Lesson 3: User Input, Variables, and Arithmetic Operations

INTRODUCTION

A number of applications require the user to enter data. The app then uses the data in some way. This lesson takes a look at providing the means for a user to interact with an app by providing input. The lesson also discusses the use of variables and arithmetic operations.

LESSON OBJECTIVES

By the end of this lesson, the student will be able to:

1. Identify the eight primitive data types.
2. Demonstrate the use of a Text Field to request input from users.
3. Demonstrate the use of a variable by declaring the variable and assigning a value to it.
4. Demonstrate the use of an array.
5. Demonstrate the use of a Spinner control.
6. Demonstrate the proper use of arithmetic operators to increment and decrement a variable.

LEARNING SEQUENCE

Required Reading	Read the following: • Online Lesson Material
Resources	View the following: • Android Application Development Tutorial – 10 – using setText method for our button (5:53) Other Resources: • Java Data Types: Understanding the 8 Primitive Data Types in Java • Java Basic Operators
Assignments	Complete the following: • Building a Simple User Interface • Quiz

INSTRUCTION

User Input

On an Android device, users can enter text in a lot of different ways. One of the most common ways is by using an onscreen keyboard (also known as a soft keyboard). The soft keyboard is usually found at the bottom of the screen. Some devices even have voice-to-text capabilities. There are a number of ways that a developer can work to have the user interact.

Text Fields

A Text Field lets the user enter some type of text into the app. When a text field is touched, the cursor is placed inside the field and the keyboard is displayed. Figure 1 shows the Text Fields available for user keyboard input in Eclipse.

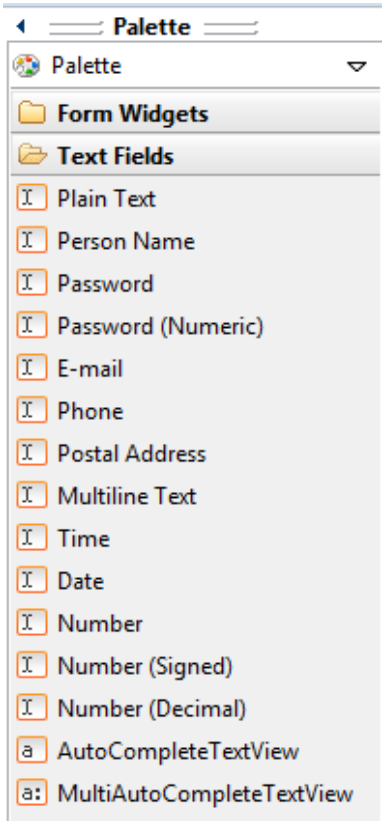


Figure 1: Text Fields

Some Text Field types only allow for specific data such as the numeric password. Use the Text Field control that will allow the user to enter the specific type of data required by the app. For example, if the Phone Text Field is used, the letters on the keyboard will not be available since a letter is not used in a phone number. The strings.xml file, located in the res/values folder contains commonly used text strings for the application.

Build It

For practice, complete the lesson, Building a Simple User Interface at the Android developer website. Even though the lesson references Android Studio, an alternate IDE, Eclipse can also be used to complete the lesson. Refer to the Exercise Instructions provided here.

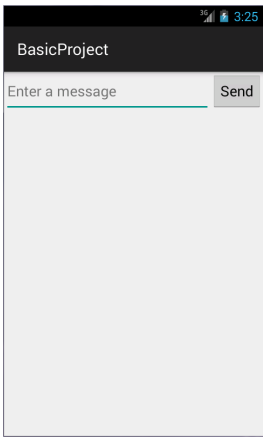


Figure 2: The completed User Interface

The result is displayed in Figure 2.

String Arrays

If a control holds multiple text strings, it is preferable to use a string array. An array is a container object that holds a fixed number of values that are all a single type. In this next example, the strings.xml file will be edited through the Android Resources in Eclipse instead of editing the code directly. To add a string array to the strings.xml file:

1. In the strings.xml file, click the **Add** button in the Android Resources window, shown in Figure 3.

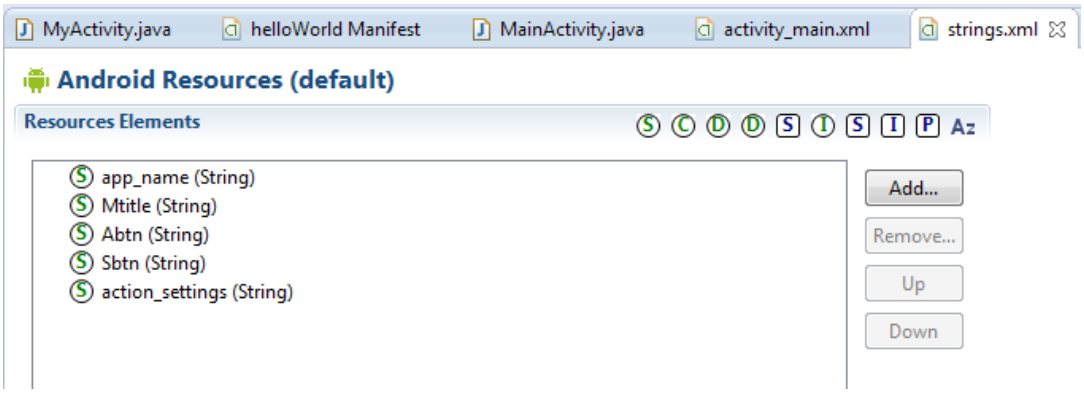


Figure 3: Android Resources

2. Select **String Array** and click **OK**.

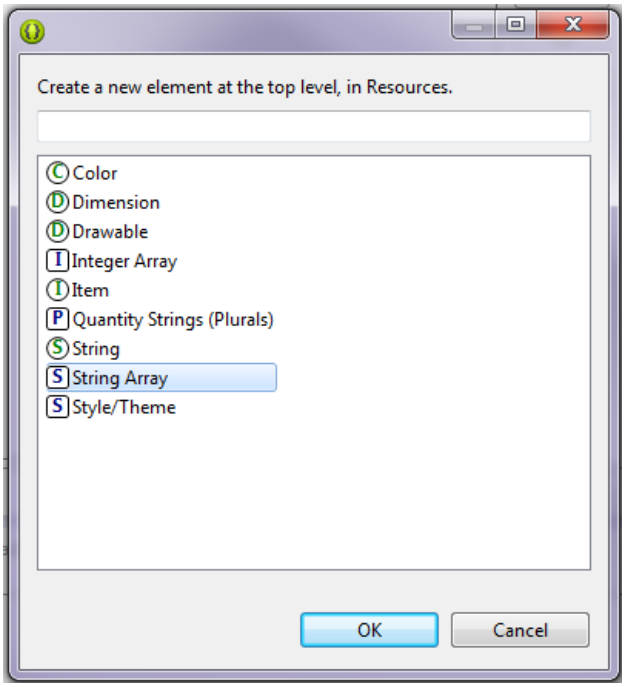


Figure 4: Adding a String Array

3. Enter a name.

Figure 5: String Array Name

4. With String Array selected, click the **Add** button. Click on **Item**, and then click **OK**.

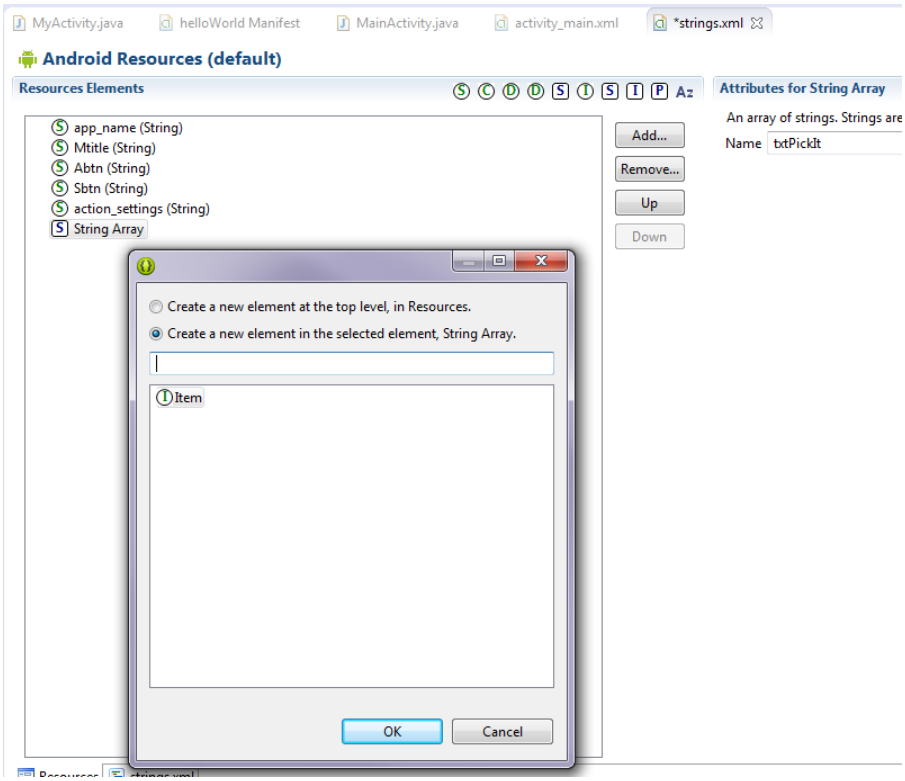


Figure 6: Adding Items to the String Array

5. Enter the value.
6. Repeat steps 4 and 5 until all values have been entered.
7. Submit the following for grading:
    - One screenshot showing values have been added.

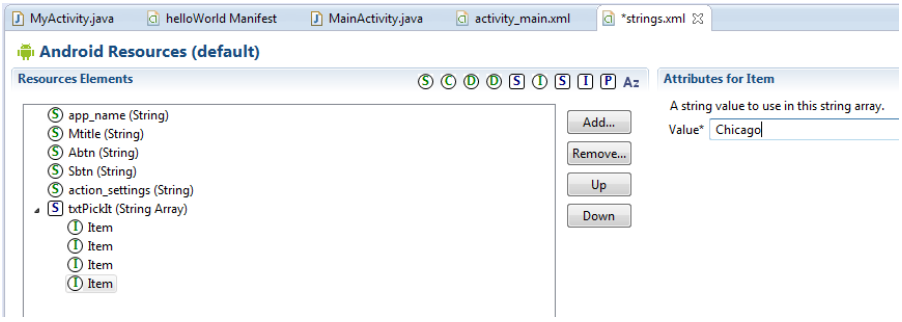


Figure 7: Add a value for each item

The resulting strings.xml file is shown in Figure 8.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <string name="app_name">Calculations</string>
   <string name="Mtitle">Your total is 0</string>
   <string name="Abtn">Add Button</string>
   <string name="Sbtn">Subtract Button</string>
   <string name="action_settings">Settings</string>
   <string-array name="txtPickIt">
      <item >Las Vegas</item>
      <item >Orlando</item>
      <item >New York City</item>
      <item >Chicago</item>
   </string-array>

</resources>
```

Figure 8: Edited strings.xml file

In this example, four items have been added to the String Array. A Spinner control can then be added to the app so that the user can make a selection. A Spinner control displays a prompt with the items in a popup window.

The Spinner control needs to be added to the activity_main.xml file.

1. Click the **Form Widgets** category in the Palette of Eclipse.
2. Drag and drop the **Spinner control** below the Subtract Button.

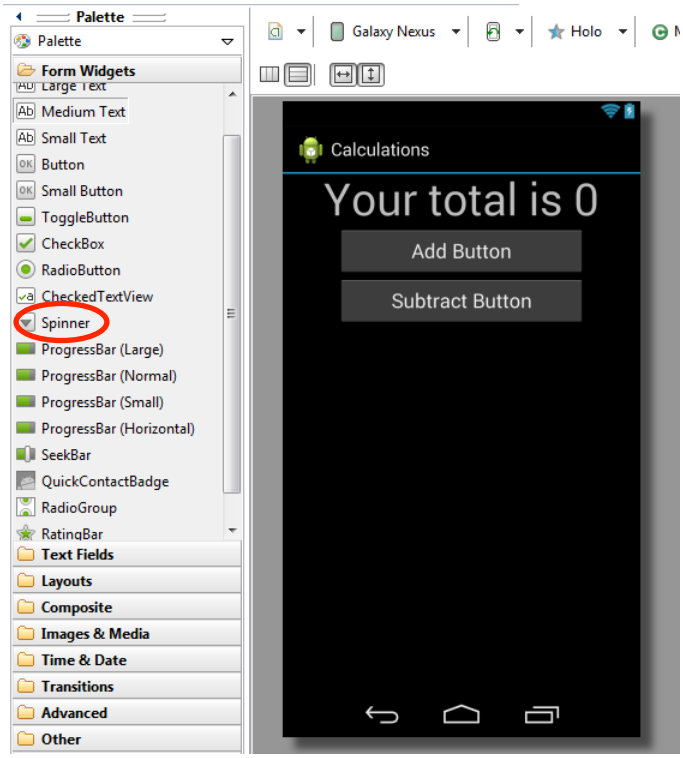


Figure 9 Selecting the Spinner Control from the Form Widgets folder

3. Change the Id property of the Spinner control to @id/txtPickIt and click outside the field.

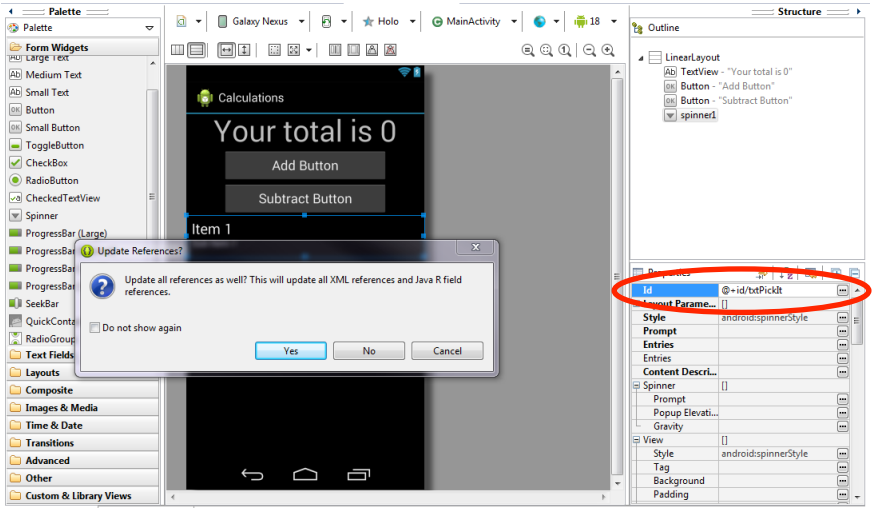


Figure 10 Changing the Id property of the Spinner Control

4. Select ***Yes*** to update all references.
5. Submit the following for grading:
    - One screenshot showing that a Spinner control has been added to the app so that the user can make a selection.

The result is shown in the emulator.

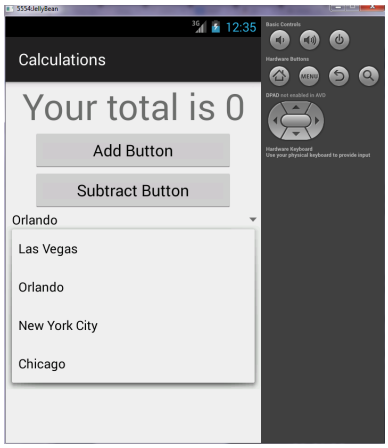


Figure 11: Spinner control

The code that was added to the activity_main.xml file is shown in Figure 12.

```
<Spinner
    android:id="@+id/txtPickIt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:prompt="@string/city_prompt"
    android:entries="@array/txtPickIt" />
```

Figure 12: Corresponding XML code in activity_main.xml

Auto-Complete Suggestions

As a user types, a subclass of `EditText` called `AutoCompleteTextView` can be used to provide suggestions. Auto-complete suggestions can be used with an array, like in the previous example, or it can be used with a database. Entering the name of a country or the name of a state are both examples of when this subclass can be used.

Add the following to the layout file:

```
<AutoCompleteTextView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/autocomplete_state"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

Figure 13: AutoCompleteTextView Example

The array that contains the text suggestions need to be added to the strings.xml file.

Variables

Variables can be used for a couple of different purposes. Variables store values that can change over time. For example, a variable called counter was used to store the value in the Calculations project. A variable can also be used to represent constants that do not change over time.

To use a variable:

1. The variable must be declared.
2. An initial value must be assigned to the variable.

The following types of variables are available in Java:

- Non-static fields – belongs to an object (i.e., each instance of a class); also referred to as instance variables
- Static fields – belongs to a class; also referred to as class variables. Only one copy of this variable exists, and the same value is applied to all instances
- Local Variables – used by a method to store a temporary state, so they are only visible to the method in which they are declared
- Parameters – passed to a method when the method is called

Variable names are case sensitive. It is recommended that variable names begin with a letter. Other characters that can be used to make up a variable name include letters, numbers, an underscore (_) or a dollar sign ($). The dollar sign is usually never used, though.

If a variable name consists on one word, use all lowercase letters. For example, a variable named speed would be declared as follows:

```
speed = 0
```

A variable name that consists of two words would use the following convention:

```
bikeGear = 10
```

If the variable stores a constant value, use all upper case letters. If the variable name for a constant uses two words, use the underscore in between. The following variable is the number of gears found on a bicycle:

```
NUMBER_GEARS = 18
```

Primitive Data Types

There are eight primitive types of data built into Java that are used to classify data types. That means data from one declared variable type cannot be used in another. Read Java Data Types: Understanding the 8 Primitive Data Types in Java for a description of each type along with examples.

Arithmetic Operations

Many apps require the ability to perform arithmetic operations to add, subtract, multiply, and divide. Go to Java Basic Operators and read the Arithmetic Operators section to view the description and example of each operator. Parenthesis can be used to force the order of operations, if required.

The arithmetic operators that are used in mathematical expressions are shown in the table below (assume A=5 and B=10).

Operator	Description
+	Addition (A + B results in 15)
-	Subtraction (B - A results in 5)
*	Multiplication (A * B results in 50)
/	Division (B / A results in 2)
%	Modulus (B % A results in 0. In this example, a modulus divides 10 by 5 with a remainder of 0; the modulus is the remainder)
++	Increment (A++ results in 6)
--	Decrement (A- - results in 4)

The calculation project used the arithmetic operator that increments and decrements a value. For example, when the counter variable needed to be incremented by 1, the following statement was used:

```
counter++;
```

To decrement the counter by one:

```
counter--;
```

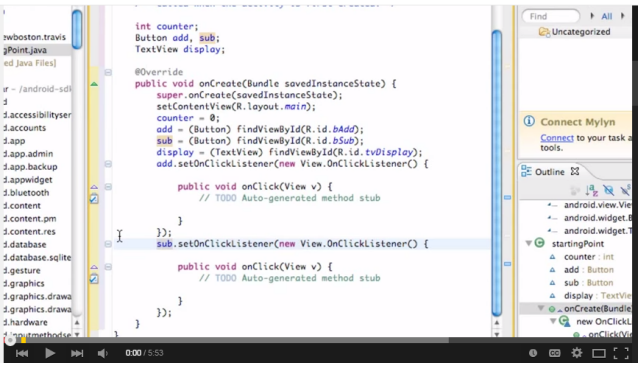
Parentheses are used to force the order of operations. Without using parentheses, 5 * 4 + 10 / 2 results in 25, Using parentheses changes the result: 5 * ((4 + 10) / 2) results in 35.

The standard order of operations is as follows:

1. Exponents and roots
2. Multiplication and division
3. Addition and subtraction

In the examples given, the red expression is calculated first, the purple second, and green is third according to the standard order of operations.

View Android Application Development Tutorial – 10 – using setText method for our button (5:53) to learn how to add functionality to the calculation buttons.



SUMMARY

This lesson discussed user input, variables and operations. An example using a Text Field to allow for a specific type of data was discussed. Additionally, a Spinner control using a string array, which holds multiple text strings, was demonstrated. Finally, using variables and arithmetic operations within an app were discussed.

ASSIGNMENTS

Note: Assignment 1, Building a Simple User Interface, details were provided within the lesson. Refer to information here if needed.

1. Building a Simple User Interface
2. Quiz