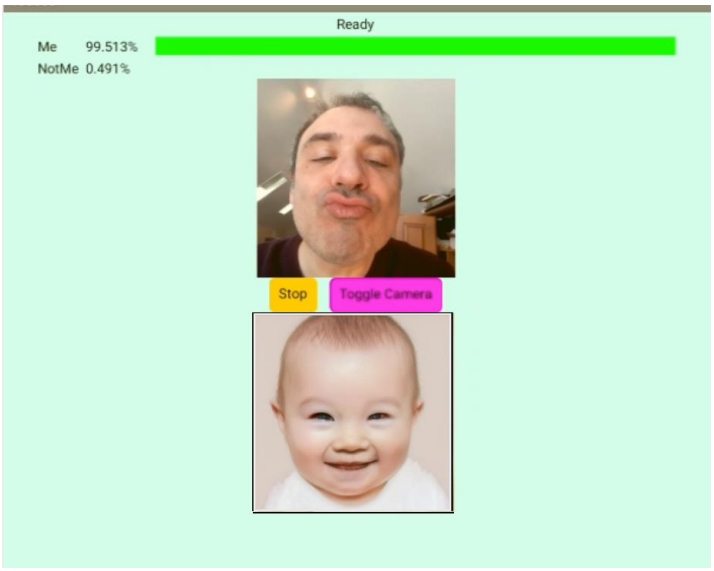
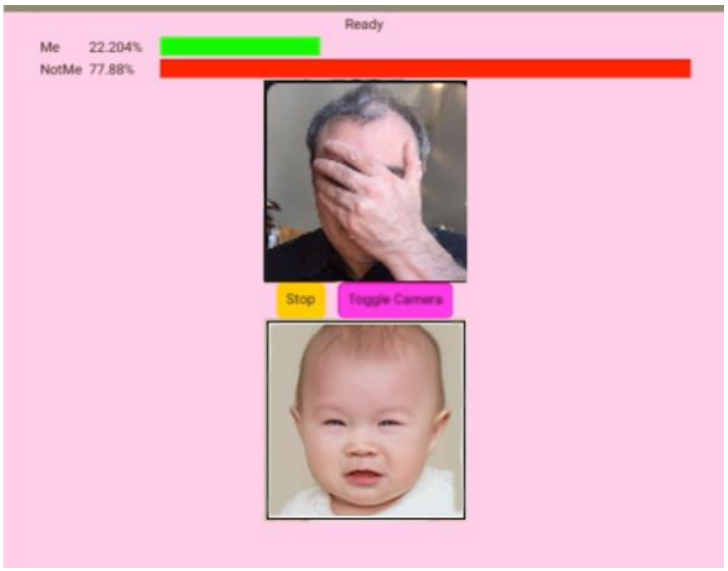
PICaboo: Part 2





Game of Peekaboo

Very young babies do not quite understand yet how to make sense of the visual world around them. A cute game that parents often play with their very young babies in the US is called Peekaboo where a parent hides her face (with her hands or with a pillow etc.) and "disappear" to the surprise of the baby and then suddenly show her face saying "Peekaboo, I see you!" much to the delight of the baby, who is astonished to see the parent suddenly appear out of "nowhere". Try the PICaboo app that your teacher will provide you. Show and hide your face with your hand and see how the baby in the app responds.

PICaboo: Part 2

START HERE

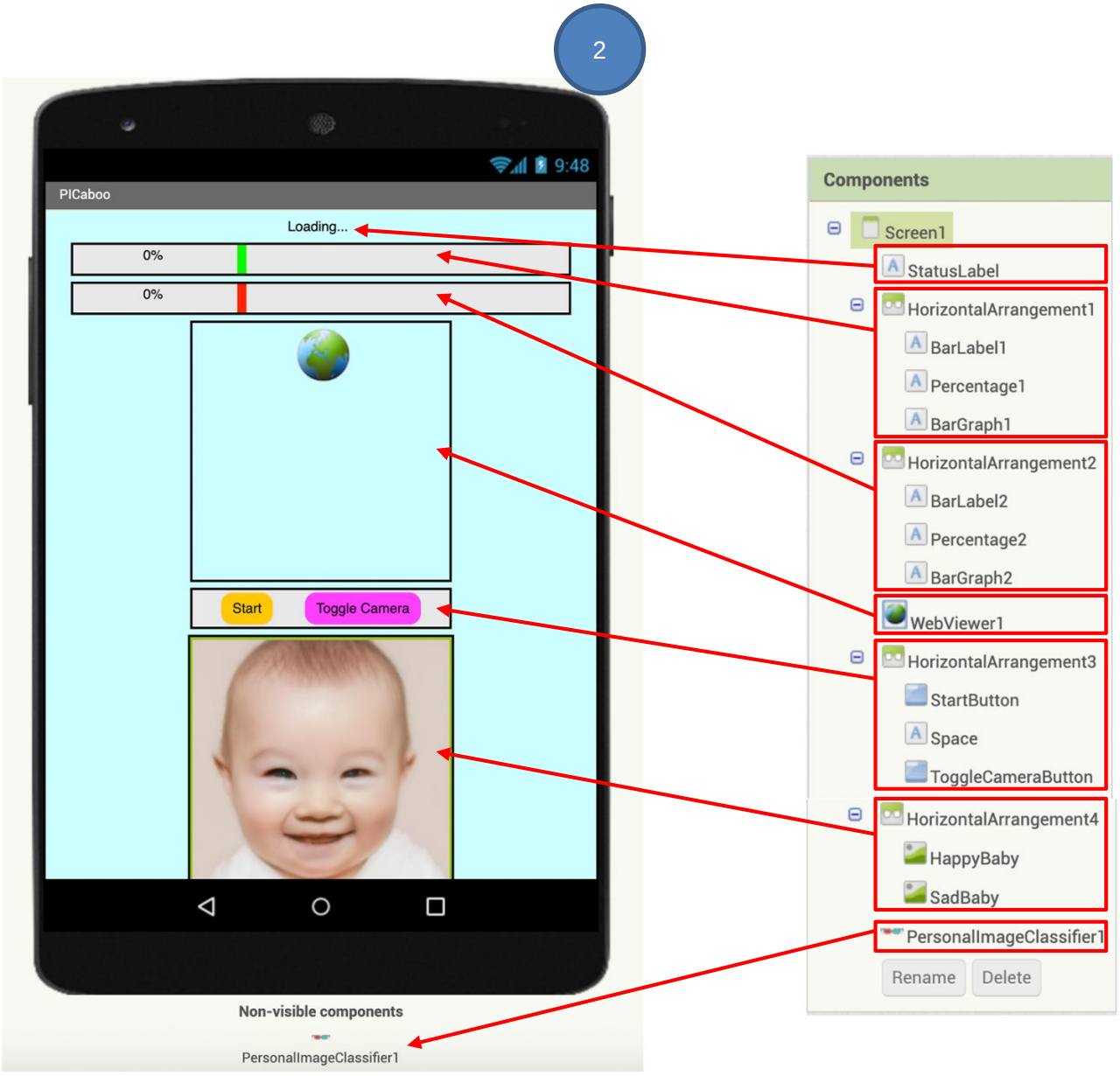
In this lesson, you will make a Peekaboo game app with your Personal Image Classification (PIC) model where the baby in the app will smile if your face is shown and will cry if your face is hidden.

❑ Go to **http://ai2.appinventor.mit.edu** and open the template file provided by your teacher.

1

❑ Below are the components already included in the Designer. Review what each component is used for.

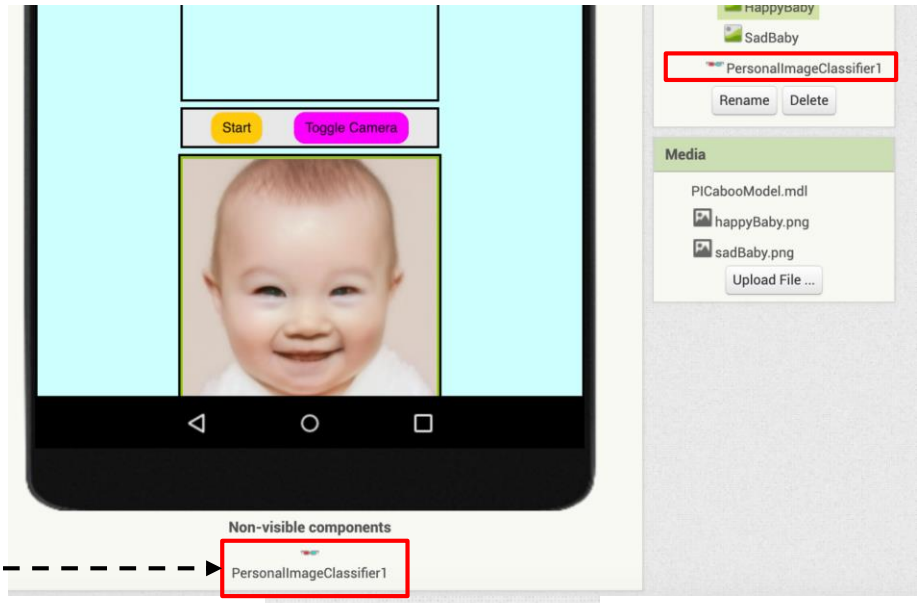
2



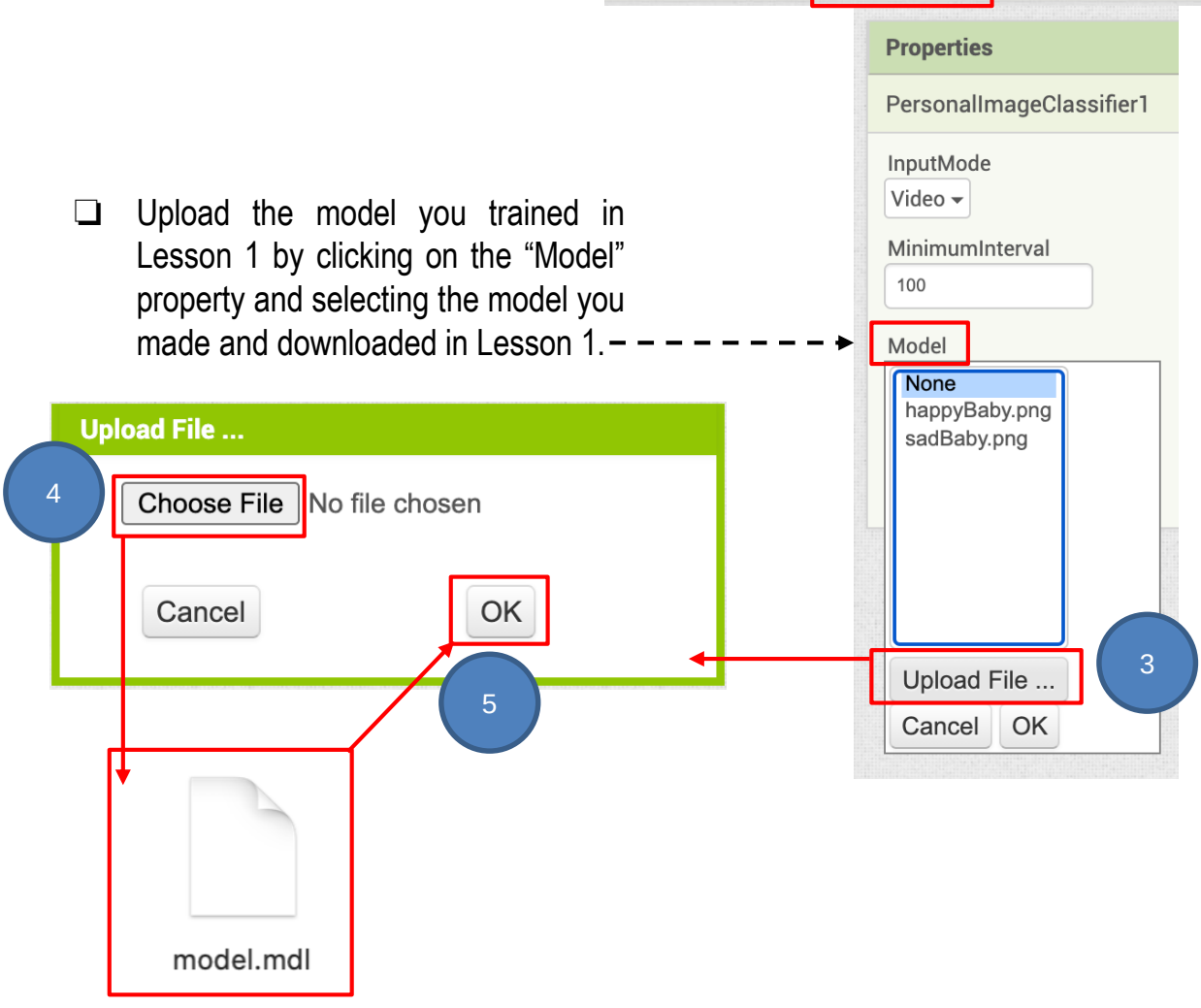
PICaboo: Part 2

ADD MODEL TO PERSONALIMAGECLASSIFIER

- Look at the non-visible component **PersonalImageClassifier1**. This is the component that will classify your poses based on your trained model.



- Upload the model you trained in Lesson 1 by clicking on the "Model" property and selecting the model you made and downloaded in Lesson 1.

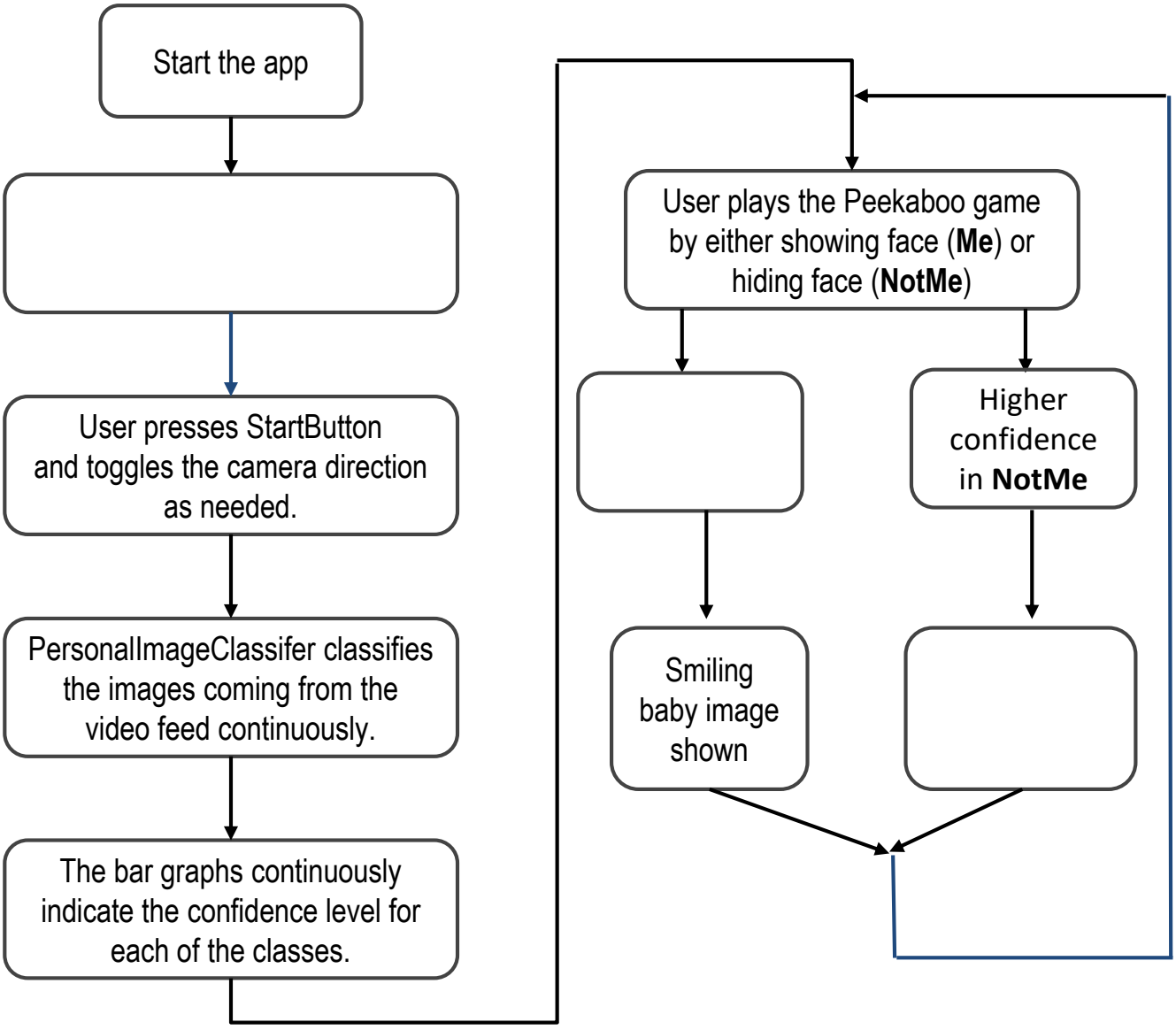


PICaboo: Part 2

SEQUENCE OF EVENTS

Follow the sequence of events in the running of the PICaboo. Fill in the missing blocks with the letter corresponding to the correct event. 6

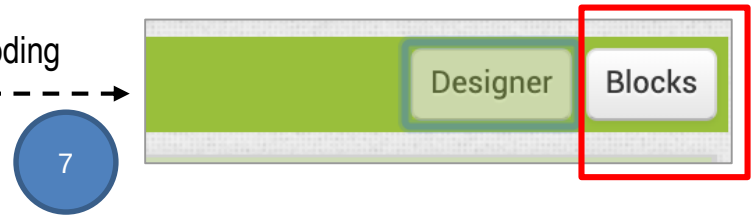
A. The app shows “Ready”	B. Higher confidence in **Me**	C. Crying baby image shown



PICaboo: Part 2

REVIEW GIVEN BLOCKS

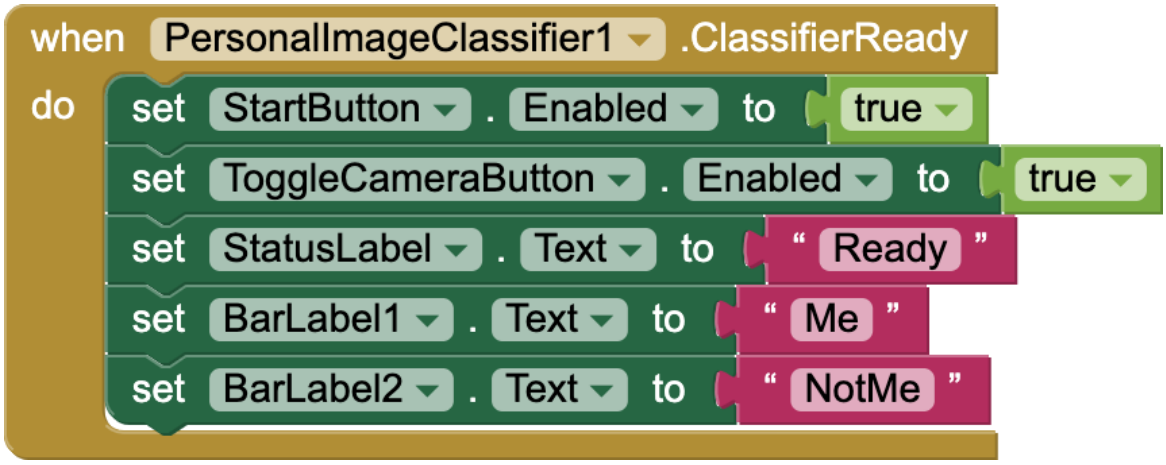
❑ Now switch to the Blocks Editor to start coding the behaviour for the app.



❑ Look at the some of the code blocks that are already included with your partner. These blocks set up the **PersonalImageClassifier** when the app first starts.

In case the **PersonalImageClassifier** returns an error, the Error event block displays the error in the **StatusLabel**.

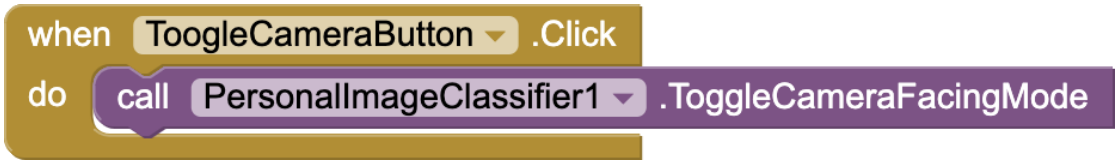
The **when PersonalImageClassifer.ClassifierReady** block is called when the classifier is ready to start classifying images. **StartButton** and **ToggleCameraButton** are enabled. "Ready" is displayed in the **StatusLabel. BarLabel1** and **BarLabel2** are assigned the names of the models labels (**Me** and **NotMe**). These are the classes you trained in the model.



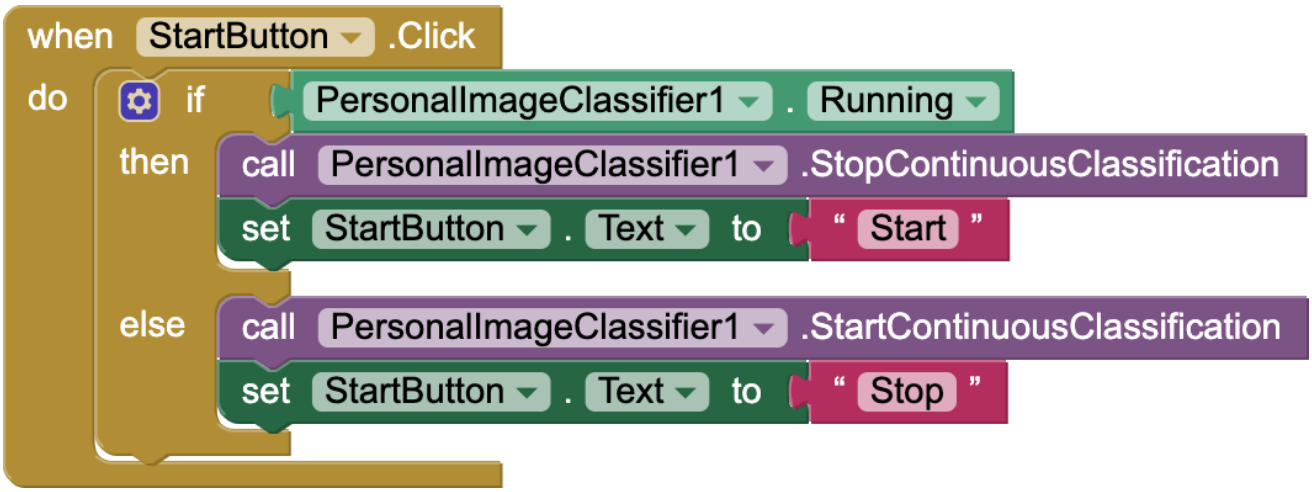
PICaboo: Part 2

REVIEW GIVEN BLOCKS

- When **ToggleCameraButton** is pressed, the direction the camera is facing is switched from Front to Back and vice versa.



- **StartButton** is a start/stop toggle button and when it is clicked, if the **PersonalImageClassifier1** has been running, its continuous classification process is stopped and otherwise this process is started. Button texts are also changed from Start to Stop or vice versa accordingly.

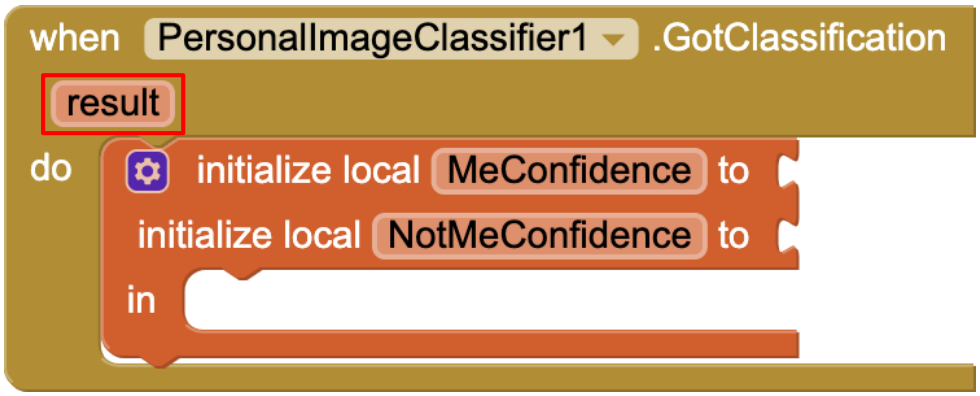


PICaboo: Part 2

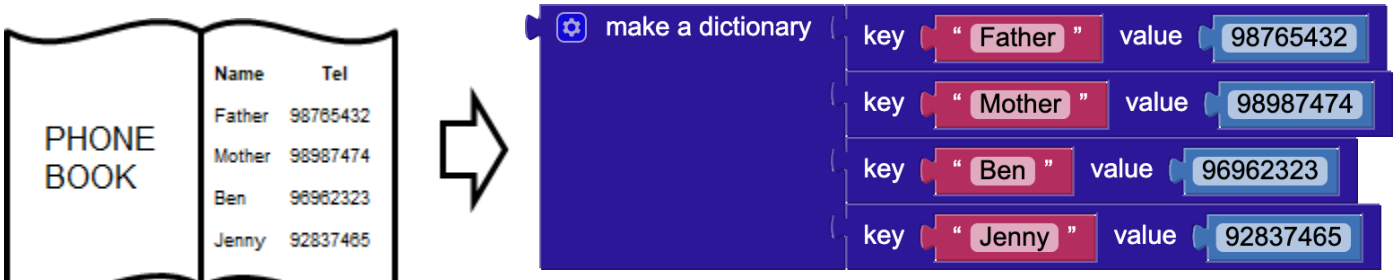
RESULT OF THE IMAGE CLASSIFICATION

Before coding the PICaboo app, let's see how the **result** of classification from the component **PersonalImageClassifier1** is received and used.

- When the **PersonalImageClassifier1** produces a classification for what it sees, the **when PersonalImageClassifier1.GotClassification** event handler block is called. In this lesson your main task is to complete the code block set for this event handler.



- The **result** lists the classifications in the form of dictionary data structure, which is simply a list of key-value pairs. Below is an example:



The data structure of these key-value pairs can be represented with the code blocks of **Dictionary**.

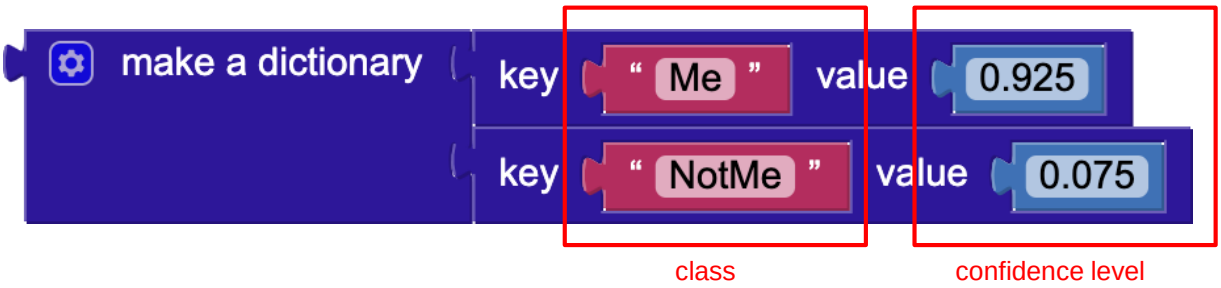
- In the above example of key-value pairs, "Name" is the key, "Tel" is the value, and the phonebook is the dictionary.

- In the PICaboo app, what are the list of key-value pairs returned in the **result** by the **when PersonalImageClassifier1.GotClassification** block?

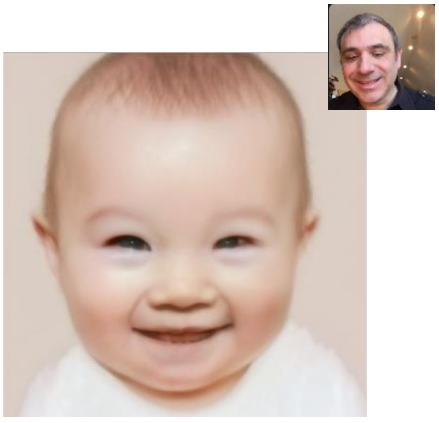
PICaboo: Part 2

DICTIONARY DATA STRUCTURE

- The **result** returned by PersonelImageClassifier1 lists the name (key) of two classes ("**Me**" and "**NotMe**") with their confidence levels (value). This data structure can be represented with code blocks as below. It is important to note that the key-value pairs in **result** are listed in the order of highest to lowest confidence levels.



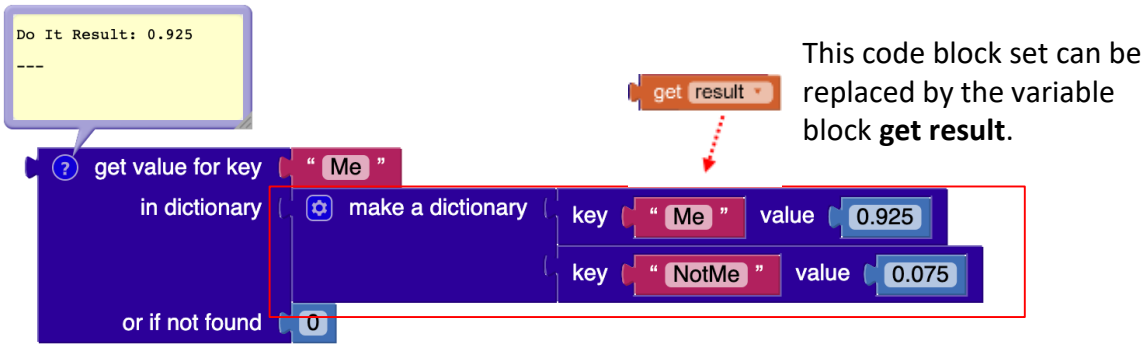
- For example, the **result** could be composed of the class "**Me"** with the high confidence level of 0.925 (i.e. 92.5%) and the class "**NotMe**" with a low confidence level of 0.075 (i.e. 7.5%). These confidence level numbers have been computed by the model you created during the training phase. At this instance the baby in the PICaboo app would be smiling as the baby (i.e. the machine) is confident that he is seeing your face.



PICaboo: Part 2

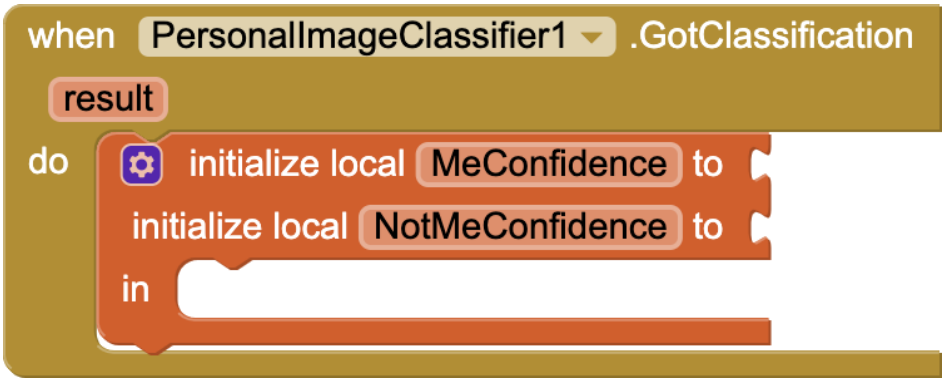
RESULT OF THE IMAGE CLASSIFICATION

❑ In order to find out the value of a particular key from the **result**, the **get value for key** block is used. In the example below, if you searched for the key "**Me"** in the given dictionary, you would get 0.925 which is its associated confidence level (value). If no key was matched, you would get the value "0".



❑ The **when PersonalImageClassifier1.GotClassification** event handler block starts by initializing two local variables. Local variables can *only* be used inside the code block where they are defined:

- **MeConfidence**: is the confidence level for the "**Me"** class (showing face)
- **NotMeConfidence**: is the confidence level for the "**NotMe"** class (hiding face).

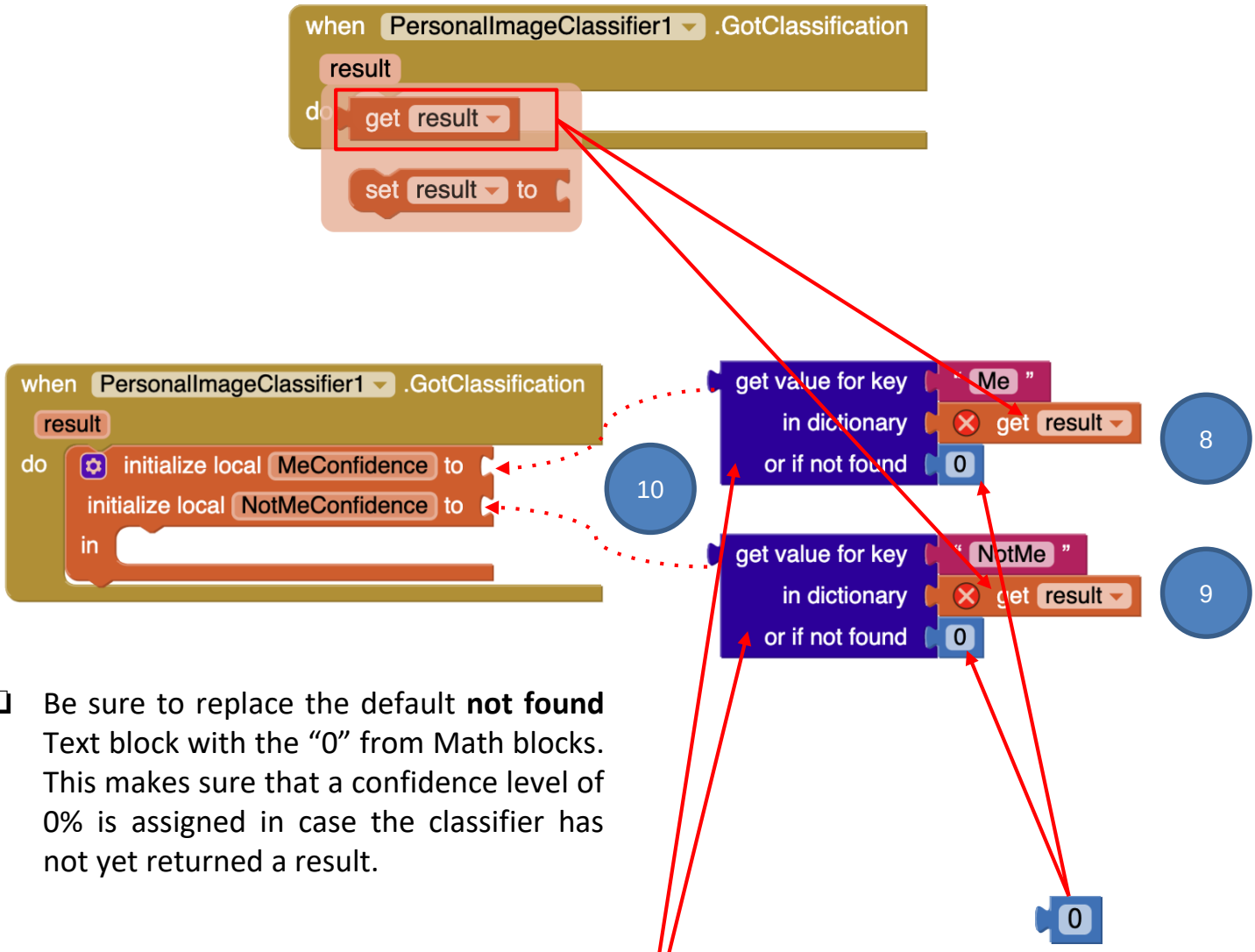


You will soon see how these variables will be used.

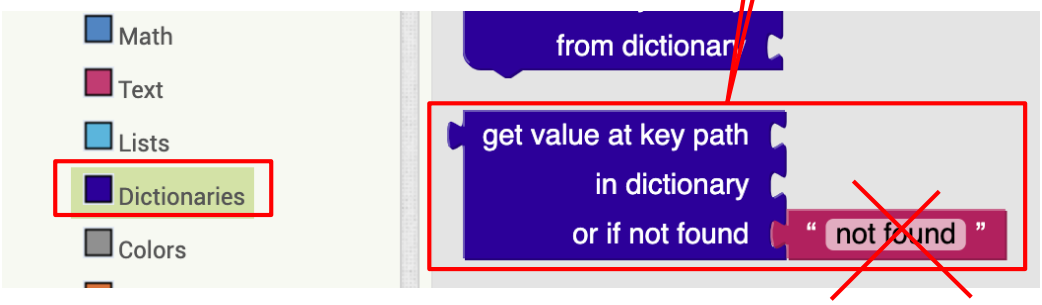
PICaboo: Part 2

INITIALIZING LOCAL VARIABLES

- ❑ Initialize the local variables as shown. If you mouse over the parameter **result**, you will be able to get the needed block.



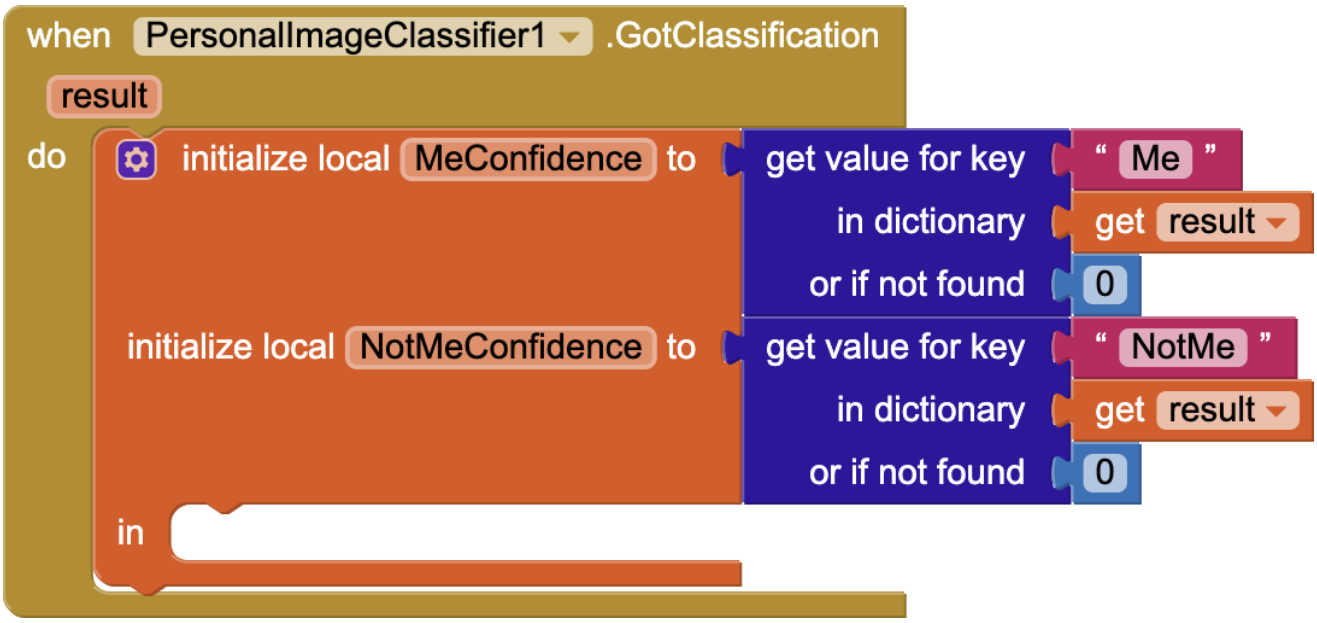
- ❑ Be sure to replace the default **not found** Text block with the "0" from Math blocks. This makes sure that a confidence level of 0% is assigned in case the classifier has not yet returned a result.



PICaboo: Part 2

INITIALIZING LOCAL VARIABLES

❑ When you are done, the local variables should be initialized as shown below.

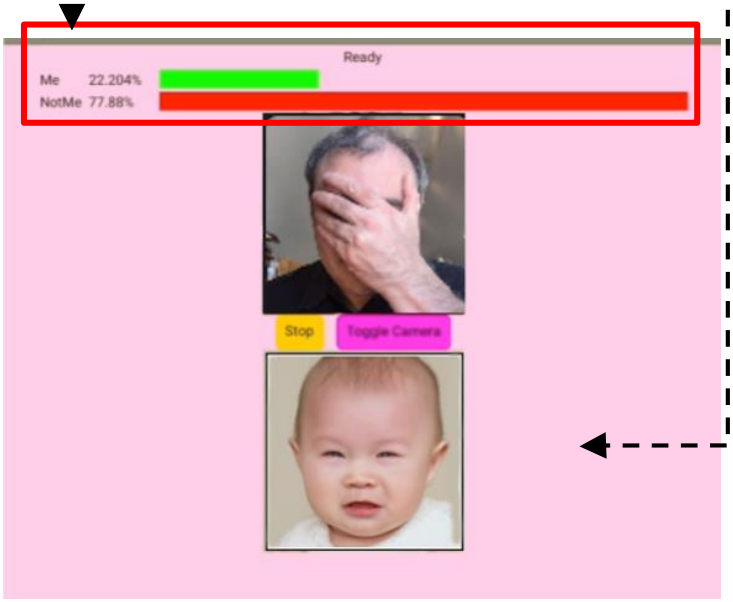


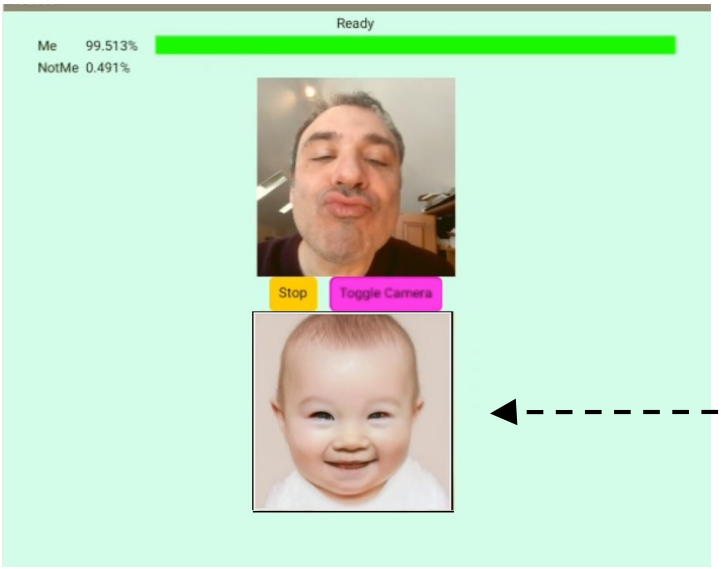
PICaboo: Part 2

THE SEQUENCE OF EVENTS

❑ When the classifier gets its classification…

- Set the **Percentage1** text and **BarGraph1** width % correctly.
- Set the **Percentage2** text and **BarGraph2** width % correctly.
- If the classification with higher confidence is "**Me**" (i.e. the machine is highly confident that it is seeing your face) then show the happy baby image and hide the sad baby image and make the **Screen1** background colour in light green.
- Otherwise, show the sad baby image and hide the happy baby image and make the **Screen1** background colour in light pink.

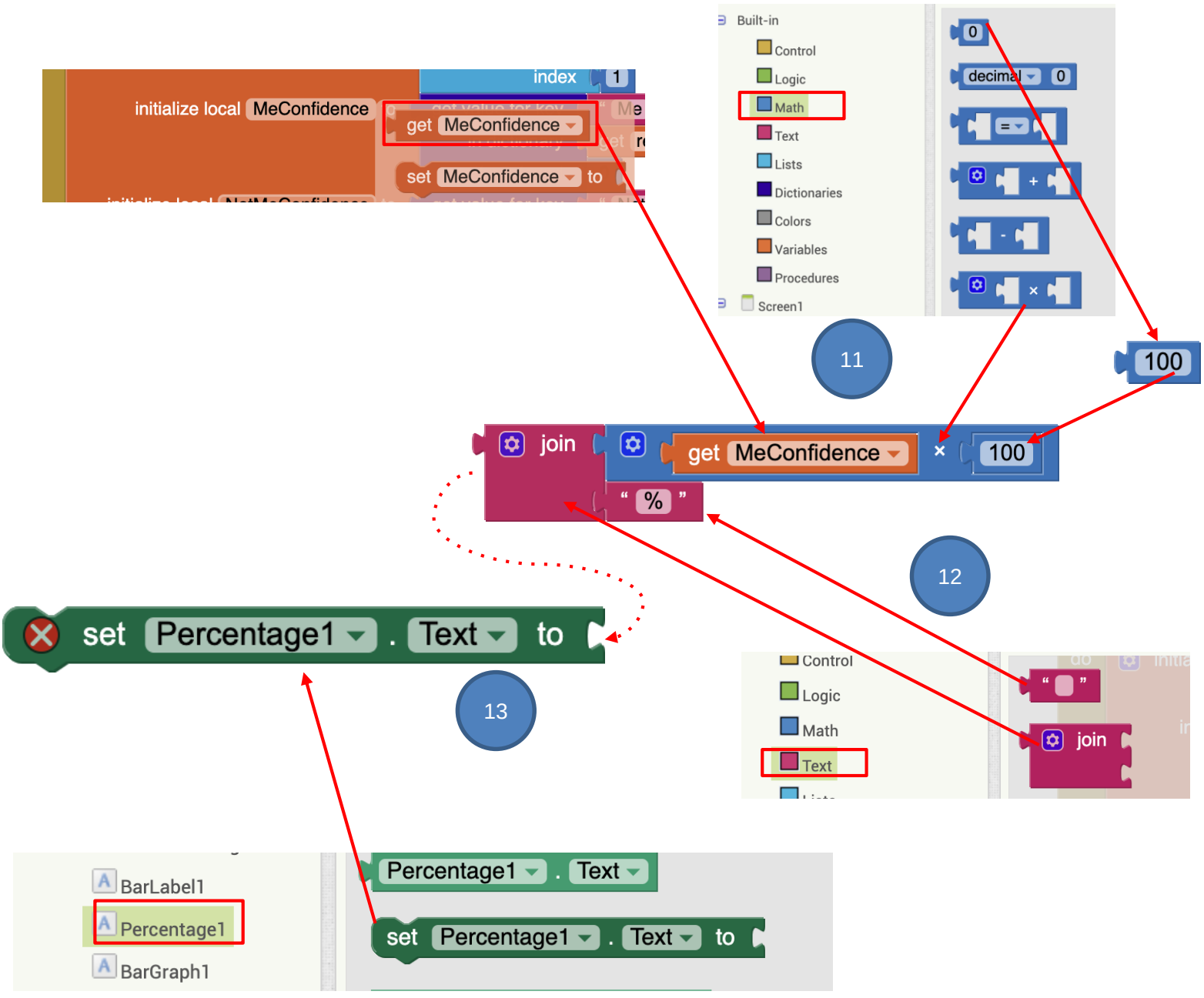




PICaboo: Part 2

GETTING THE CONFIDENCE LEVEL AS A PERCENTAGE

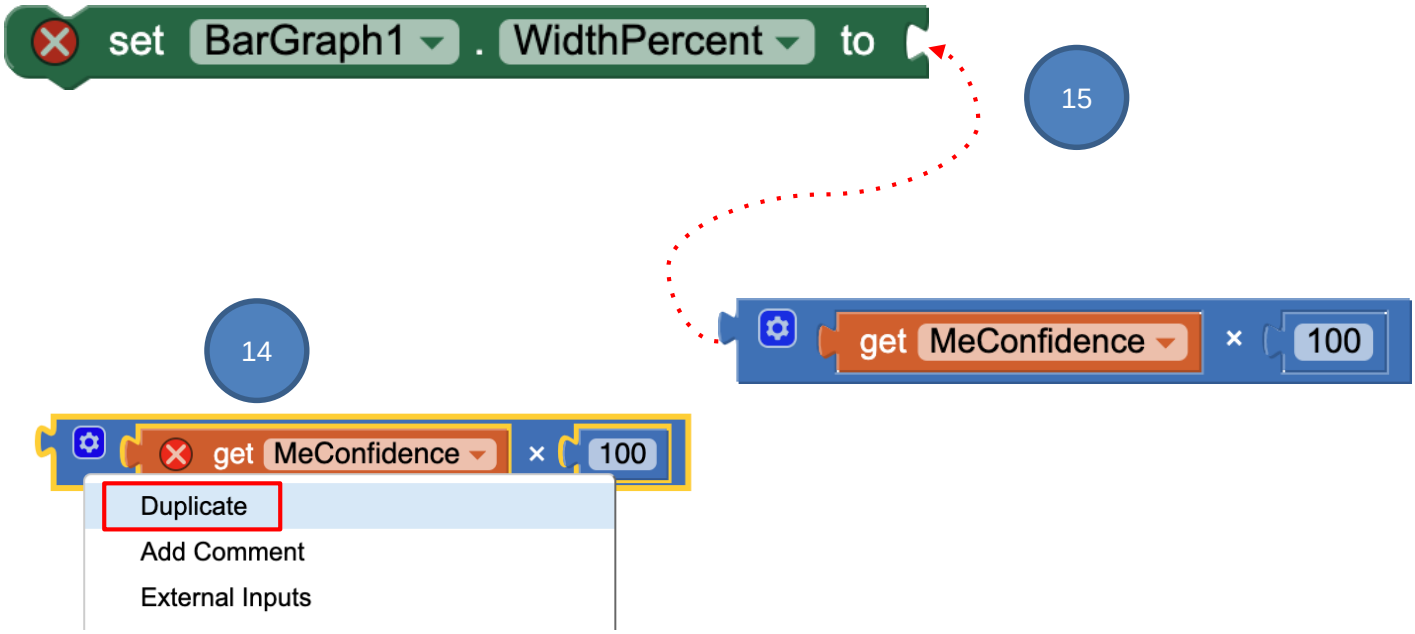
- ❑ First set the **Percentage1** text to the confidence level of the "**Me**" class. As this value is given as a decimal between 0 and 1 you will need to convert it to a percentage (by multiplying with a 100) and add the "%" symbol. For example if the confidence level was 0.75, this proper percentage would be 75 %.



PICaboo: Part 2

DISPLAYING THE CONFIDENCE LEVEL AS A BAR GRAPH

- **BarGraph1** is a coloured label the width of which will graphically represent the confidence level of the "**Me**" class. Set its WidthPercent accordingly. You can duplicate part of earlier code you wrote.

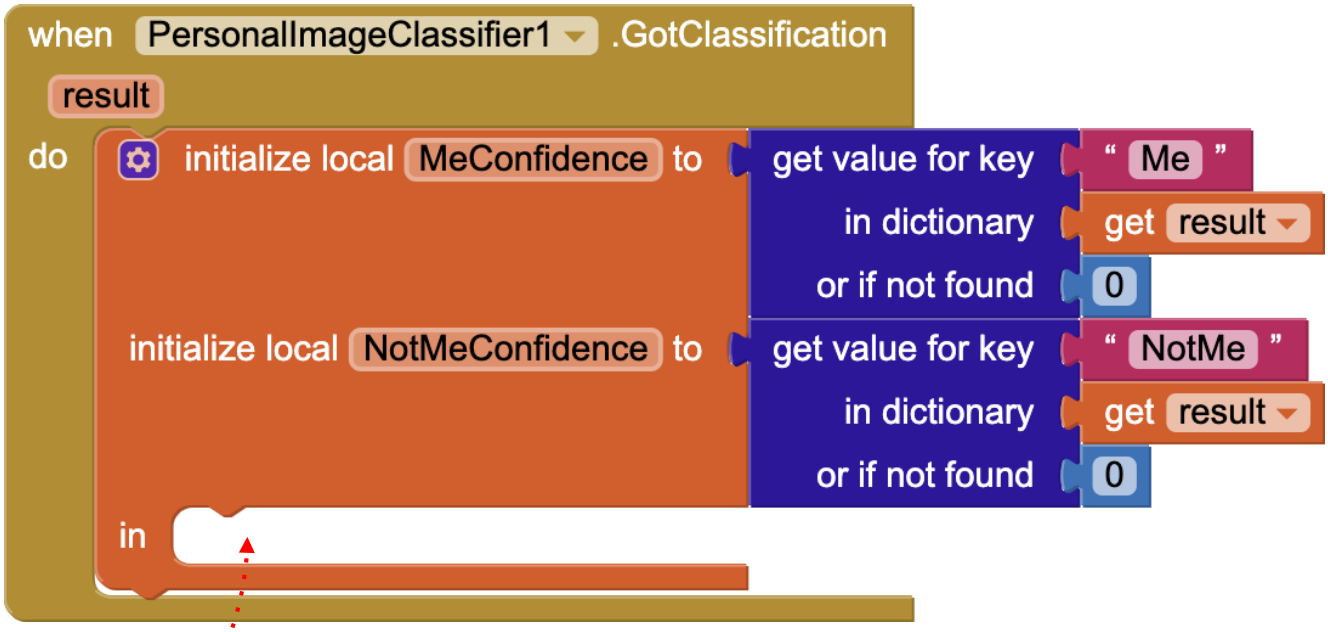


- Do a similar thing for **Percentage2** and **BarGraph2** to get the following. Be sure to use the **NotMeConfidence** local variable for the latter cases.

PICaboo: Part 2

DISPLAYING THE CONFIDENCE LEVEL AS A BAR GRAPH

❑ Snap these code blocks in place as shown below.

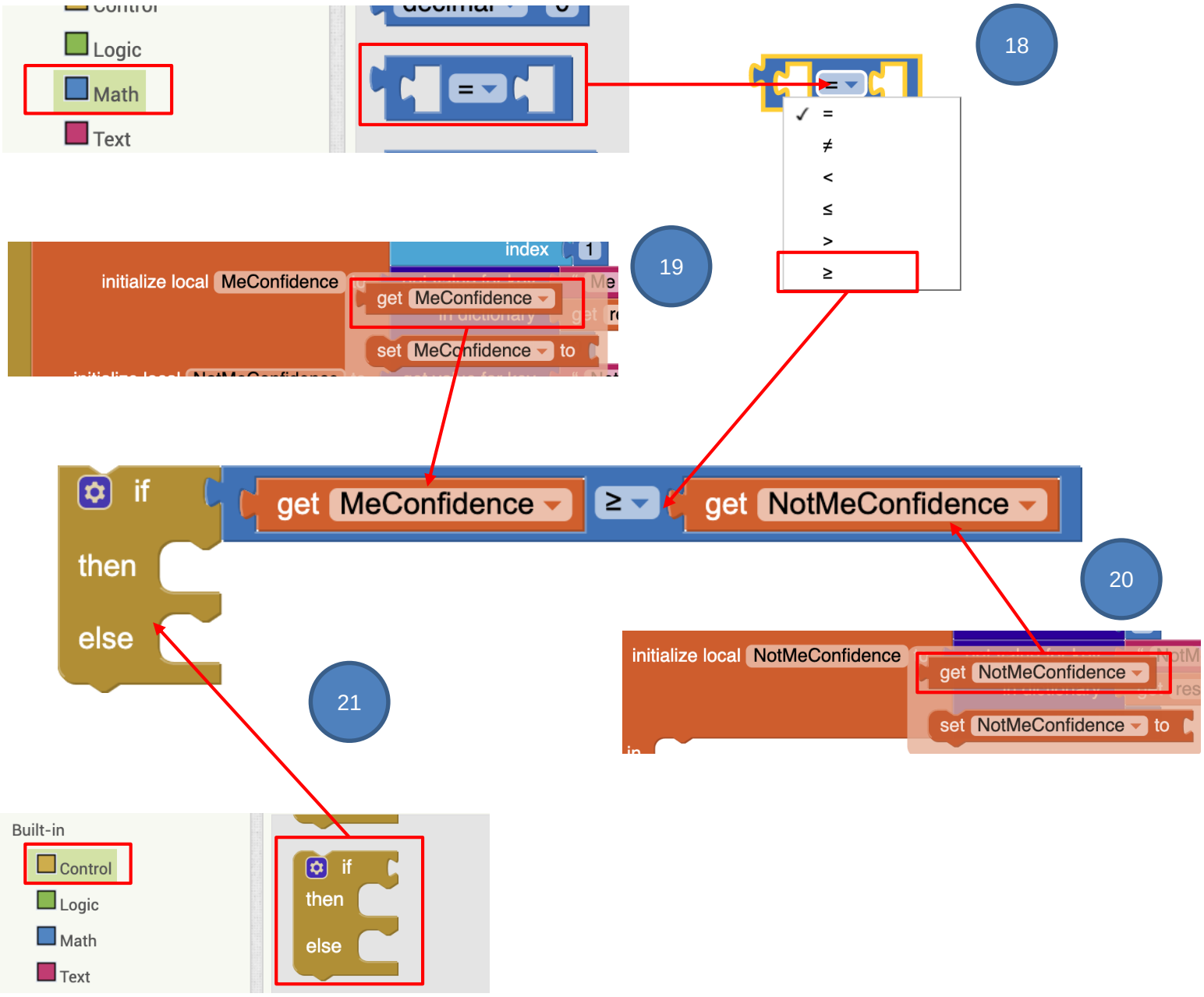


17

PICaboo: Part 2

THE CONDITIONAL STATEMENT

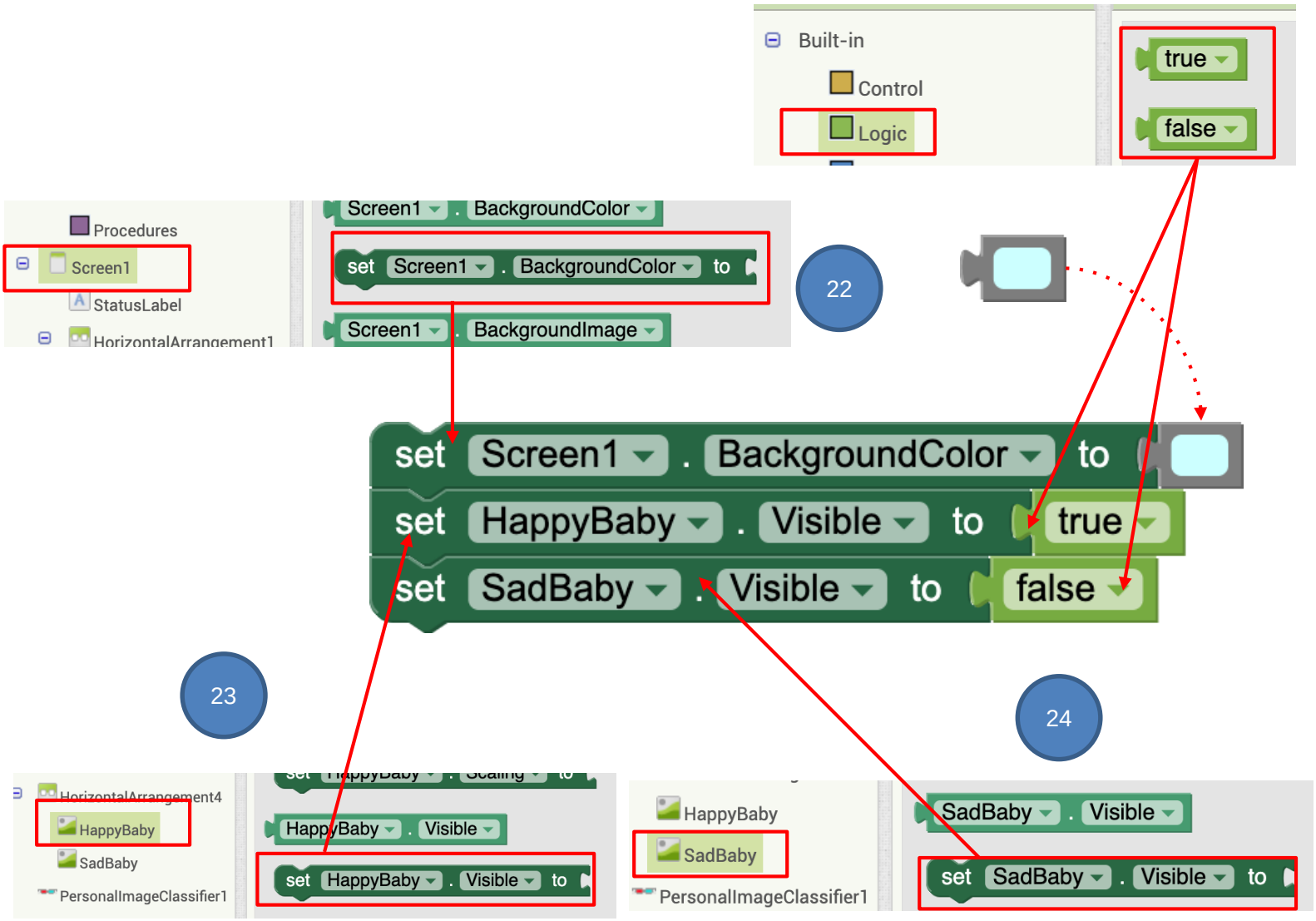
- ❑ Now build the **if-then-else** conditional block set which will test if the class with higher level of confidence is the "**Me**" class, representing the machine recognizing you showing your face with a higher level of confidence.



PICaboo: Part 2

DISPLAYING A HAPPY BABY

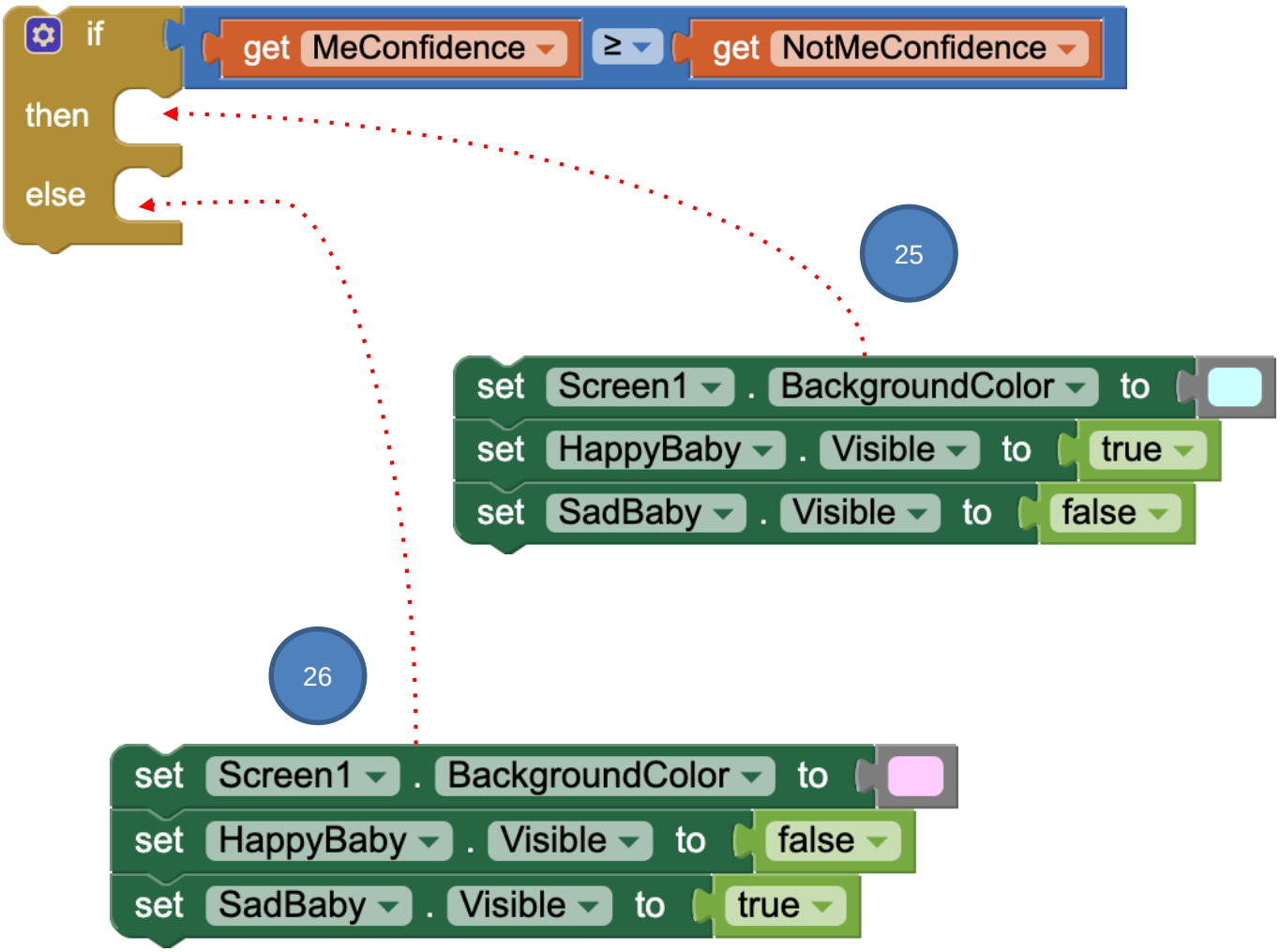
- Now we build the code blocks for the **then** part of the **if then else** block set. You are given a light green colour block in the template which you can use.



PICaboo: Part 2

DISPLAYING A SAD BABY

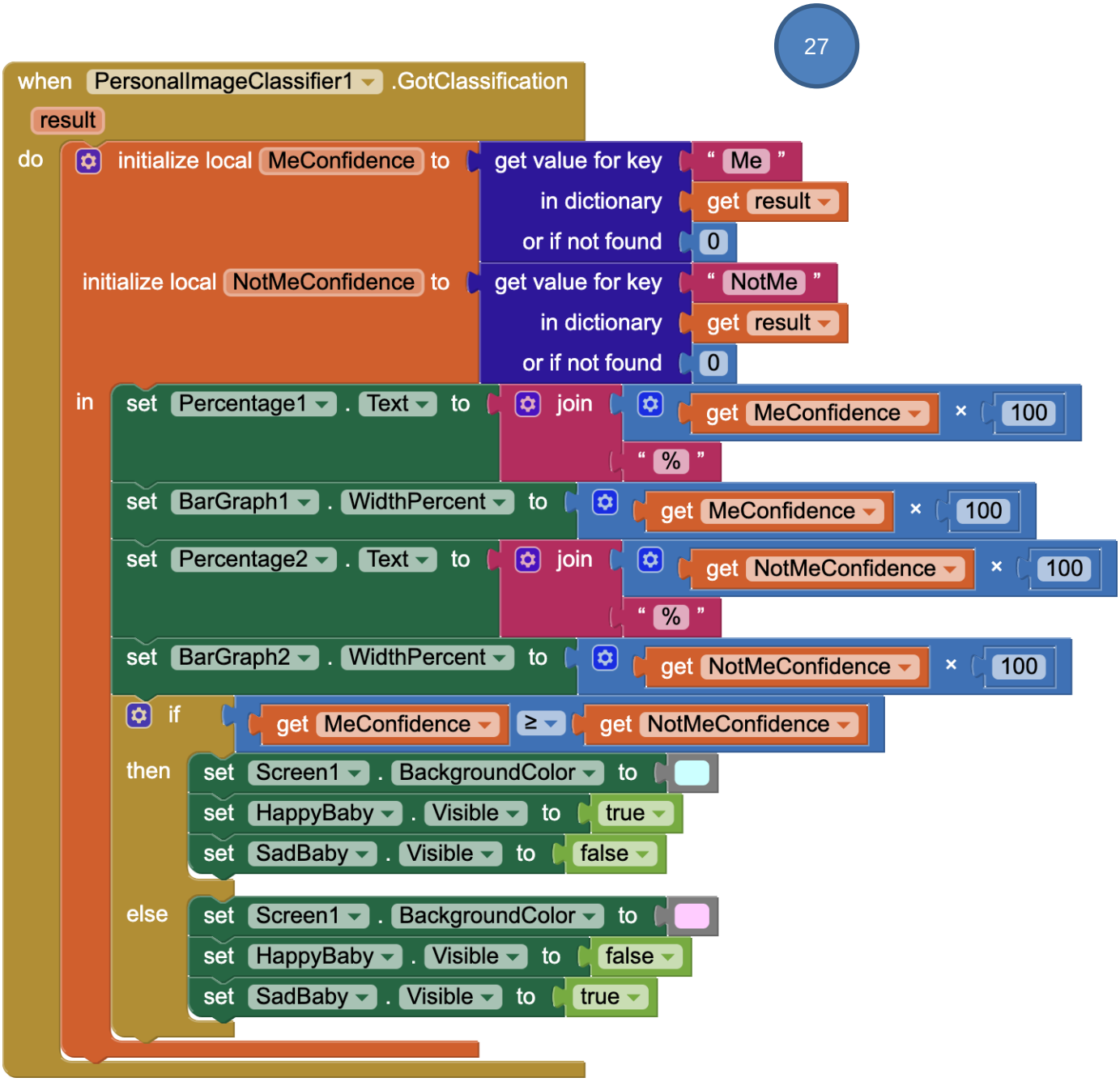
❑ Now similarly build code blocks for the **else** part of the **if-then-else** block and snap these code blocks to where they belong as shown below. You are given a light pink colour block in the template which you can use.



PICaboo: Part 2

RESULT OF THE IMAGE CLASSIFICATION

- ❑ When you complete coding, the **GotClassification** event handler block set should look like this.



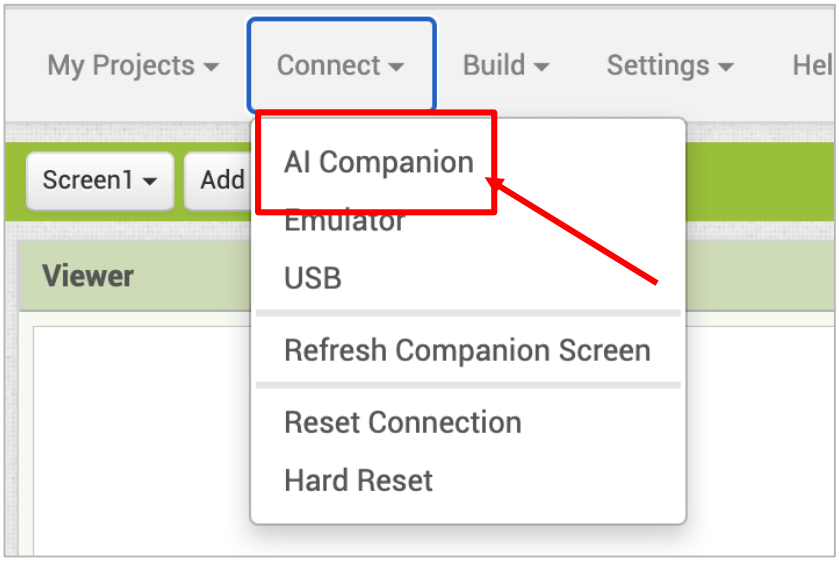
PICaboo: Part 2

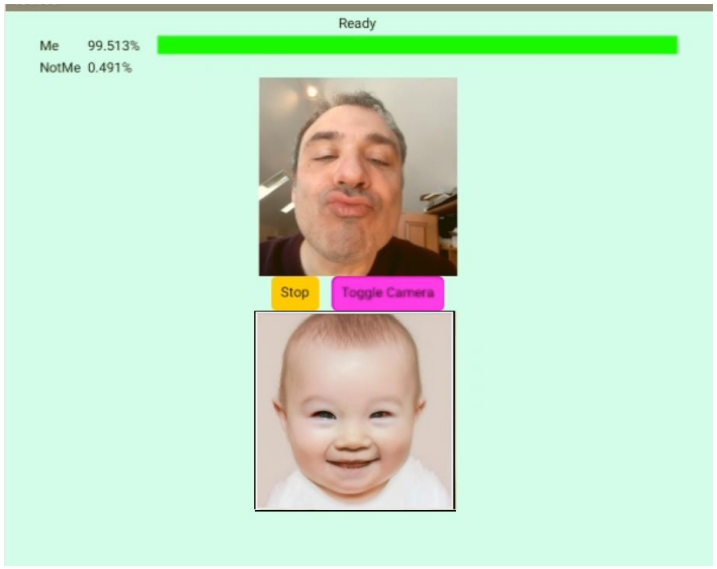
TESTING AND DEBUGGING

- ❑ It is time to test the Peekaboo game out! Connect to the AI Companion, and see if you can play Peekaboo with the baby? Remember to crack your finger open slightly when covering your face so that you can see what is going on. Does the baby smile when the baby can see your face and cry when you hide your face? What happens when you wave your hand up and down in front of your face repeatedly showing and hiding your face? What happens then to the confidence levels?

- ❑ Congratulations, you are done!

28





PICaboo: Part 2

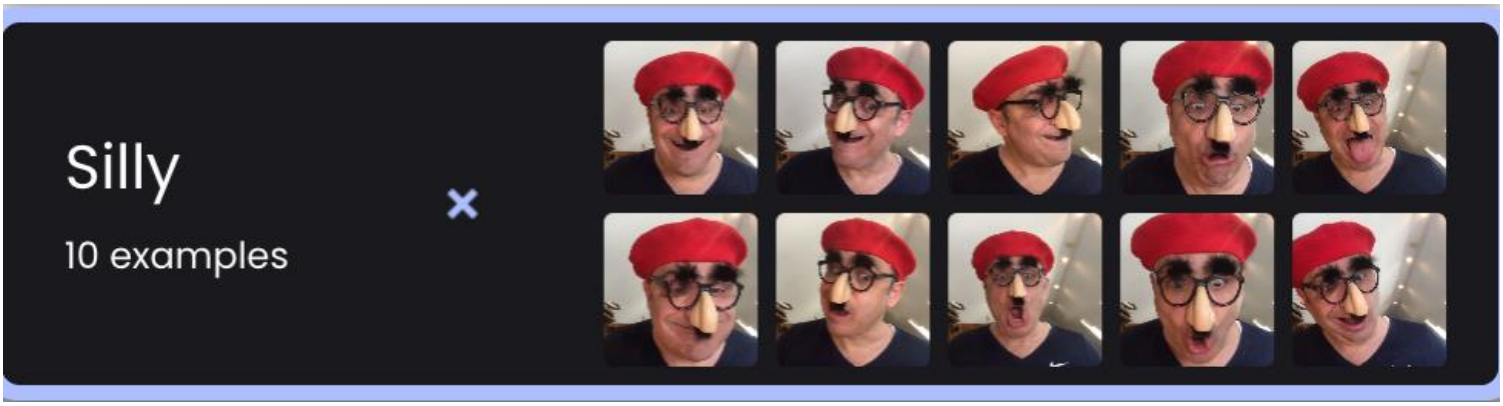
EXPAND YOUR PROJECT

Here are some ideas about how to expand your project. Explore them or come up with your own.

❑ Retrain your model so that in addition to recognizing you showing your face (“Me”) and you hiding your face (“NotMe”), it recognizes a third facial posture, such as perhaps you making a silly face (“Silly”).

- Save a copy of your original code for the PICaboo app and create a copy to work further on.

- Upload the new and expanded model and change the app code so that there are now three bar graphs for the confidence levels of three labels and three images of the baby image to respond to each case (happy baby, sad baby and laughing baby for when you make the “silly” face).



PICaboo: Part 2

EXPAND YOUR PROJECT

❏ A really ambitious class project is to make an app for your teacher that automatically takes attendance using Personal Image Classifier (PIC).

- In the PIC trainer, create a label for each student in the class (such as “Eva”, “Phillip”, “Andrea”, etc.) and for each label (i.e. student) take 25 images of that student from various angles.

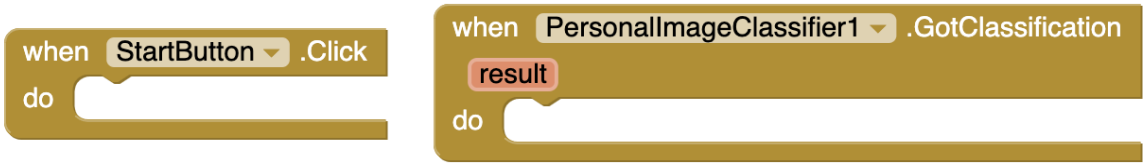
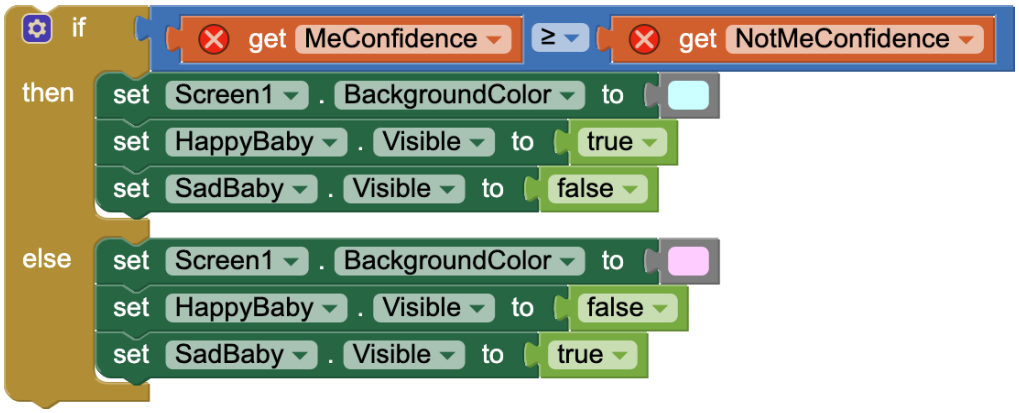
- Train the classifier on the images. This may take a while as there will be so many images to process.
- Then test your model to see if it correctly identifies randomly selected students.
- If the model is accurate and you build a good app, then all your teacher would have to do, at the start of the class, is to have each student pass by a mobile device running this app and when a student is recognized with a high level of confidence, the student is marked as present.
- It may be wise to make a simplified version of this app first for a small class of size, so you can understand how the code would change.
- In the process you may discover some inherent limitations of the PIC program. Note these down and share with your teacher and even let the MIT team know.

❏ What are your own amazing ideas?

PICaboo: Part 2

COMPUTATIONAL THINKING CONCEPTS

The following are the computational thinking concepts learnt in Lesson 2.

L3UAI2 PICaboo
1. Events 
2. Conditionals 
3. Data Structures 

PICaboo: Part 2

COMPUTATIONAL THINKING PRACTICES

The following are the computational thinking practices used in this unit.

L3UAI2 PICaboo
1. Being incremental and iterative Iteratively build a complex app over a series of lessons by first training a an image classification model, then uploading the model to the app and finally build the app by extracting and using information from the classifications.
2. Testing and Debugging Test that the Personal Image Classifier (PIC) correctly classifies images in the browser based on its training and then test that the app built on mobile device correctly responds based on the confidence levels of the image classifications.
3. Algorithmic Thinking Complete an algorithm to graphically display confidence levels and make decisions about the machine's response to these confidence levels.