax

```
final EditText tickets = (EditText)findViewById(R.id.txtTickets);
```

Simplifying User Input (continued)

The EditText class extracts the value from the user's input for the number of tickets and assigns the value to the variable named tickets. Variables that the program has not used appear in gray text. This color is removed when a value is assigned later in the program (Figure 3-28).



MainActivity.java | activity_main.xml

```java
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets =(EditText)findViewById(R.id.txtTickets);
    }




}
```

Figure 3-28 Coding the EditText class for the Text Field

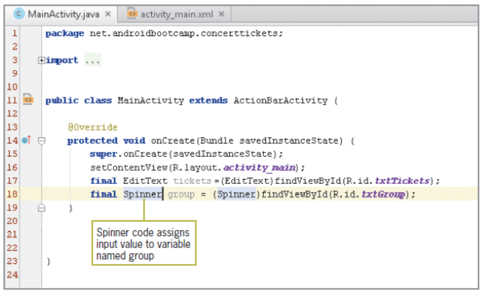
Simplifying User Input (continued)

Coding the Spinner Control

Code Syntax

```java
final Spinner group = (Spinner)findViewById(R.id.txtGroup);
```

The Spinner control assigns the value from the user's input to the variable named group (Figure 3-29).



MainActivity.java | activity_main.xml

```java
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets =(EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
    }




}
```

Figure 3-29 Coding the Spinner control

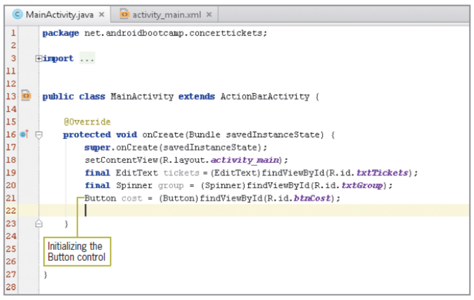
Simplifying User Input (continued)

Instantiating the Button Control

Code Syntax

```
final TextView result = ((TextView)findViewById(R.id.txtResult));
```

The Button control is initialized and the Button type is imported (Figure 3-30).

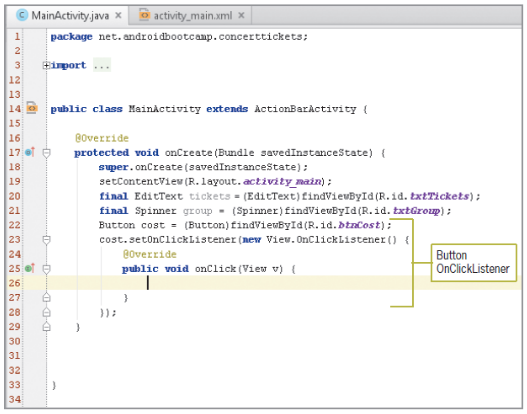


```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);

    }


}
```

Figure 3-30 Instantiated Button class

Simplifying User Input (continued)

The Button control is initialized and an OnClickListener auto-generated stub appears in the code window (Figure 3-31).

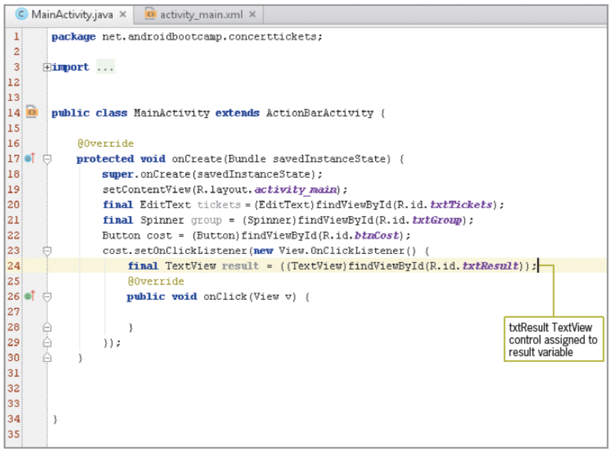


```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

            }
        });
    }




}
```

Figure 3-31 Initializing the OnClickListener

Simplifying User Input (continued)

The TextView control txtResult is assigned to the variable named result (Figure 3-32).



```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            final TextView result = ((TextView)findViewById(R.id.txtResult));
            @Override
            public void onClick(View v) {

            }
        });
    }




}
```

Figure 3-32 TextView control code

Declaring Variables

- Declare the variable
- Assign a value to the variable

Code Syntax

```
double costPerTicket = 79.99;
int numberOfTickets;
double totalCost;
```

Primitive Data Types

Type	Meaning	Range	Default Value
byte	Often used with arrays	–128 to 127	0
short	Often used with arrays	–32,768 to 32,767	0
int	Most commonly used number value	–2,147,483,648 to 2,147,483,647	0
long	Used for numbers that exceed int	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0
float	A single precision 32-bit floating-point number	+/–3.40282347 ^38	0
double	Most common for decimal values	+/–1.79769313486231570 ^308	0
char	Single character	Characters	0
boolean	Used for conditional statement	True or false	False

Table 3-3 Primitive data types in Java

Declaring Variables (continued)

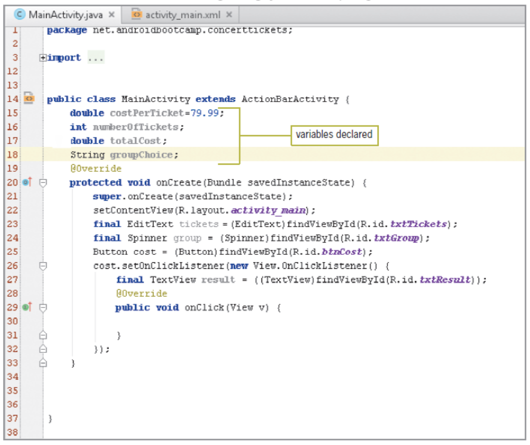
- **String Data Type**
  - The String type is a class and not a primitive data type
  - A string can be a character, word, or phrase
- **Declaring the Variables**
  - Typically declared at the beginning of an Activity
  - Variables must be declared before you can use them

Code Syntax

```
String groupChoice;
```

Declaring Variables (continued)

The variables are declared at the beginning of the activity (Figure 3-33).



```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {
    double costPerTicket=79.99;
    int numberOfTickets;
    double totalCost;
    String groupChoice;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            final TextView result = ((TextView)findViewById(R.id.txtResult));
            @Override
            public void onClick(View v) {

            }
        });
    }



}
```

Figure 3-33 Declaring variables for the activity

Declaring Variables (continued)

Code Syntax

```
numberOfTickets = Integer.parseInt(tickets.getText( ).toString( ));
```

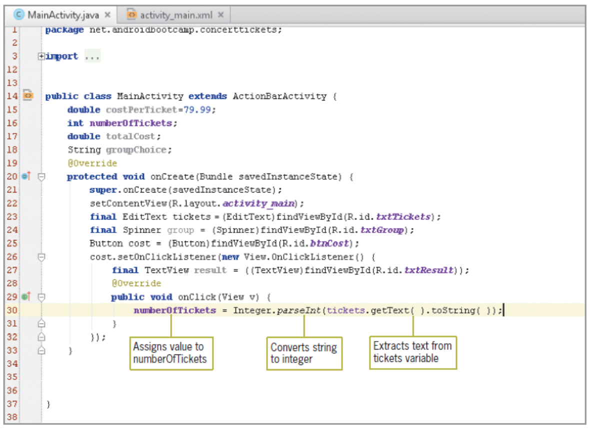
- **GetText() Method**
  - Read data stored in the EditText control with the **GetText()** method
  - Data is read in as a string, by default
  - A **Parse** class is used to convert strings into numbers

Numerical Data Type	Parse Types
Integer	Integer.parseInt( )
Float	Float.parseFloat( )
Double	Double.parseDouble( )
Long	Long.parseLong( )

Table 3-4 Parse type conversions

Declaring Variables (continued)

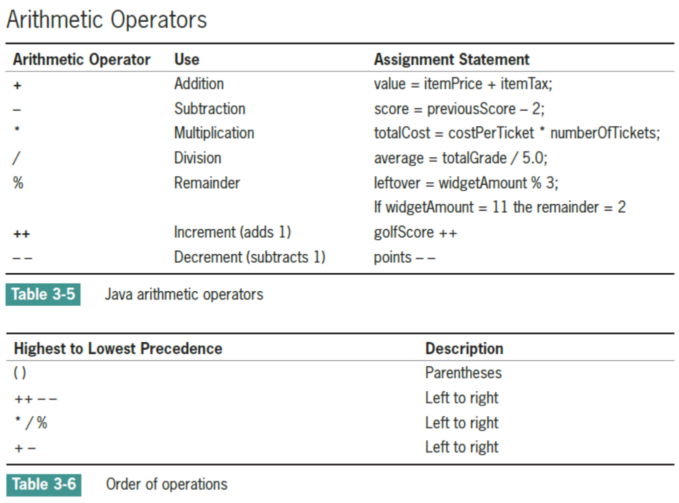
The GetText( ) method extracts the text from tickets, converts the string to an integer, and assigns the value to numberOfTickets (Figure 3-34).


MainActivity.java ×
activity_main.xml ×
Assigns value to numberOfTickets
Converts string to integer
Extracts text from tickets variable

```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {
    double costPerTicket=79.99;
    int numberOfTickets;
    double totalCost;
    String groupChoice;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            final TextView result = ((TextView)findViewById(R.id.txtResult));
            @Override
            public void onClick(View v) {
                numberOfTickets = Integer.parseInt(tickets.getText( ).toString( ));
            }
        });
    }




}
```

Figure 3-34 Converting a string to an integer

Working with Mathematical Operations



Arithmetic Operators

Arithmetic Operator	Use	Assignment Statement
+	Addition	value = itemPrice + itemTax;
–	Subtraction	score = previousScore – 2;
*	Multiplication	totalCost = costPerTicket * numberOfTickets;
/	Division	average = totalGrade / 5.0;
%	Remainder	leftover = widgetAmount % 3;
		If widgetAmount = 11 the remainder = 2
++	Increment (adds 1)	golfScore ++
– –	Decrement (subtracts 1)	points – –

Table 3-5 Java arithmetic operators

Highest to Lowest Precedence	Description
( )	Parentheses
++ – –	Left to right
* / %	Left to right
+ –	Left to right

Table 3-6 Order of operations

Working with Mathematical Operations
(continued)

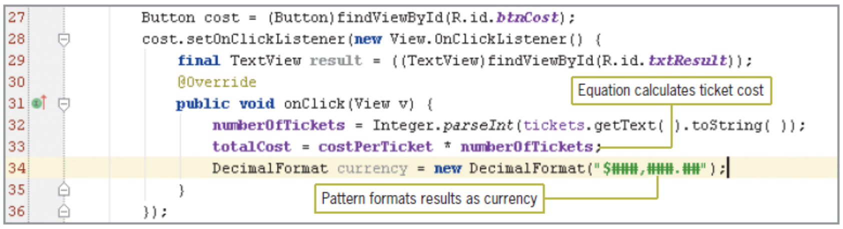
- **Formatting Numbers**
  - Currency format requires a dollar sign, a comma (if needed), a decimal point, and two decimal places
  - Java has a class called **DecimalFormat** that provides patterns, such as *$###,###.##* for displaying on the Android screen

Working with Mathematical Operations
(continued)

Code Syntax

```
DecimalFormat currency = new DecimalFormat("$###,###.##");
```

The equation computes the total cost of the tickets and DecimalFormat creates a currency format to use when the total cost is displayed (Figure 3-35).



```
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            final TextView result = ((TextView)findViewById(R.id.txtResult));
            @Override
            public void onClick(View v) {
                numberOfTickets = Integer.parseInt(tickets.getText( ).toString( ));
                totalCost = costPerTicket * numberOfTickets;
                DecimalFormat currency = new DecimalFormat("$###,###.##");
            }
        });
```

Equation calculates ticket cost
Pattern formats results as currency

Figure 3-35 Calculating and formatting the ticket cost

Displaying Android Output

GetSelectedItem() Method

– The **GetSelectedItem()** method returns the text label of the currently selected Spinner item

Code Syntax

```
groupChoice = group.getSelectedItem( ).toString( );
```

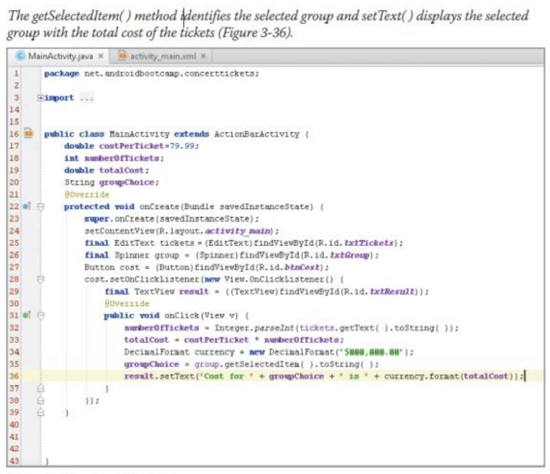
SetText () Method

– The **SetText()** method displays text in a TextView control

Code Syntax

```
result.setText("Cost for " + groupChoice + " is " +
currency.format(totalCost));
```

Displaying Android Output



The getSelectedItem( ) method identifies the selected group and setText( ) displays the selected group with the total cost of the tickets (Figure 3-36).

```
package net.androidbootcamp.concerttickets;

import ...


public class MainActivity extends ActionBarActivity {
    double costPerTicket=79.99;
    int numberOfTickets;
    double totalCost;
    String groupChoice;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText tickets = (EditText)findViewById(R.id.txtTickets);
        final Spinner group = (Spinner)findViewById(R.id.txtGroup);
        Button cost = (Button)findViewById(R.id.btnCost);
        cost.setOnClickListener(new View.OnClickListener() {
            final TextView result = ((TextView)findViewById(R.id.txtResult));
            @Override
            public void onClick(View v) {
                numberOfTickets = Integer.parseInt(tickets.getText( ).toString( ));
                totalCost = costPerTicket * numberOfTickets;
                DecimalFormat currency = new DecimalFormat("$###,###.##");
                groupChoice = group.getSelectedItem( ).toString( );
                result.setText("Cost for " + groupChoice + " is " + currency.format(totalCost));
            }
        });
    }




}
```

Figure 3-36 Completed code

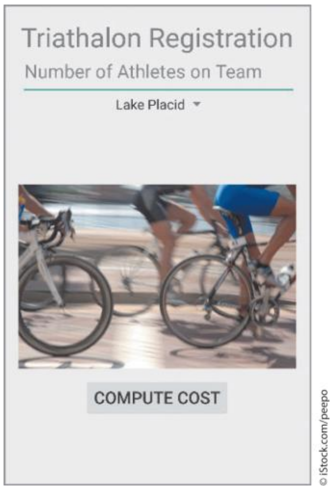
Summary

- Assign a theme to an Activity or an entire application to prevent apps from looking too similar
- Preview a theme by tapping or clicking the AppTheme button in the emulator, and then selecting a theme
- Use Text Fields to request input from users
- The strings.xml file is part of every Android application by default and contains strings used in the application, such as text displayed in a TextView, Spinner, or Button control
- To provide guidelines so users enter the correct data in a Text Field control, use the control's hint property to display light-colored text, also called a watermark, describing what to enter

Summary

- To handle the input that users enter into a Text Field control, you use the EditText class, which extracts the text and converts it for use in the Java code
- To use a variable, you must first declare the variable and then assign a value to it
- After assigning variables to hold the values entered in controls, you often need to convert the values in the assigned variables to the correct data type so the values can be used in calculations

Practice 2



- Prompt: Select a location
  - Lake Placid
  - Big Island Hawaii
  - Miami
- Compute cost
  - $72/athlete