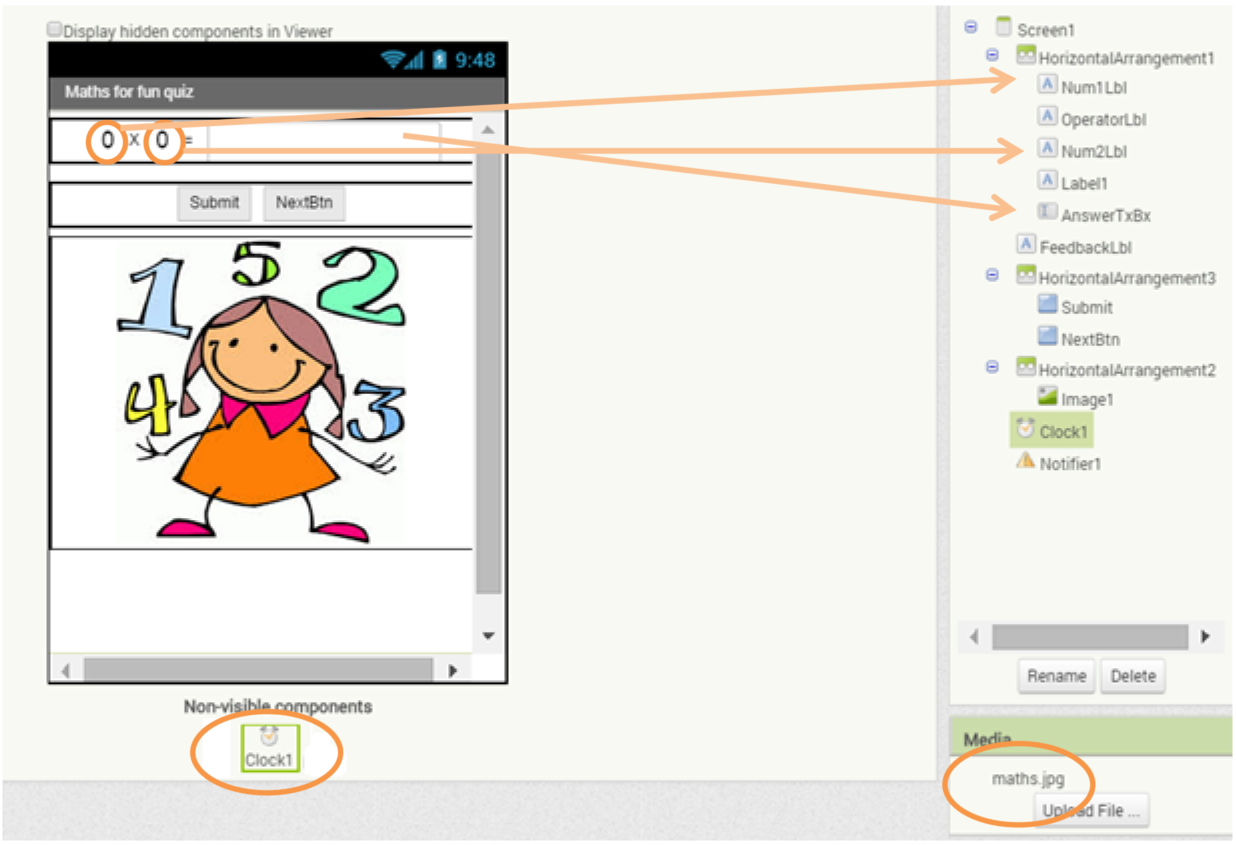
Maths Gameapp

Create a new App inventor project – name it MathsGame

Set up the Components

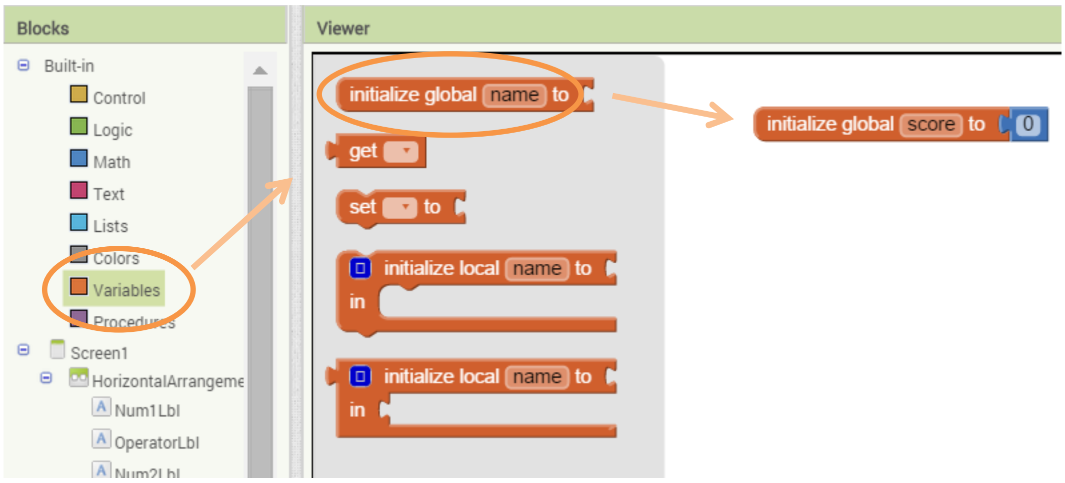
Use the component designer to create the interface for MathGame. When you finish, it should look something like the snapshot below (more detailed instructions below the snapshot).



Note the **Clock** (non-visible components)

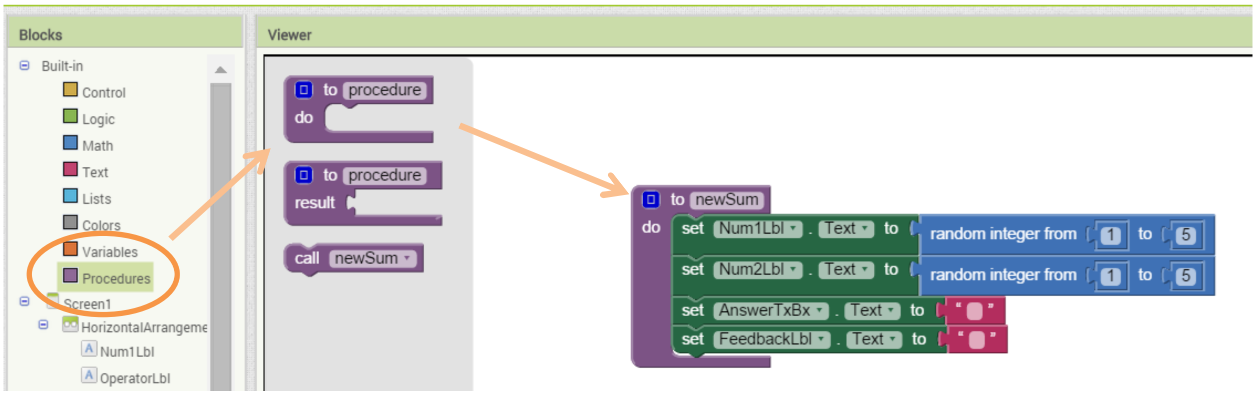
Creating variables

Create a variable to store the game score



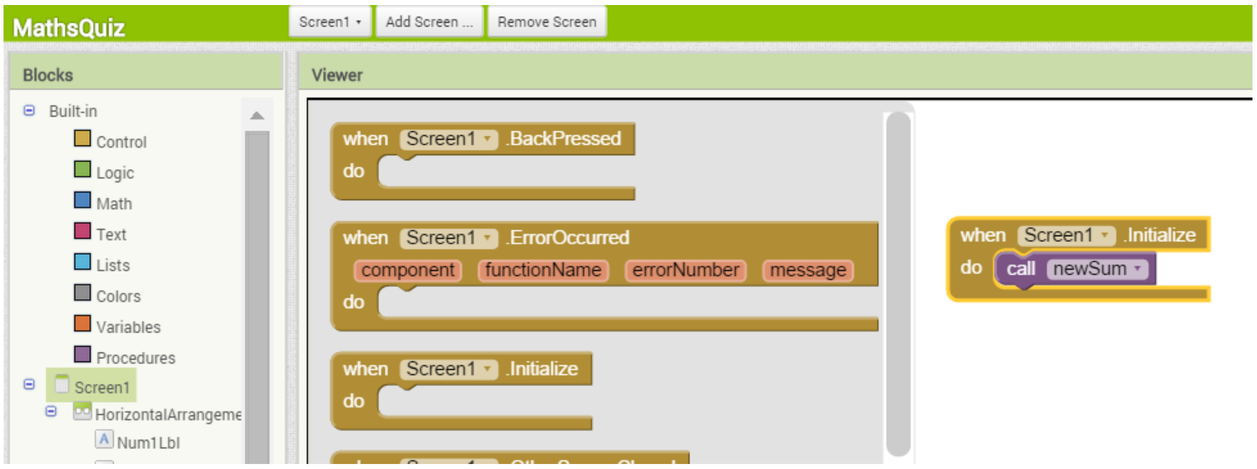
Setting the game up – creating a procedure to show the sum

Each time we need to show a new sum on screen we need to generate a random number for the first and second number. We'll create a procedure to group these lines of code (actions) together.



```
to newSum
do  set Num1Lbl . Text to  random integer from 1 to 5
    set Num2Lbl . Text to  random integer from 1 to 5
    set AnswerTxBx . Text to " "
    set FeedbackLbl . Text to " "
```

Setting the initial screen

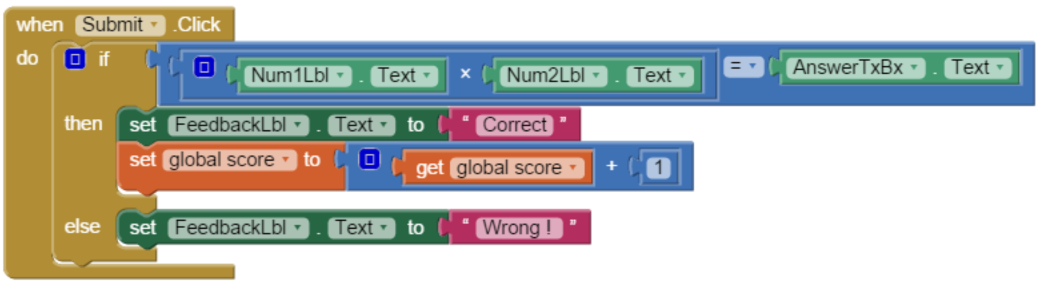


We need to set up the first sum – so we can use our **newSum** procedure. Handy !

Once you put these blocks together, connect your phone and test this feature out!

The submit button

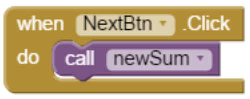
When clicked we need to check the answer in the TextBox (entered by the user) and compare it with the expected answer. To do this we'll use an IF .. ELSE statement. If TRUE we let the user know the answer was correct and also update the score otherwise(FALSE) we let the user know the answer was incorrect.


when Submit .Click
do if Num1Lbl . Text × Num2Lbl . Text = AnswerTxBx . Text
then set FeedbackLbl . Text to " Correct "
set global score to get global score + 1
else set FeedbackLbl . Text to " Wrong ! "

Once you put these blocks together, connect your phone and test this feature out!

The next button

When the user options to move to the next question, we need to update the **sum. (There's that procedure again!)**

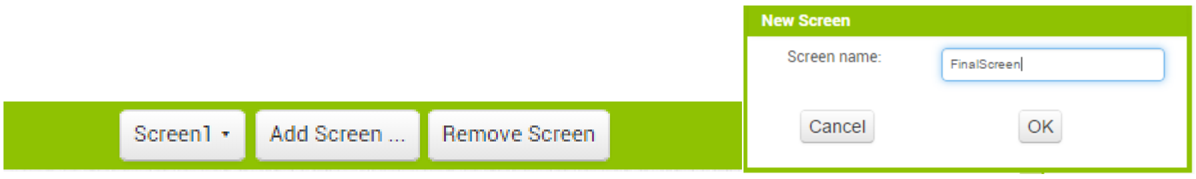


Once you put these blocks together, connect your phone and test this feature out!

Adding a timer

Ok this game could go on for a while! Let's add a timer so that after 20 seconds the game finishes. When complete we'llsave the score, navigate to a new screen and show the final score. The options to quit and save the score or play again will be shown. If play again is selected then we can reset the score and show a new sum and if not we will quit the application. We'll need a **Clock** and a **TinyDb** component for this.

First we will add a new screen and call it "FinalScreen":



Whenever the clock timer runs out we will stop the timer,save the score and navigate to the newly created Final screen.

Saving the score

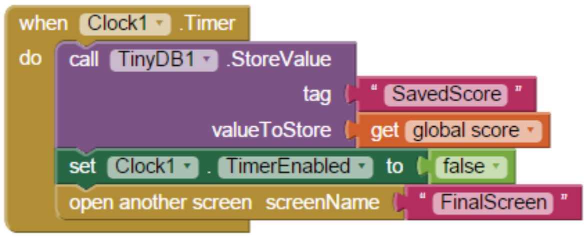
We have used a global variable to keep track of the score in the app. However, if an app sets the value of a variable and the user then quits the app, the value of that variable will not be remembered the next time the app is run.

To save the data and *persist*the data, we need the ability to store values. To do this, we will use a non-visible component called **TinyDB.** This component is found in the Storage tab in the palette and will be used to store the data to be persisted (kept if the app is shut down, closed, suspended etc).

Use your mouse to add a TinyDB component to your application:

The TinyDB works by storing value and referring to the saved data by *Tags.* We will create a tag called "SavedScore" and set the value to store to equal the global score variable:

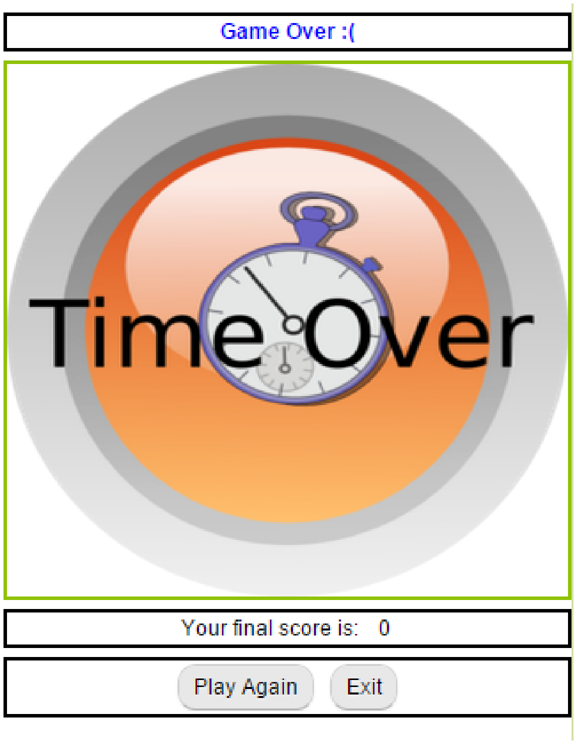
When the timer runs out, we want to save the value that is currently stored in the global variable "Score" to the TinyDB, disable the timer and navigate to the Final Screen.



```
when Clock1 .Timer
do  call TinyDB1 .StoreValue
                        tag  " SavedScore "
               valueToStore  get global score
    set Clock1 . TimerEnabled to false
    open another screen  screenName  " FinalScreen "
```

Designing the Final Screen

Use the component designer again to create the interface for Final Screen. Don't be afraid to be creative but an example screen will be shown:

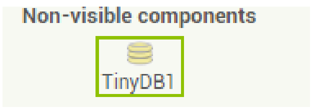


Designing the Final Score using Persisted Data (TinyDB)

The final screen will have a label to display the final score.

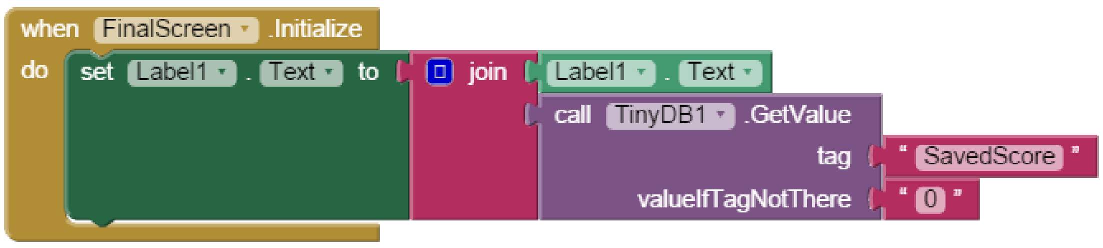
To do this we will use the TinyDB component to get the data stored whenever the timer ran out previously and display the score in a new label.

Drag the TinyDB component the new Screen:



Now add a new label to your Game Screen and give it the text "Your final score is: ":

Whenever the screen is initialised, we want to check in the TinyDB if we have a previous score saved and if we do, display the score in the new label and if not display 0.

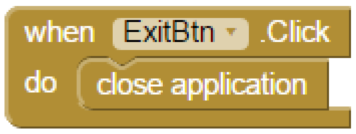


```
when FinalScreen .Initialize
do  set Label1 . Text to  join  Label1 . Text
                                call TinyDB1 .GetValue
                                              tag  " SavedScore "
                               valueIfTagNotThere  " 0 "
```

*Note using ***join*** to join the current text in the label with either 0 or the saved score from the TinyDB.

Exiting the application

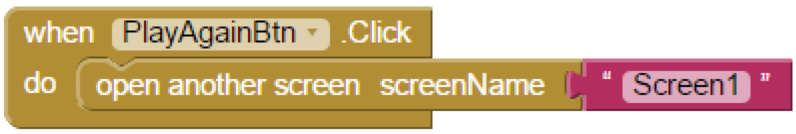
The screen will contain two buttons: Play Again and Exit. Add the following code blocks when the Exit button is clicked to close the application:



```
when ExitBtn .Click
do  close application
```

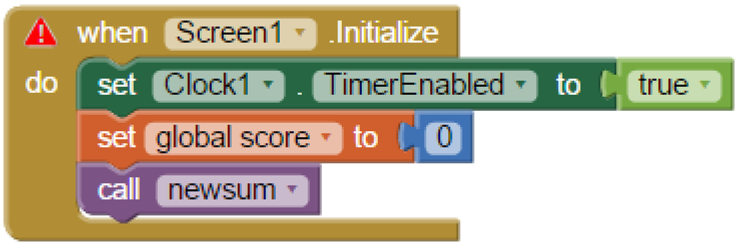
Playing again

We want the ability to allow the user to play again. Add the following code blocks to the Play Again button:



```
when PlayAgainBtn .Click
do  open another screen  screenName  " Screen1 "
```

As we have navigated back to Screen1, we will now have to reset the score to 0, restart the timer and call the newsum procedure when the screen is initialised (shown). Add the following code blocks on Screen1 to allow this:

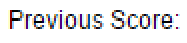


```
when Screen1 .Initialize
do  set Clock1 . TimerEnabled to  true
    set global score to  0
    call newsum
```

Displayed the Previous Score (Using TinyDB)

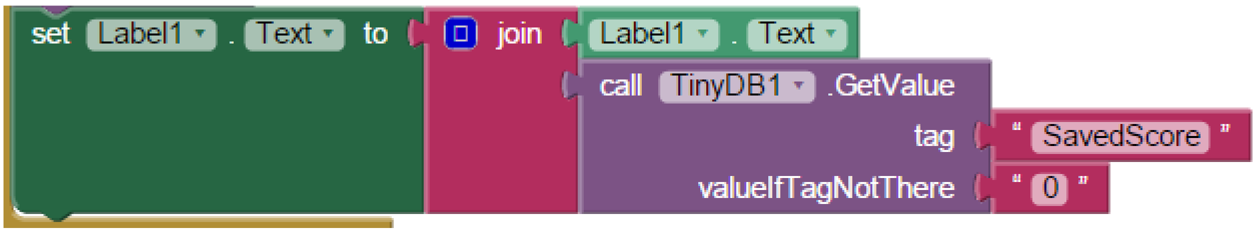
We want to display the score from the previous game. To do this we will use the TinyDB component to get the data stored whenever the timer ran out previously and display the score in a new label.

First add a new label to your Game Screen and name it “PreviousScore: ”:


Previous Score:

Whenever the screen is initialised, we want to check in the TinyDB if we have a previous score saved and if we do, display the score in the new label and if not display 0.

Add the following code to the screen initialize block:



```
set Label1 . Text to  join  Label1 . Text
                            call TinyDB1 .GetValue
                                                tag  " SavedScore "
                                  valueIfTagNotThere  " 0 "
```

We will update the text of the label to equal the value of the label i.e. Previous Score: and join with the value stored in the TinyDBby using the “SavedScore” tag and if not found, add 0 to the label.

Now your application should be finished but don’t forget to add an icon to the app.

Possible improvements

1. Change the time
2. Add a sound when the user answers correctly and another sound if wrong!
3. Display the correct answer if the user get a question wrong
4. Make it more difficult – e.g. time tables 1 – 12.
5. If the user keeps hitting the submit button with a correct score entered in the answer text box the score is increased every time. Can you fix this bug?