



京都先端科学大学

Introduction to Design (Track 3)

4.2 Hands-on: Counter App and BMI Calculator App

Zilu Liang

www.zilu-liang.net/id3

Important!!

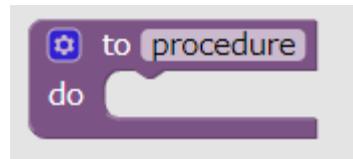
Please disinfectize your hands before entering the classroom!
入室前にアルコールを使用して手指消毒を行ってください。

Please disinfectize your chair and table!

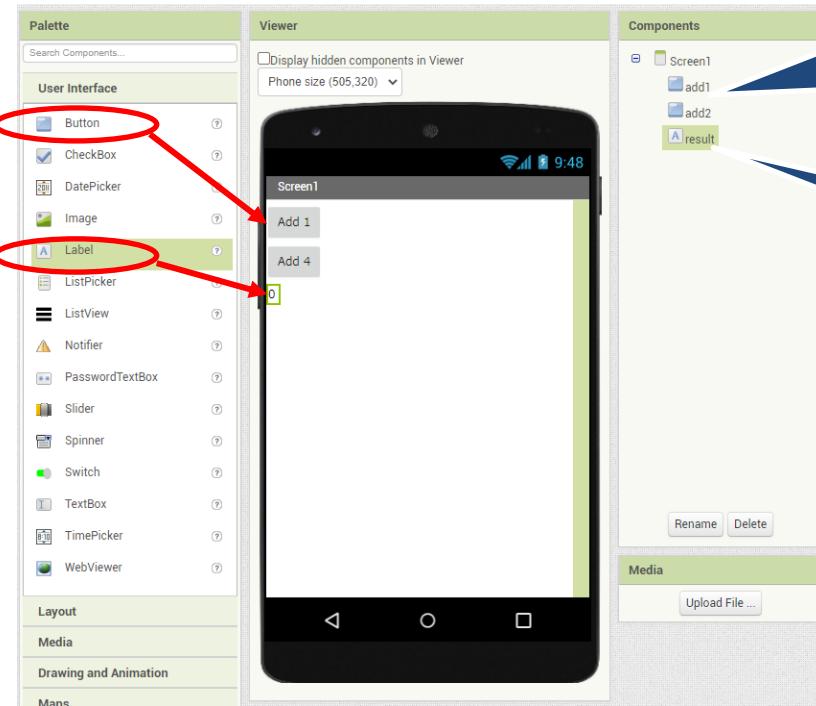
- ①ペーパーにアルコールを噴霧してください。
- ②アルコールが噴霧されたペーパーで、使用箇所（テーブル、椅子など）を拭き取ってください。
- ③使用済のペーパーは廊下のごみ箱に捨ててください。



I. Counter app with ‘to-do’ procedure



Build UI in designer view



- Drag and drop two buttons
- Rename the buttons to 'add1' and 'add2'

- Drag and drop a label
- Rename the label to 'result'

Switch to blocks view

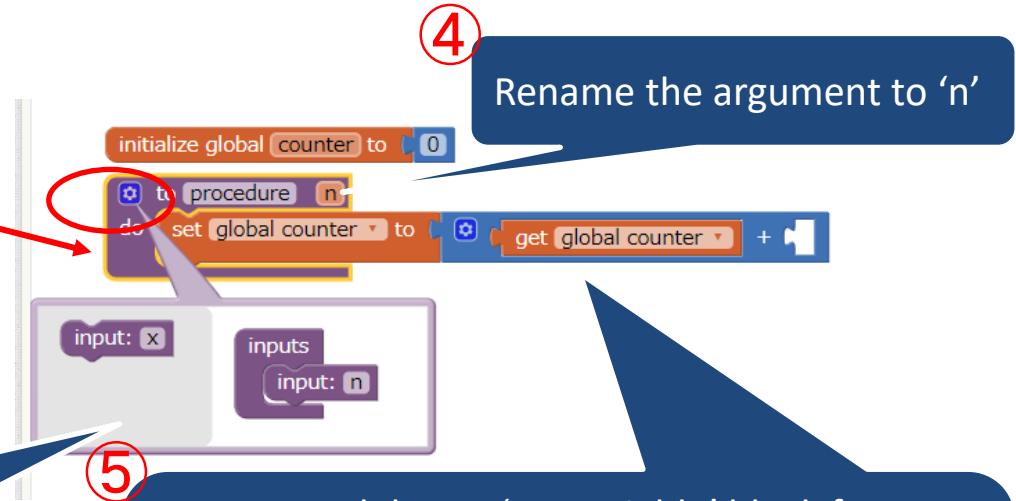
The image shows the Scratch 3.0 interface in the 'Blocks' view. The top navigation bar includes 'counterApp' (project name), 'Screen1 ▾', 'Add Screen ...', 'Remove Screen', 'Publish to Gallery', 'Designer' (highlighted with a red oval), and 'Blocks' (highlighted with a red oval). The left sidebar lists categories: 'Built-in' (Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures) and 'Screen1' (click_button, label). The 'Any component' category is also listed. The main workspace is titled 'Viewer' and contains two warning icons: a yellow triangle with '0' and a red circle with '0'. A 'Show Warnings' button is at the bottom of the workspace. To the right of the workspace are tool icons: a backpack, a circle with a dot, a plus sign, a minus sign, and a trash can.

Initialize a global variable

The image shows the Scratch interface. On the left is the 'Blocks' palette, which is divided into sections: 'Built-in' (Control, Logic, Math, Text, Lists, Dictionaries, Colors), 'Screen1' (add1, add2, result), and 'Any component'. The 'Math' category under 'Built-in' and the 'Variables' category under 'Screen1' are circled with red arrows pointing to them from a callout bubble. On the right is the 'Viewer' area, which contains a single script: 'initialize global [counter] to [0]'. A red arrow points from the 'Variables' circle to the 'counter' variable in the script.

Rename the global variable to 'counter'

Create a 'to-do' procedure



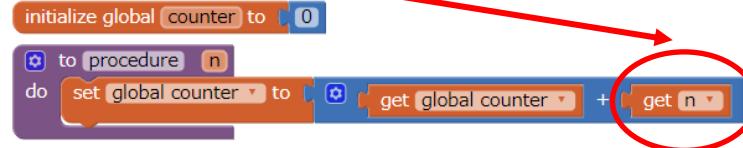
- ③
- Click on the mutator
 - Inside the mutator, add an argument by dragging an input block on the left side and dropping it inside the inputs block on the right side

- ④
- Rename the argument to 'n'
- ⑤
- Drag and drop a 'set variable' block from 'Variables'
 - Drag and drop an 'add' block from 'Math'
 - Drag and drop a 'get 'variable value'' block from 'Variables'

Create a ‘to-do’ procedure



- Hover mouse on the argument ‘n’
- Drag the ‘get’ block and drop it inside the ‘add’ block

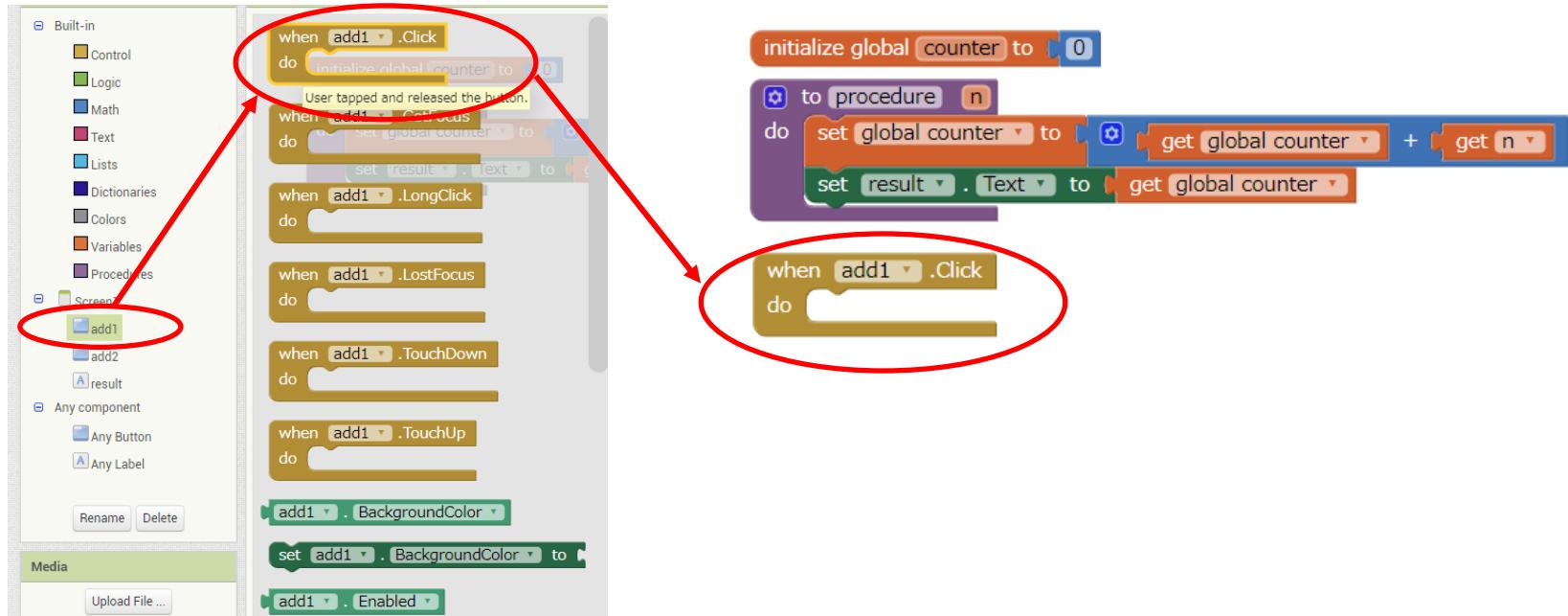


Create a ‘to-do’ procedure



Set the result.text to the value of
the global counter

Add click handler to button 'add1'



Call the procedure in click handler

The image shows a Scratch script in the 'Viewer' window and its corresponding code blocks in the 'Blocks' palette.

Blocks Palette:

- Built-in:
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1:
 - add1
 - add2
 - result
- Any component:
 - Any Button
 - Any Label

Viewer Window:

The script consists of the following blocks:

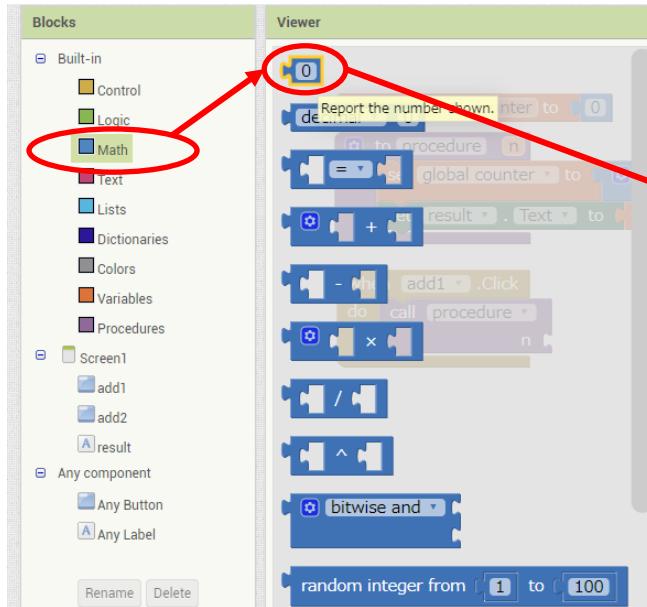
- A **when green flag clicked** hat block.
- An **do** control loop block.
- Inside the loop:
 - An **call [procedure v] parameter n** control block.
 - An **set [global counter v] to [0]** control block.
 - An **do** control loop block.
 - Inside the inner loop:
 - An **set [global counter v] to [get [global counter v] + get [n v]]** control block.
 - An **set [result v] to [get [global counter v]]** control block.

Code:

```
when green flag clicked
do
    call [procedure v]
    set global counter to 0
    do
        set global counter to (get global counter) + (get n)
        set result to (get global counter)
    end
end
```

A red oval highlights the **call [procedure v]** block, and another red oval highlights the **call [procedure v]** block inside the inner loop.

Set the procedure argument



```
initialize global [counter] to [0]
[when green flag clicked v]
  [call procedure [n v]]
    [do
      [set [global counter] to [get [global counter v] + get [n v]]]
      [set [result v] to [get [global counter v]]]
    end]
  [end]
end
```

when [add1 v].Click
do [call procedure [n v]]

Set it to 1

Add click handler to button 'add2'

The image shows a Scratch script editor interface. On the left, the 'Blocks' palette lists categories like Built-in, Screen1, and add1. The 'Viewer' pane shows a script for 'add2' with a circled 'when [add2 .Click] do' block. A red arrow points from this block to a second circled 'when [add2 .Click] do' block in a larger script on the right. This second script starts with 'initialize global [counter] to [0]' and includes a 'call procedure [n v 1]' block. Another red arrow points from the 'n v 1' block to a blue callout box.

initialize global [counter] to [0]

when [add1 .Click] do

call procedure [n v 1]

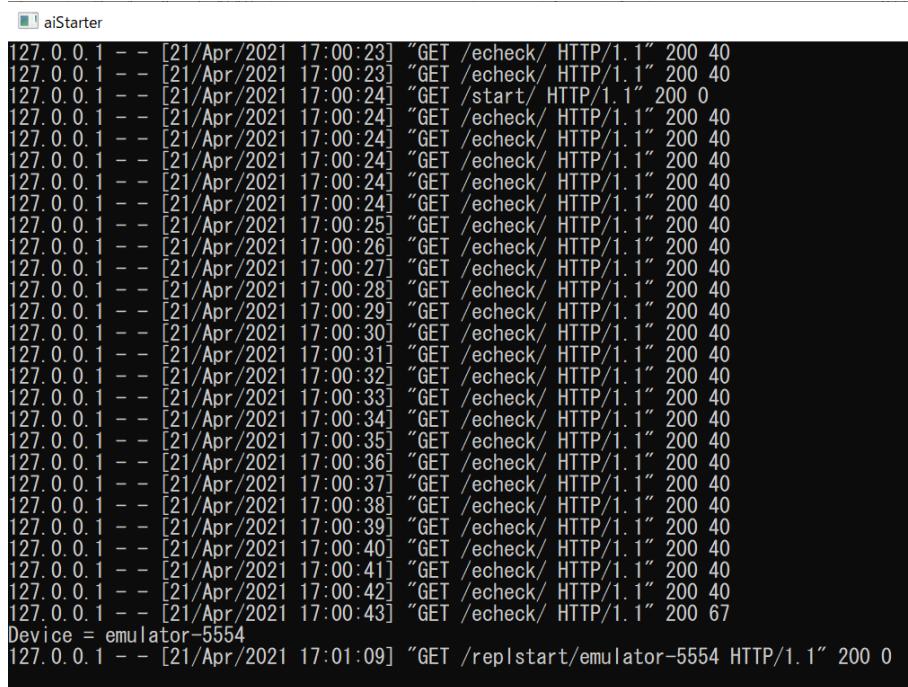
when [add2 .Click] do

call procedure [n v 4]

Add a number and set it to 4

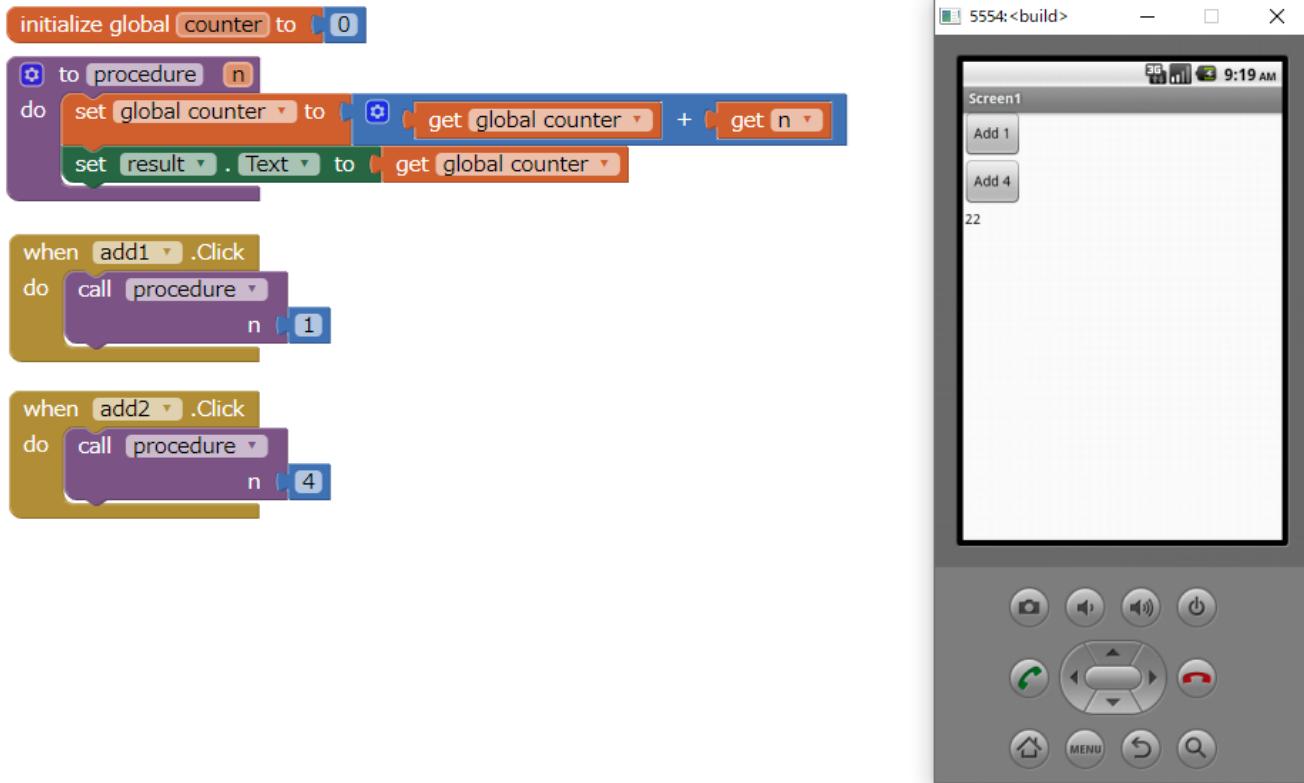
Test App on Emulator

Click on the aiStarter on your computer, you should see the following window open.



```
aiStarter
127.0.0.1 -- [21/Apr/2021 17:00:23] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:23] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /start/ HTTP/1.1" 200 0
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:24] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:25] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:26] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:27] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:28] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:29] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:30] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:31] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:32] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:33] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:34] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:35] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:36] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:37] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:38] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:39] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:40] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:41] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:42] "GET /echeck/ HTTP/1.1" 200 40
127.0.0.1 -- [21/Apr/2021 17:00:43] "GET /echeck/ HTTP/1.1" 200 67
Device = emulator-5554
127.0.0.1 -- [21/Apr/2021 17:01:09] "GET /replstart/emulator-5554 HTTP/1.1" 200 0
```

Test App on Emulator



Code Anatomy: Counter App



initialize global [counter] to [0] → Initialize the global variable 'counter' and set it to 0

[to [procedure] n]
do
[set [global counter] to [get [global counter] + [get n]]
[set [result] . [Text] to [get [global counter]]]

when [add1 v].Click
do
[call [procedure] n [1]]

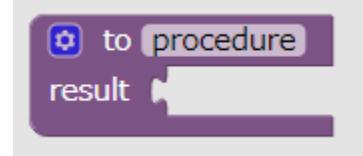
When 'add1' button is clicked, call 'procedure' and pass the argument 'n=1'

when [add2 v].Click
do
[call [procedure] n [4]]

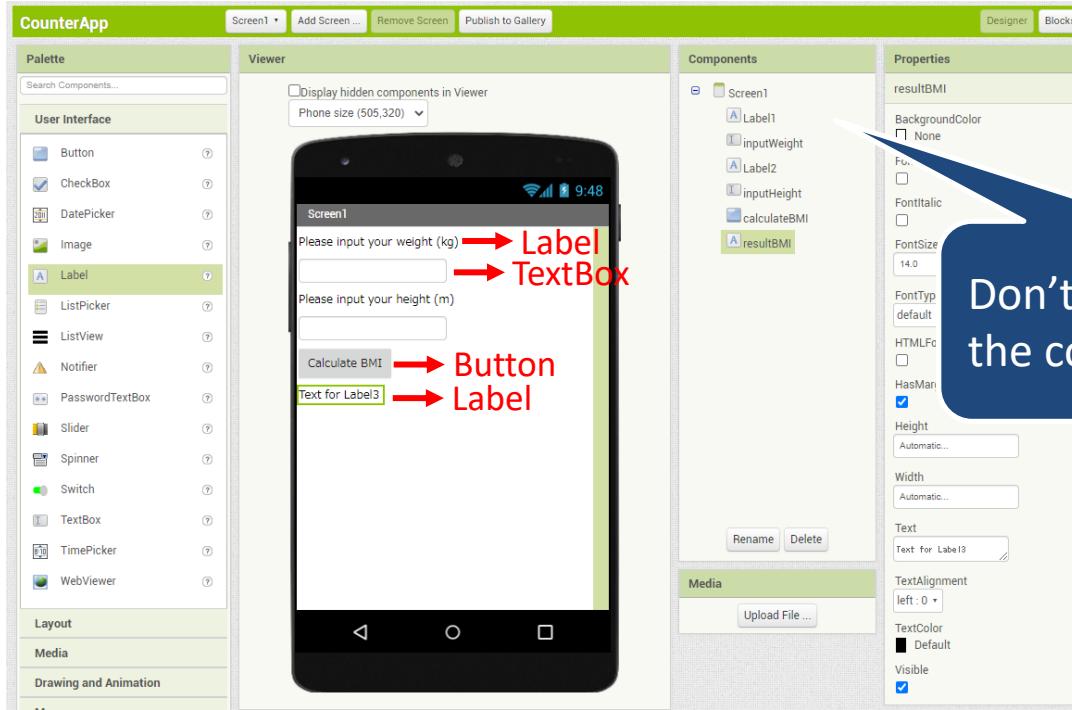
When 'add2' button is clicked, call 'procedure' and pass the argument 'n=4'

→ Define a 'procedure' that takes in the argument 'n', increase the counter by 'n', and show the result in the label

II. BMI calculator app with ‘to-result’ procedure



Build UI in designer view



Don't forget to rename
the components

Create a 'to-result' procedure

The image shows the Scratch script editor interface. On the left is the 'Blocks' palette with categories like Built-in, Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. The 'Procedures' category is highlighted with a red oval. In the 'Viewer' window, a script is being built:

- A 'Control' block: [+] to [procedure] do []
- An 'Events' block: [when green flag clicked]
- A 'Control' block: [repeat [] until []]
- Inside the repeat loop:
 - A 'Control' block: [set [result] to [0]]
 - A 'Math' block: [add [1] to [result]]
 - A 'Text' block: [say [result] for [1] seconds]
 - A 'Math' block: [change [height] by [1]]
 - A 'Math' block: [change [weight] by [1]]
 - A 'Control' block: [next costume]
- A 'Control' block: [end repeat]

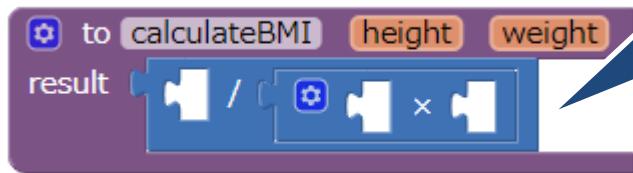
A callout box contains the following instructions:

- Rename the procedure to 'calculateBMI'
- Click the mutator and add two arguments
- Rename the arguments to 'height' and 'weight'

Below the script, two additional blocks are shown:

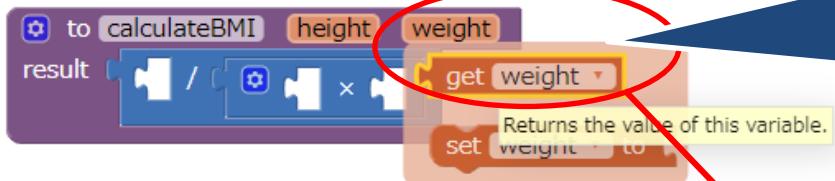
- 'input: x'
- 'inputs'
 - 'input: height'
 - 'input: weight'

Create a ‘to-result’ procedure

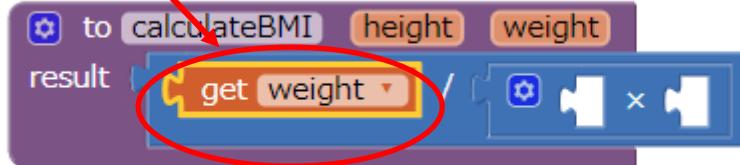


- Drag a division block from ‘Math’ and drop it inside the procedure
- Drag a multiplication block from ‘Math’ and drop it inside the denominator

Create a ‘to-result’ procedure



- Hover mouse on argument ‘weight’
- Drag the ‘get weight’ block and place it inside the numerator



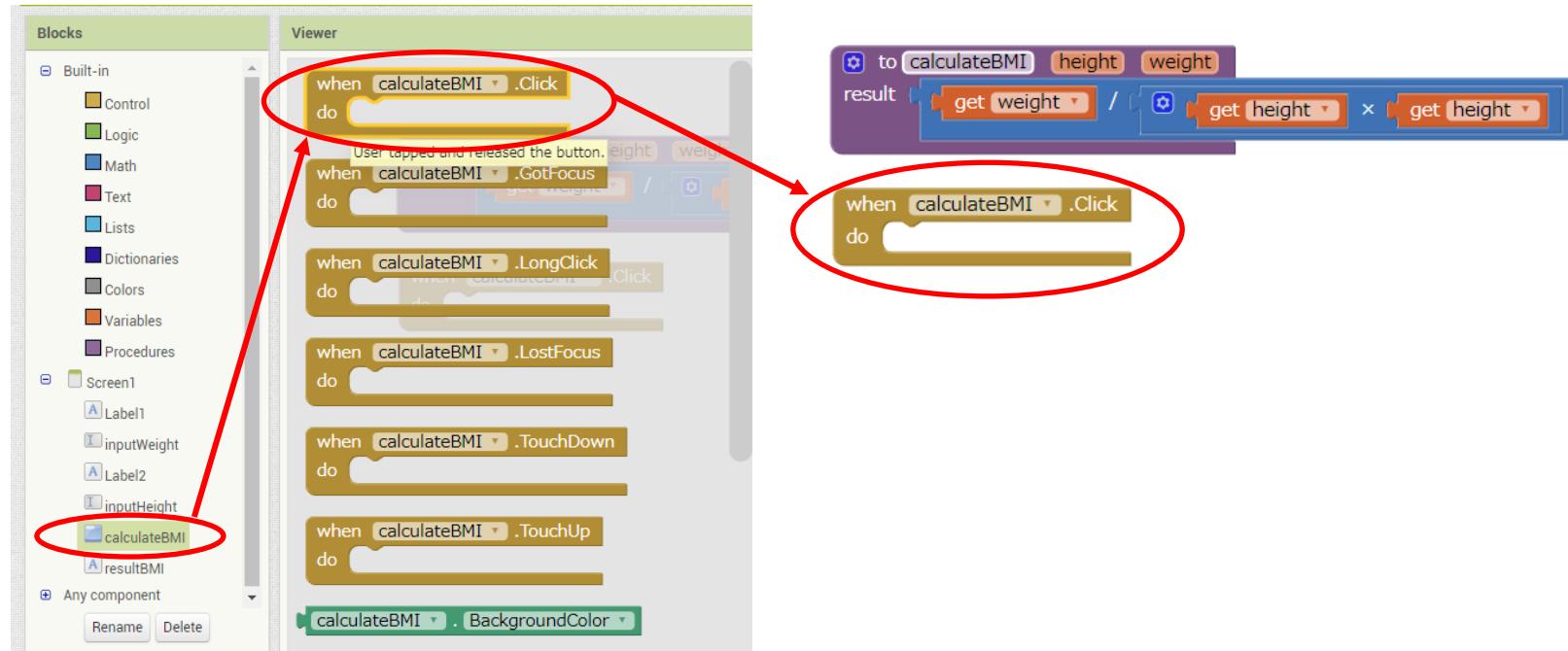
Create a ‘to-result’ procedure



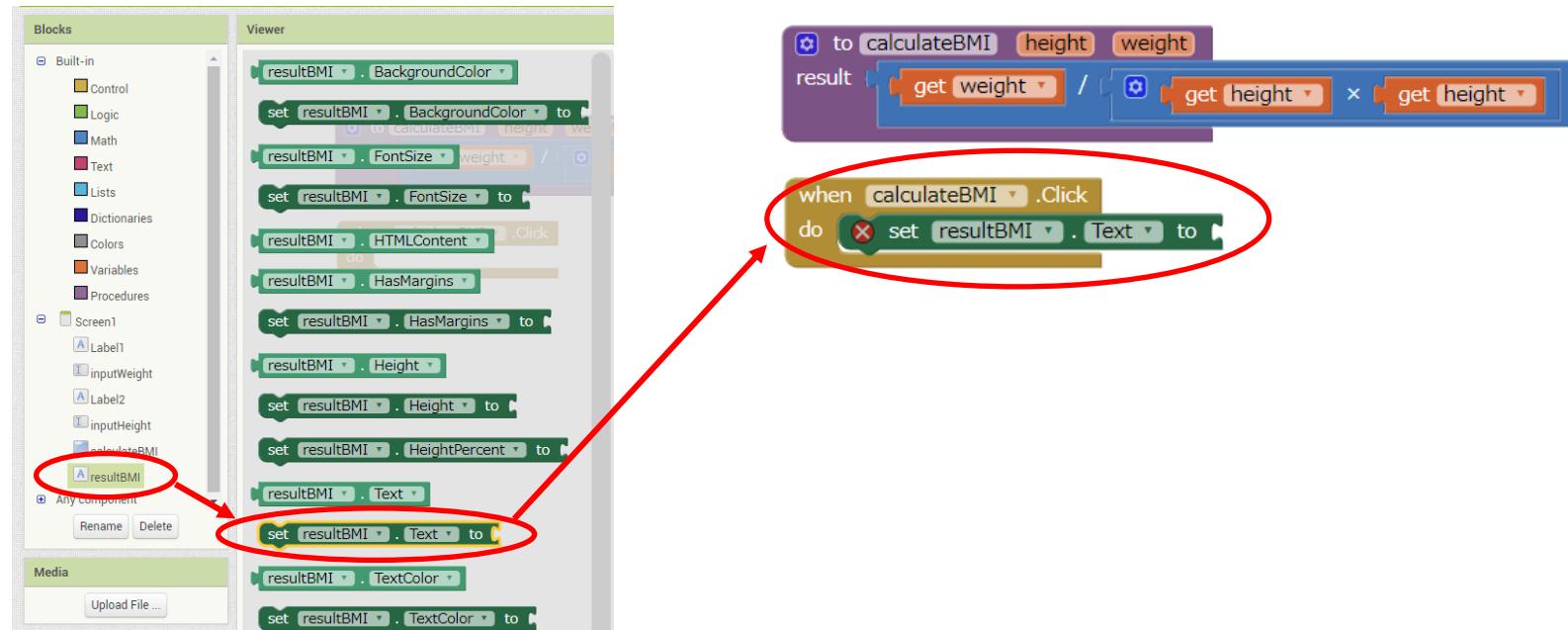
- Hover mouse on argument ‘height’
- Drag the ‘get weight’ block and place it inside the denominator



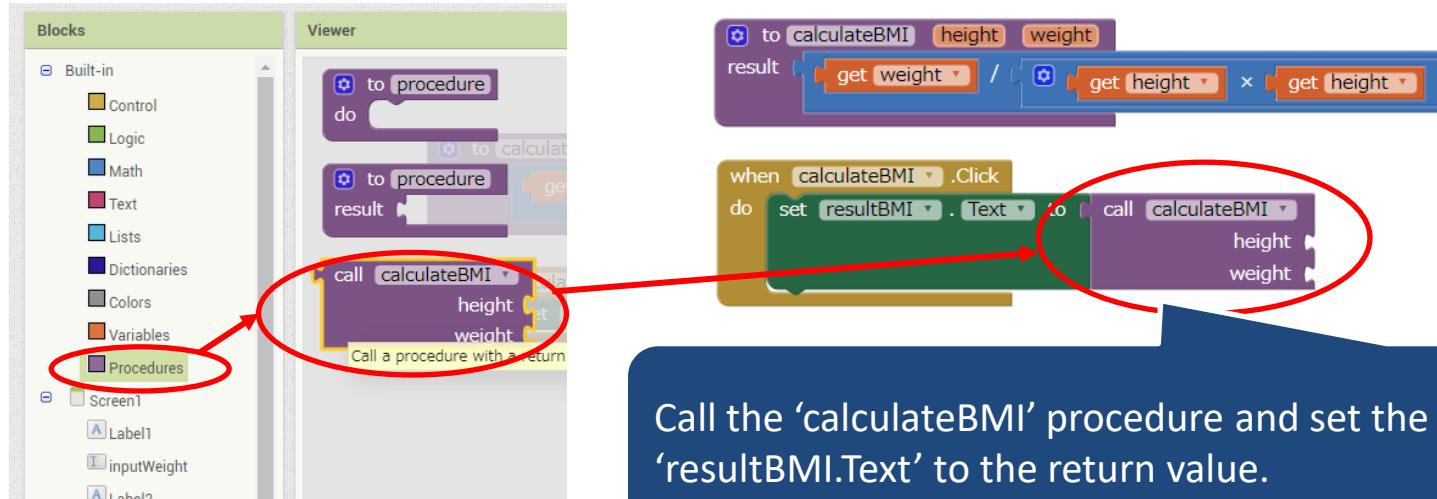
Add click handler to button 'calculateBMI'



Add click handler to button 'calculateBMI'

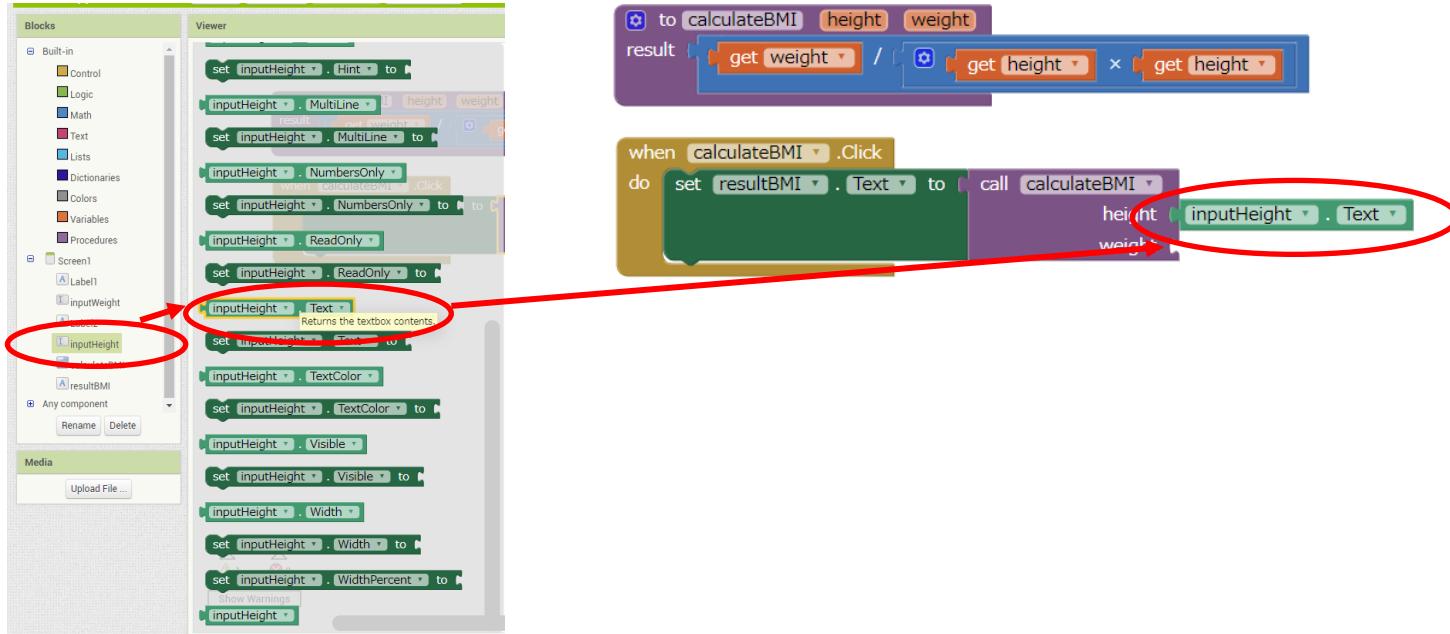


Add click handler to button 'calculateBMI'

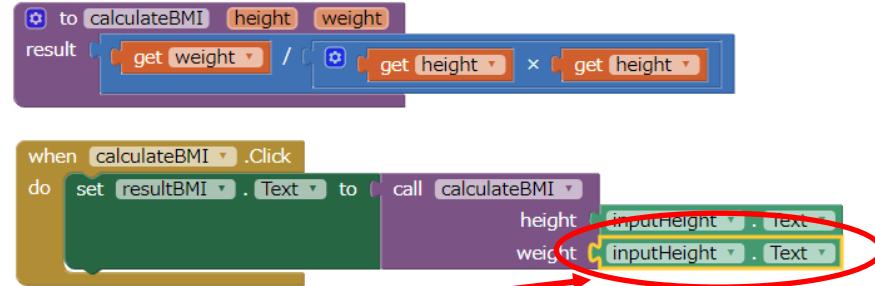
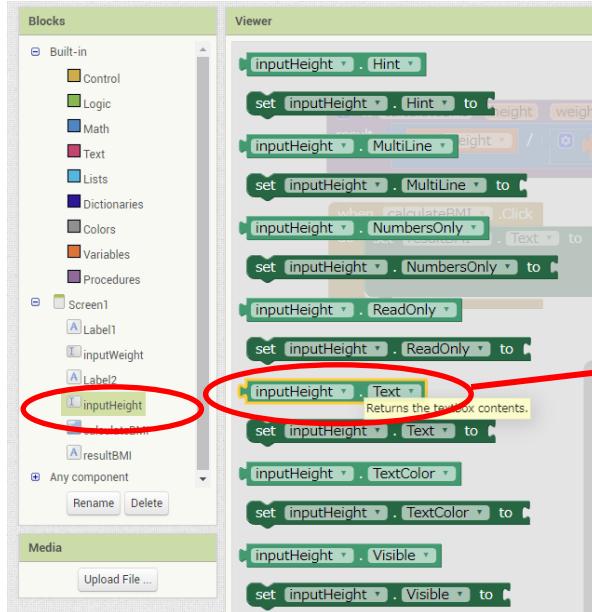


Call the 'calculateBMI' procedure and set the 'resultBMI.Text' to the return value.

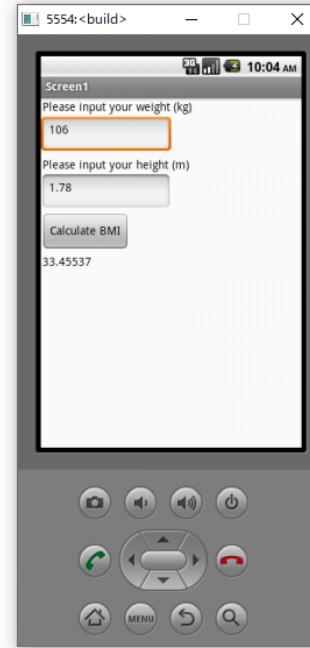
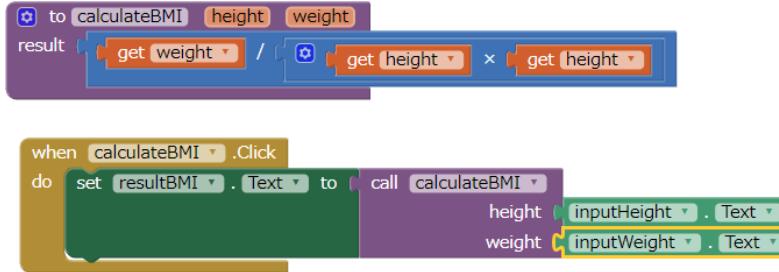
Get argument values from TextBox



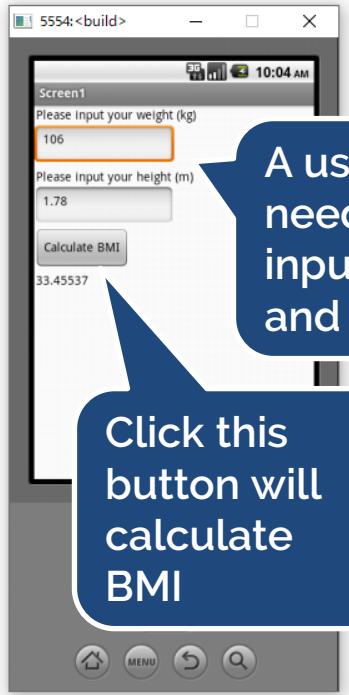
Get argument values from TextBox



Test on Emulator

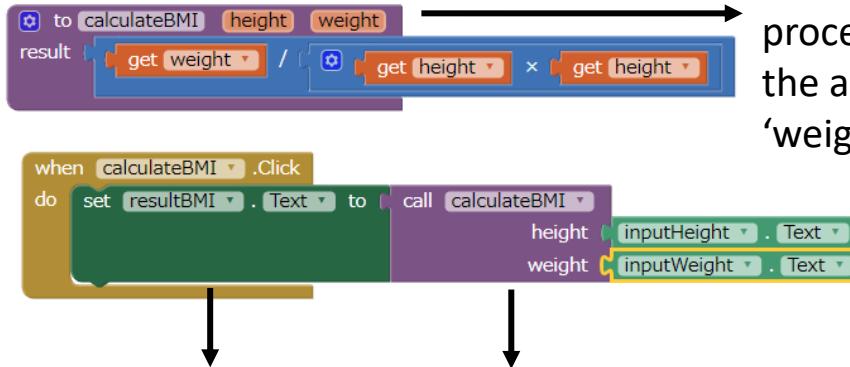


Code Anatomy: BMI Calculator App



A user needs to input weight and height

Click this button will calculate BMI



3. Set the 'resultBMI' text to the result returned from the 'calculateBMI' procedure

2. Call the 'calculateBMI' procedure and pass the two arguments 'weight' and 'height', and return the result value to the block on the left

Define a 'calculateBMI' procedure that takes in the argument 'height' and 'weight' to calculate BMI

1. Set the 'weight' and 'height' arguments to the values that a user input in the textboxes

Assignment

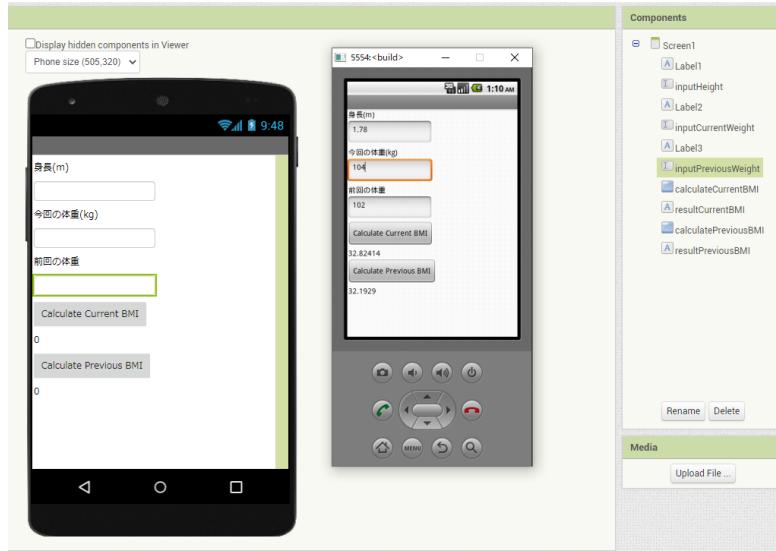
- Complete the hands-on tasks in the tutorial
 - ✓ If you finish all the steps in class, show your Counter App and BMI Calculator App to one of the instructors before you leave
 - ✓ If you cannot finish all the steps, you can work on them after class and show your Apps to one of the instructors in the class next week

Assignment (Optional)

If you have time, why not trying to add more features to the BMI Calculator app!

1. Additional feature 1: calculate previous BMI

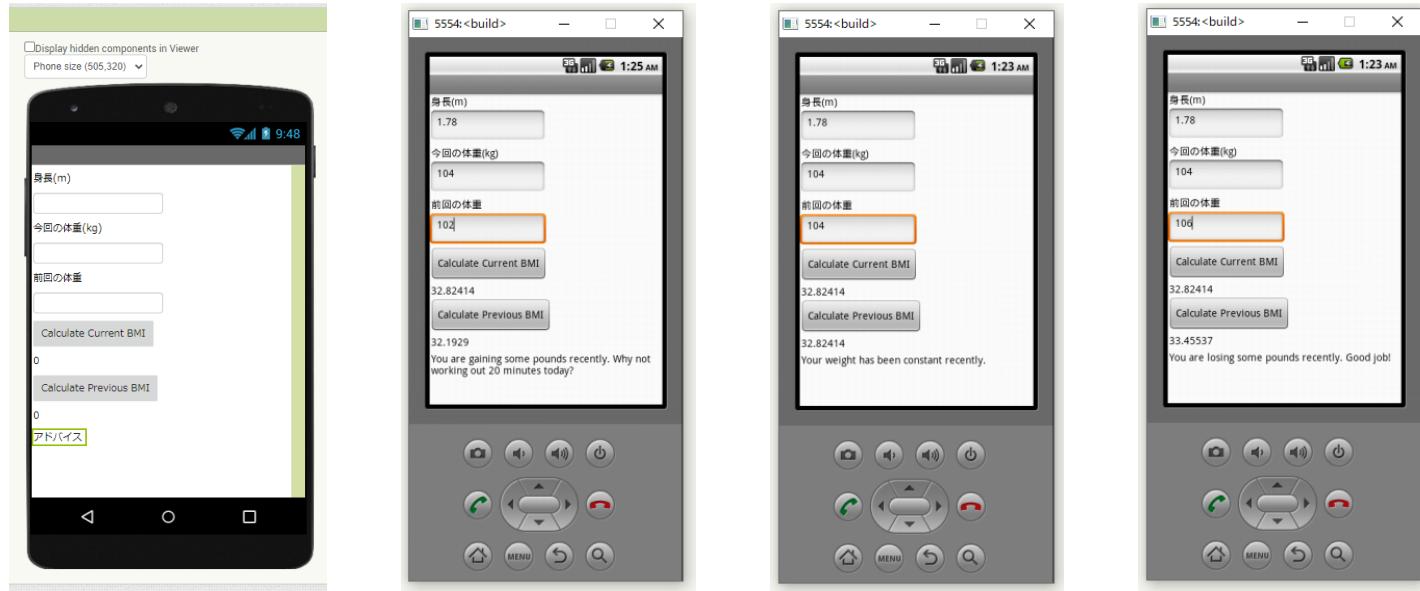
- Add a TextBox that allows user to input their previous body weight
- Add a button 'Calculate Previous BMI', so that when a user clicks this button, the app calculates the previous BMI and show it below the button (you must use the 'calculateBMI' procedure in the click handler)



Assignment (Optional)

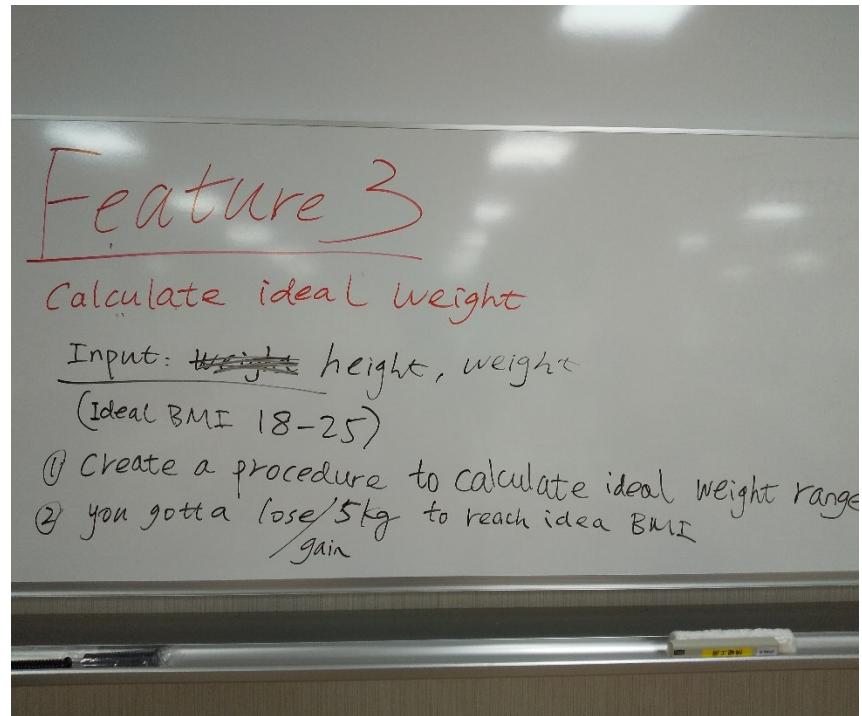
2. Feature 2: show a simple advice based on the comparison of current weight and previous weight

(Hint: you need to use the ‘if-then-else if-then-else’ block in the drawer of ‘Control’)



Assignment (Optional)

3. Feature 3: Calculate the ideal weight range
 - (1) Create a procedure to calculate ideal weight range based on a user's height
 - (2) Provide advice like 'another 5 kg to lose/gain before reaching the idea BMI'





京都先端科学大学