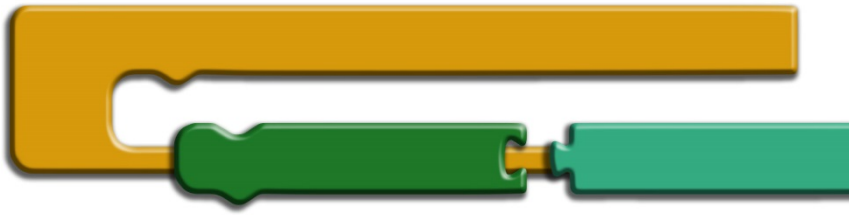
starting out with >>> APP INVENTOR FOR ANDROID

CHAPTER 5

Repetition Blocks, Times, and Dates



TONY GADDIS · REBECCA HALSEY

Topics

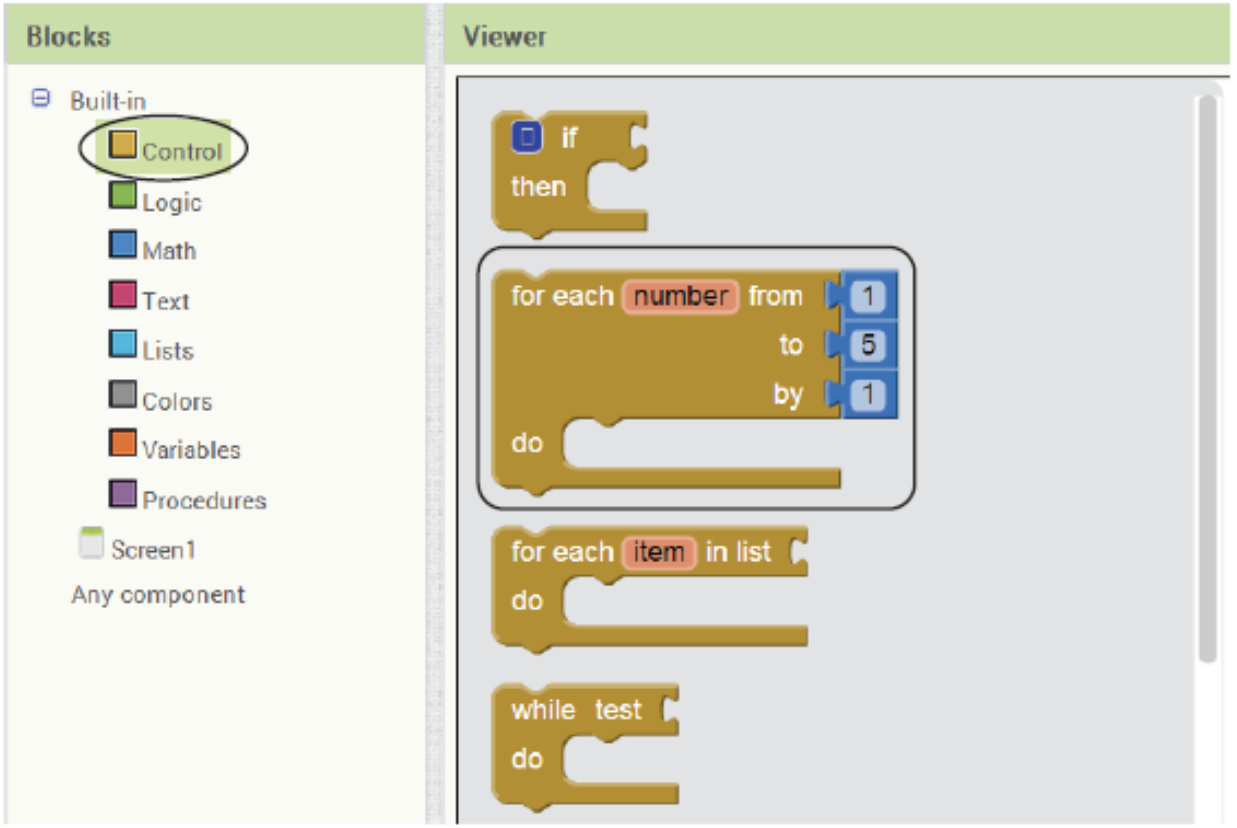
- The *Notifier* Component
- The *while* Loop
- The *for each* Loop
- The *Clock* Component
- The *DatePicker* Component

The `for each` Loop

- The *`for each`* loop is designed to increment a counter variable over a range of values.
- It is ideally suited for problems requiring a loop that iterates a specific number of times.

The for each Loop

Figure 5-23 The for each Block (*Source:* MIT App Inventor 2)



The `for each` Loop

1) The loop's counter variable – variable named number is automatically created.

2) The *`from`* socket – specifies the variable's starting value.

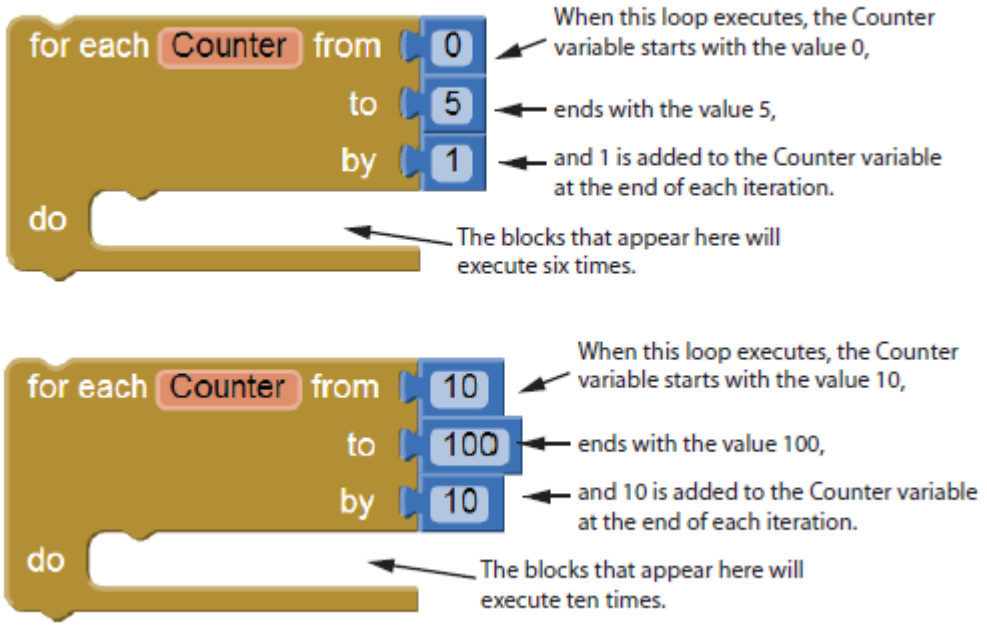
3) The *`to`* socket – specifies the counter variable's ending value.

4) The *by* socket specifies the amount added to the counter variable at the end of each iteration.

5) The blocks that are plugged into the *do* socket will execute each time the loop iterates.

The for each Loop

Figure 5-26 Examples for the `for each` Loop (*Source:* MIT App Inventor 2)



The `for each` Loop

- Top example the *`Counter`* variable starts with 0 and ends with 5.
- At the end of each iteration, 1 is added to the *`Counter`* variable.
- The loop will execute 6 times
- In the bottom example, the *`Counter`* starts with 10 and ends with 100.
- At the end of each iteration, 10 is added to the *`Counter`* variable.
- The loop will execute 10 times.

The `for each` Loop

Calculate a Running Total

- Programs that calculate the total of a series of numbers typically have two elements.
  1. A loop that reads each number in the series.
  2. A variable that accumulates the total of the numbers as they are read.
- The *accumulator* is the variable that is use to accumulate the total.
- It is very important that the *accumulator* starts with 0.

The `Clock` Component

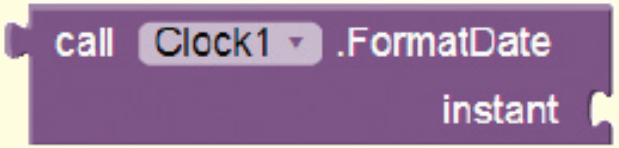
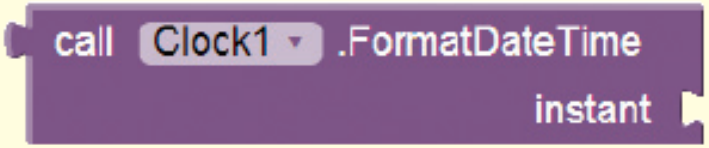
- Gets the date and time from the internal system clock and provides methods and functions for working with dates and times.
- Allows you to get the current date and time from the device's internal clock.
- It also serves as a timer that performs operations at regular time intervals.

The `Clock` Component

- The *`Clock`* component works with dates and times using a special value known as an *instant*.
- An *instant* represents a number in time.
- An *instant* contains both date and time.
- You can't print an *instant* on a screen.

The Clock Component

Table 5-3 Date and Time Formatting Functions (*Source:* MIT App Inventor 2)

Function	Description
	Formats the instant that is plugged into the `instant` socket as text describing the time only.
	Formats the instant that is plugged into the `instant` socket as text describing the date only.
	Formats the instant that is plugged into the `instant` socket as text describing both the date and the time.

The Clock Component

Figure 5-32 The ClockDemo App in the Designer (*Source:* MIT App Inventor 2)

The `Clock` Component

- In Figure 5-32 notice that
  - *`Screen1.Initialize`* event handler calls *`Clock1.Now`*
  - The *`FormatTime`* block formats the instant as text describing the time.
  - That block is assigned to the *`LabelTimeDisplay.Text`* property.

The `Clock` Component

Other Clock Methods

- The *`Clock`* component provides many other methods. Table 5-5 shows the *`Clock`* component's *`Add`* functions.

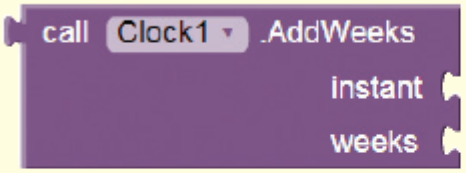
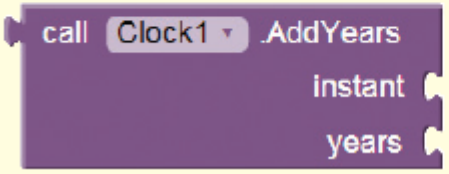
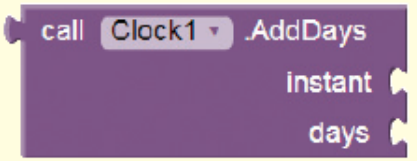
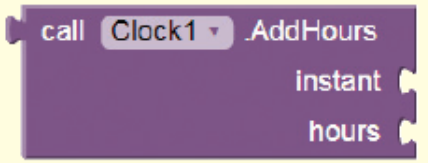
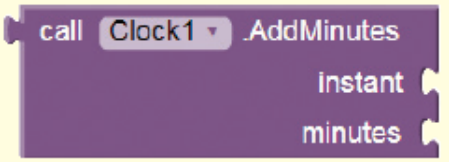
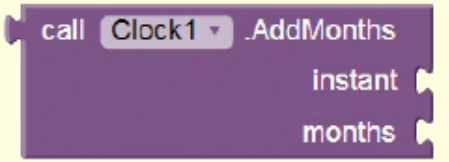
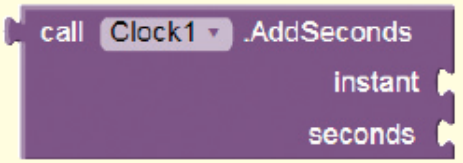
Function	Description
	Requires two arguments: an `instant` and a number of `weeks`. This function returns an instant in time that is the specified number of weeks after the given `instant`.
	Requires two arguments: an `instant` and a number of `years`. This function returns an instant in time that is the specified number of years after the given `instant`.

Table 5-5 The Clock component's Add functions (Source: MIT App Inventor 2)

Function	Description
	Requires two arguments: an `instant` and a number of `days`. This function returns an instant in time that is the specified number of days after the given `instant`.
	Requires two arguments: an `instant` and a number of `hours`. This function returns an instant in time that is the specified number of hours after the given `instant`.
	Requires two arguments: an `instant` and a number of `minutes`. This function returns an instant in time that is the specified number of minutes after the given `instant`.
	Requires two arguments: an `instant` and a number of `months`. This function returns an instant in time that is the specified number of months after the given `instant`.
	Requires two arguments: an `instant` and a number of `seconds`. This function returns an instant in time that is the specified number of seconds after the given `instant`.

The `DatePicker` Component

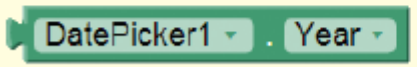
The *`DatePicker`* component appears as a button on an app's screen. When the user clicks the *`DatePicker`* button it displays a dialog box that allows the user to select a date.

Figure 5-42 The DatePicker Component *(Source: MIT App Inventor 2)*

The `DatePicker` Component

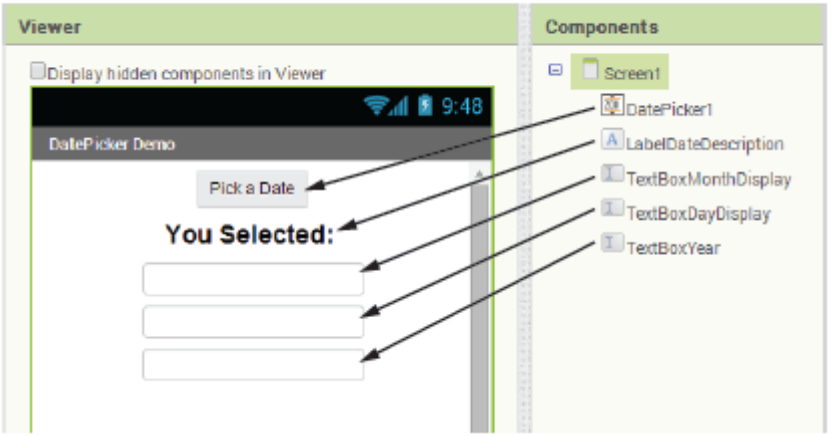
In the Blocks Editor, there are four properties in particular that you will work with.

Table 5-6 `DatePicker` properties (*Source:* MIT App Inventor 2)

Property	Description
	The Day property is set to the day of the month that was last selected using the DatePicker.
	The Month property is set to the number of the month that was last selected using the DatePicker. The months are numbered starting with 1, so January is 1, February is 2, and so forth.
	The MonthInText property is set to the name of the month (such as *January*, *February*, and so forth) that was last selected using the DatePicker.
	The Year property is set to the year that was last selected using the DatePicker.

The `DatePicker` Component

Figure 5-43 The DatePickerDemo App (*Source*: MIT App Inventor 2)



- When the user clicks the *Set* button in the *`DatePicker`*'s dialog box, an *`AfterDateSet`* event occurs.
- To retrieve the date, you can create an event handler for the *`AfterDateSet`* event.

The DatePicker Component

Figure 5-44 The DatePickerDemo App's Workspace in the Blocks Editor

(*Source:* MIT App Inventor 2)

The `DatePicker` Component

- Notice that the app has one event handler: *`DatePicker1.AfterDateSet`*.
- It displays the name of the selected month in the *`TextBoxMonthDisplay`* component.
- Figure 5-45 shows an example of the app running in the emulator.

The DatePicker Component

Figure 5-45 The DatePickerDemo App Running in the Emulator (*Source:* MIT App Inventor 2)