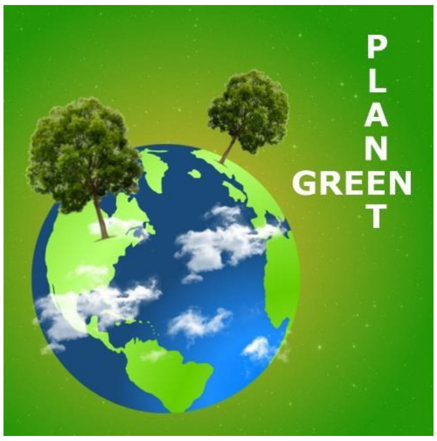
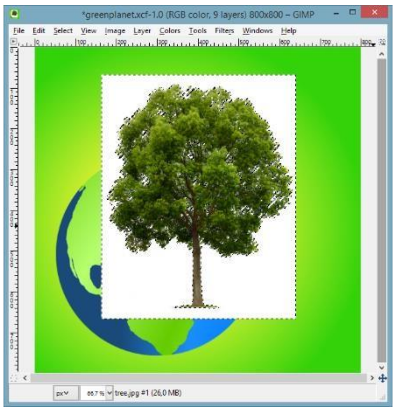
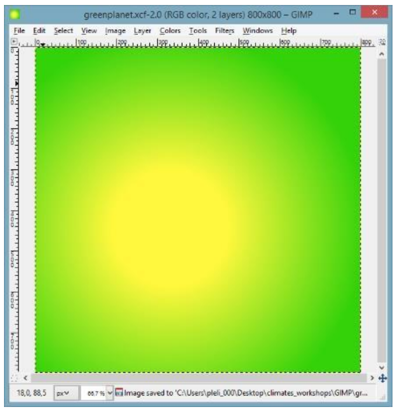
Task 1 | **GIMP | "Green Planet"**

Your task is to create an image looks like this:




*greenplanet.xcf-1.0 (RGB color, 9 layers) 800x800 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
tree.jpg #1 (26,0 MB)


greenplanet.xcf-2.0 (RGB color, 2 layers) 800x800 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Image saved to 'C:\Users\pleli_000\Desktop\climates_workshops\GIMP\gr...

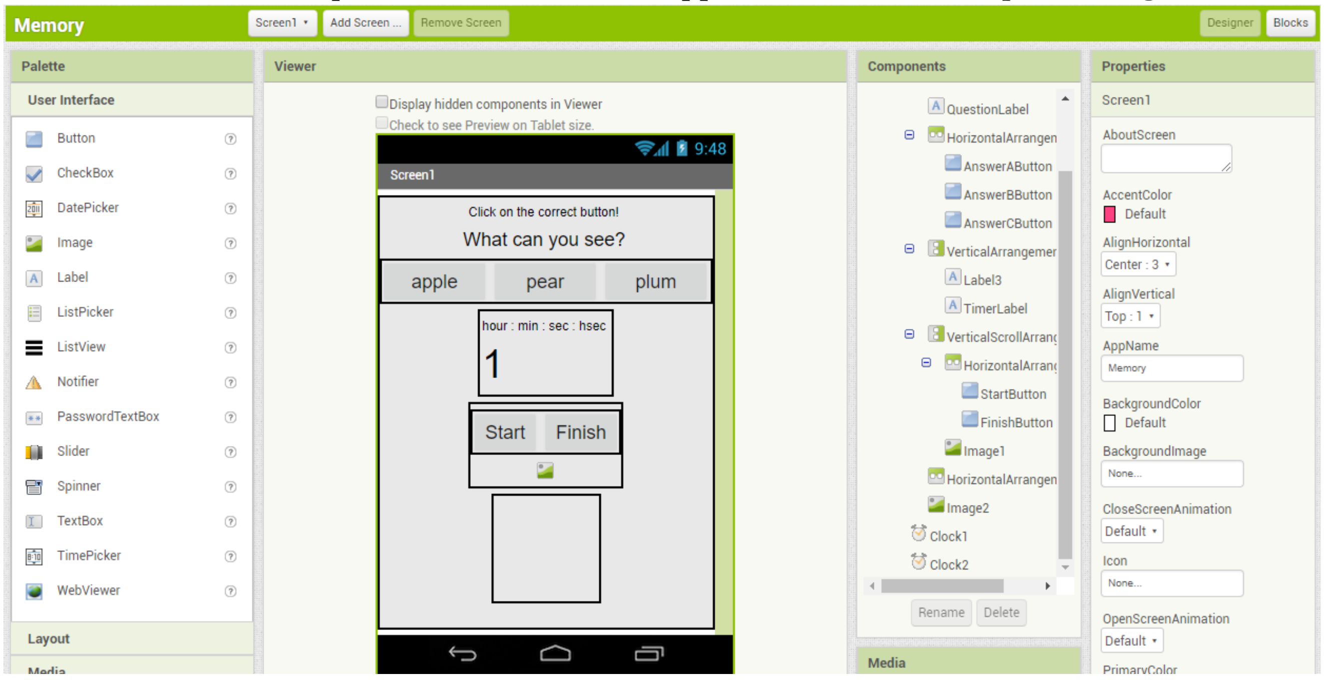
- Create an 800 x 800 pixel image with white background.
- Set the background color to light yellowish-green, radial gradient, set the *Offset* to around *30*, set the type of the gradient to *FB to BG* where the foreground color is lightyellow, and the background color is green.
- Open the *space.jpg* image as a new layer, set it to black and white (*Colors\Colorize Tool\Saturation*), set its saturation to *0* and its opacity to around *30*.
- Open the *earth1.png* or *earth2.png* image as a new layer, use the *Scale Tool* to make the globe smaller, then place the globe using the *Move Tool* in the bottom left corner of image.
- Open the *clouds.png* image as a new layer and move them onto the earth, like in the example.
- Open the *tree.jpg* image as a new layer and select the *Select By Colour Tool*. (It is helpful for us because we would like to select the all the white area in the image.) Set the *Threshold* to around *50*, and then click on the background. This will select all the white area in the image. Press the *Delete* button to remove the white areas and dismiss selection, then move the tree as you like.

PLANT
GREEN

- Create the title of your image in two parts, then set the two text layers to look similar to the sample image.
- Save your project as *GreenPlanet* in *.xcf* and also export the image in *.png* format.

Task 2 | **MIT App Inventor 2 | “Memory”**

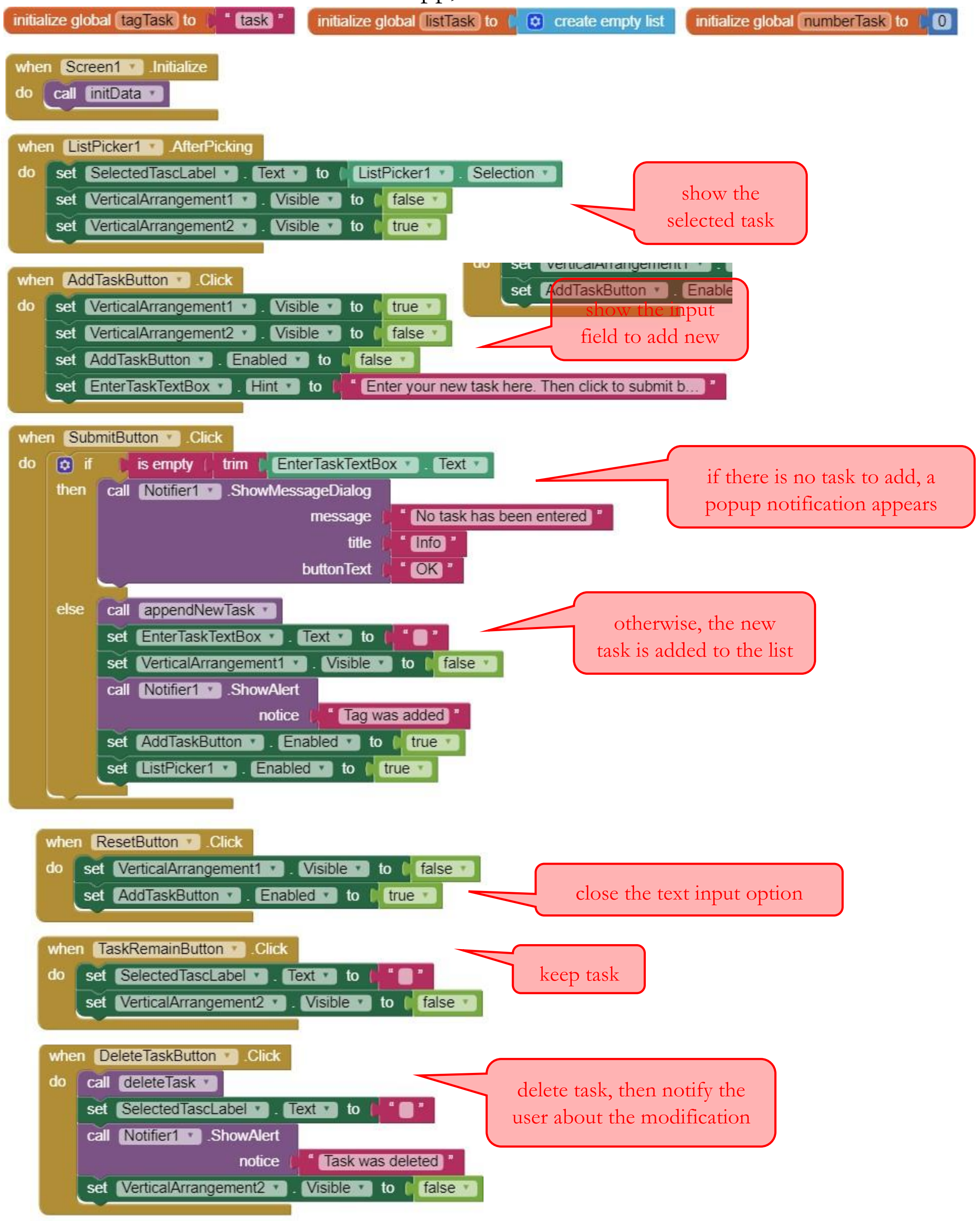
- *Your task is to try to create the app on your own according to a not so detailed description shown below.*
- Here are the components for the *To do list* app, as shown in the *Component Designer:*



Components:
- QuestionLabel
- HorizontalArrangen
  - AnswerAButton
  - AnswerBButton
  - AnswerCButton
- VerticalArrangemer
  - Label3
  - TimerLabel
- VerticalScrollArrang
  - HorizontalArrang
    - StartButton
    - FinishButton
  - Image1
- HorizontalArrangen
- Image2
- Clock1
- Clock2

Properties – Screen1:
- AboutScreen
- AccentColor: Default
- AlignHorizontal: Center : 3
- AlignVertical: Top : 1
- AppName: Memory
- BackgroundColor: Default
- BackgroundImage: None...
- CloseScreenAnimation: Default
- Icon: None...
- OpenScreenAnimation: Default
- PrimaryColor

- Here are the blocks for the *To do list* app, as shown in the *Blocks Viewer:*



```
initialize global tagTask to "task"
initialize global listTask to create empty list
initialize global numberTask to 0

when Screen1 .Initialize
do  call initData

when ListPicker1 .AfterPicking
do  set SelectedTascLabel . Text to ListPicker1 . Selection
    set VerticalArrangement1 . Visible to false
    set VerticalArrangement2 . Visible to true

do  set VerticalArrangement1 .
    set AddTaskButton . Enable

when AddTaskButton .Click
do  set VerticalArrangement1 . Visible to true
    set VerticalArrangement2 . Visible to false
    set AddTaskButton . Enabled to false
    set EnterTaskTextBox . Hint to "Enter your new task here. Then click to submit b..."

when SubmitButton .Click
do  if    is empty  trim  EnterTaskTextBox . Text
    then  call Notifier1 .ShowMessageDialog
                    message  "No task has been entered"
                    title  "Info"
                    buttonText  "OK"
    else  call appendNewTask
          set EnterTaskTextBox . Text to " "
          set VerticalArrangement1 . Visible to false
          call Notifier1 .ShowAlert
                    notice  "Tag was added"
          set AddTaskButton . Enabled to true
          set ListPicker1 . Enabled to true

when ResetButton .Click
do  set VerticalArrangement1 . Visible to false
    set AddTaskButton . Enabled to true

when TaskRemainButton .Click
do  set SelectedTascLabel . Text to " "
    set VerticalArrangement2 . Visible to false

when DeleteTaskButton .Click
do  call deleteTask
    set SelectedTascLabel . Text to " "
    call Notifier1 .ShowAlert
              notice  "Task was deleted"
    set VerticalArrangement2 . Visible to false
```

show the selected task

show the input field to add new

if there is no task to add, a popup notification appears

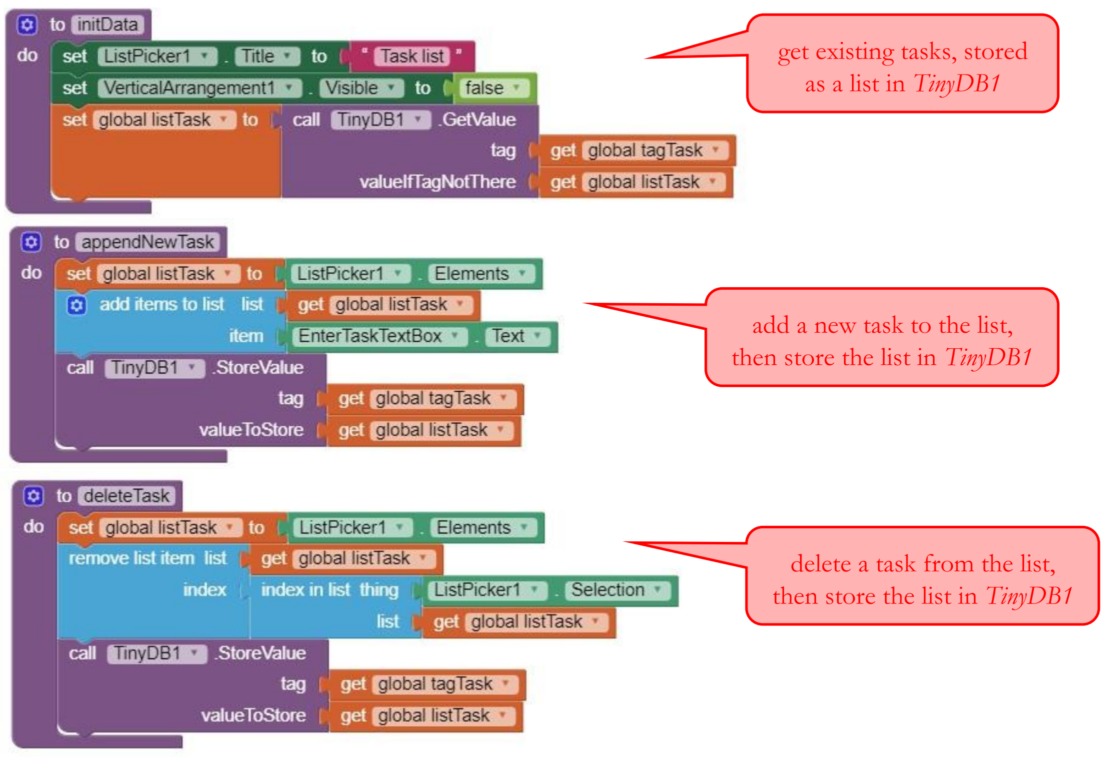
otherwise, the new task is added to the list

close the text input option

keep task

delete task, then notify the user about the modification

  - We use procedures in *App Inventor* to create new blocks that we can use repeatedly and take up less space than all of the blocks used in the original procedure. If we are using the same sets of blocks more than once, these blocks are called redundant.



```
to initData
do  set ListPicker1 . Title to " Task list "
    set VerticalArrangement1 . Visible to false
    set global listTask to  call TinyDB1 .GetValue
                                         tag  get global tagTask
                            valueIfTagNotThere  get global listTask

to appendNewTask
do  set global listTask to  ListPicker1 . Elements
    add items to list  list  get global listTask
                       item  EnterTaskTextBox . Text
    call TinyDB1 .StoreValue
                   tag  get global tagTask
          valueToStore  get global listTask

to deleteTask
do  set global listTask to  ListPicker1 . Elements
    remove list item  list  get global listTask
                     index  index in list  thing  ListPicker1 . Selection
                                            list  get global listTask
    call TinyDB1 .StoreValue
                   tag  get global tagTask
          valueToStore  get global listTask
```

get existing tasks, stored as a list in *TinyDB1*

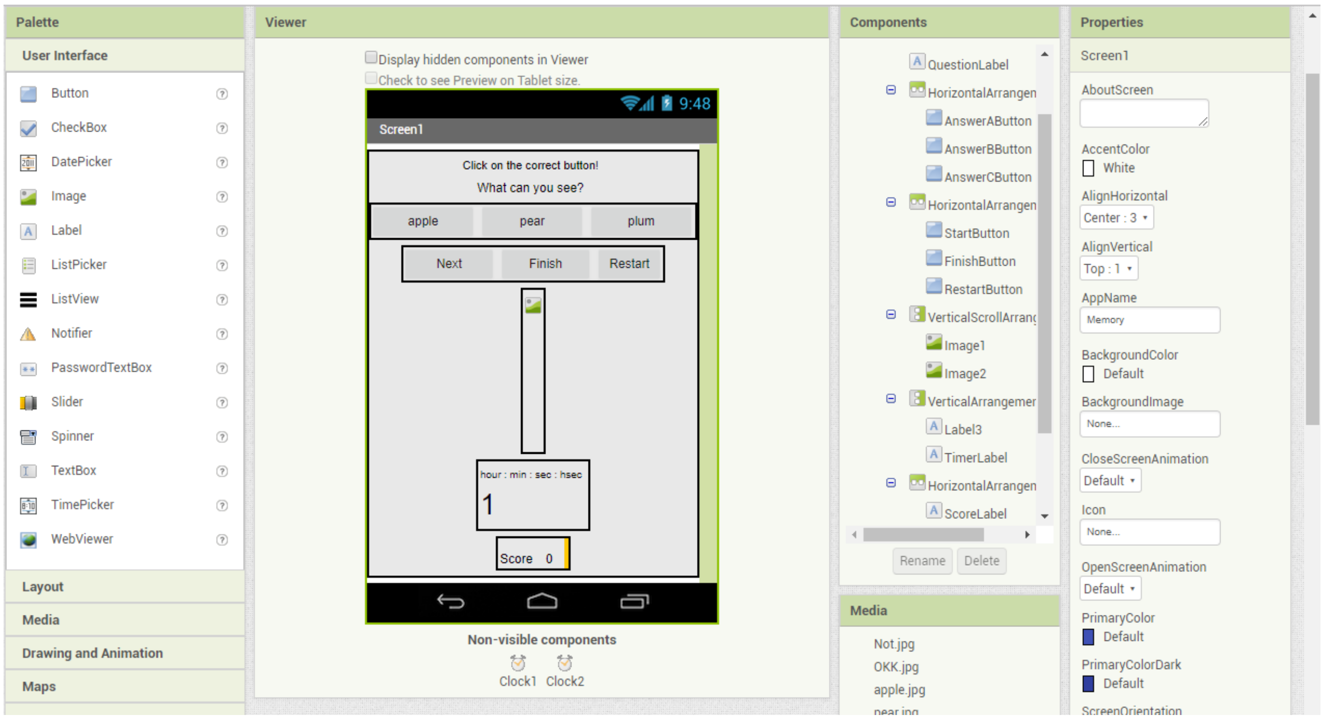
add a new task to the list, then store the list in *TinyDB1*

delete a task from the list, then store the list in *TinyDB1*

- Find out more about *Procedures* by clicking on the links below.
  - http://appinventor.mit.edu/explore/ai2/support/blocks/procedures.html
  - http://www.appinventor.org/Procedures2

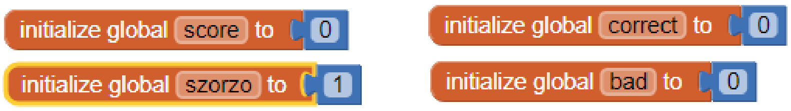
Task 3 | **MIT App Inventor 2 | "Memory2"**

- *The task is to supplement the previous app with a section that counts the points for correct answers, or may be deducted for incorrect answers. Time can be taken into account. If you have given a good answer quickly, it is a plus point if you make a bad answer quickly, then deduction.*
- Here are the components for the Memory2 app, as shown in the *Component Designer:*



Screen1
AboutScreen
AccentColor: White
AlignHorizontal: Center : 3
AlignVertical: Top : 1
AppName: Memory
BackgroundColor: Default
BackgroundImage: None...
CloseScreenAnimation: Default
Icon: None...
OpenScreenAnimation: Default
PrimaryColor: Default
PrimaryColorDark: Default
ScreenOrientation

- You can use new variables.

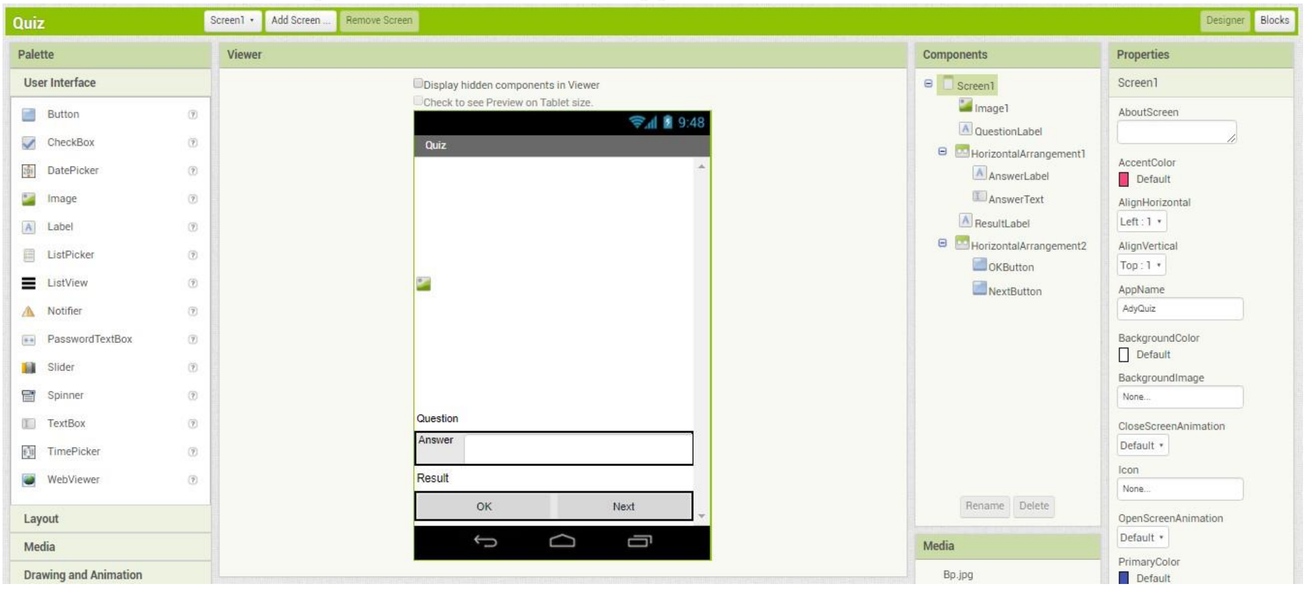


```
initialize global score to 0
initialize global correct to 0
initialize global szorzo to 1
initialize global bad to 0
```

- Be careful, not allow to click two times for the correct button!

Task 4 | **MIT App Inventor 2 | "Quiz 1"**

- Here are the components for the *Quiz 1* app, as shown in the *Component Designer:*

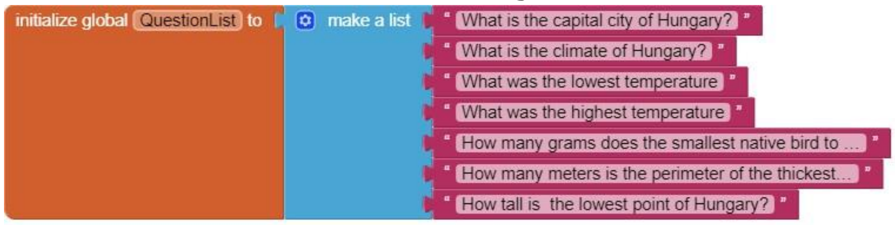


```
Screen1
  Image1
  QuestionLabel
  HorizontalArrangement1
    AnswerLabel
    AnswerText
  ResultLabel
  HorizontalArrangement2
    OKButton
    NextButton
```

Properties (Screen1): AboutScreen; AccentColor: Default; AlignHorizontal: Left : 1; AlignVertical: Top : 1; AppName: AdyQuiz; BackgroundColor: Default; BackgroundImage: None...; CloseScreenAnimation: Default; Icon: None...; OpenScreenAnimation: Default; PrimaryColor: Default

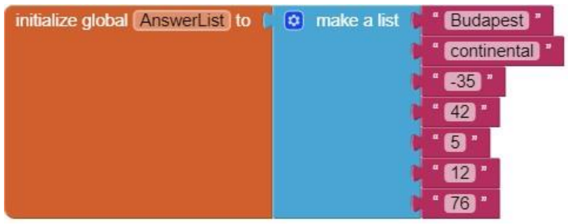
Media: Bp.jpg

- Create 3 variables consisting of 3 lists.
  - In the first one, have to be stored the *Questions.*



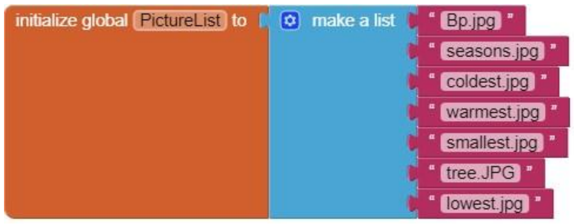
```
initialize global QuestionList to make a list
    "What is the capital city of Hungary?"
    "What is the climate of Hungary?"
    "What was the lowest temperature"
    "What was the highest temperature"
    "How many grams does the smallest native bird to ..."
    "How many meters is the perimeter of the thickest..."
    "How tall is  the lowest point of Hungary?"
```

  - In the second one, have to be stored the *Correct answers.*



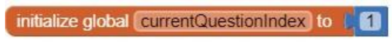
```
initialize global AnswerList to make a list
    "Budapest"
    "continental"
    "-35"
    "42"
    "5"
    "12"
    "76"
```

  - In the third one, have to be stored the *Images* for the questions.



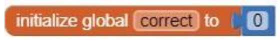
```
initialize global PictureList to make a list
    "Bp.jpg"
    "seasons.jpg"
    "coldest.jpg"
    "warmest.jpg"
    "smallest.jpg"
    "tree.JPG"
    "lowest.jpg"
```

- Create 2 more variables.
  - One of them has to store the number of the current question.



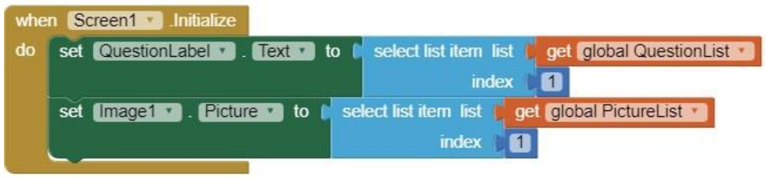
```
initialize global currentQuestionIndex to 1
```

  - The other one has to store the number of the correct answers.



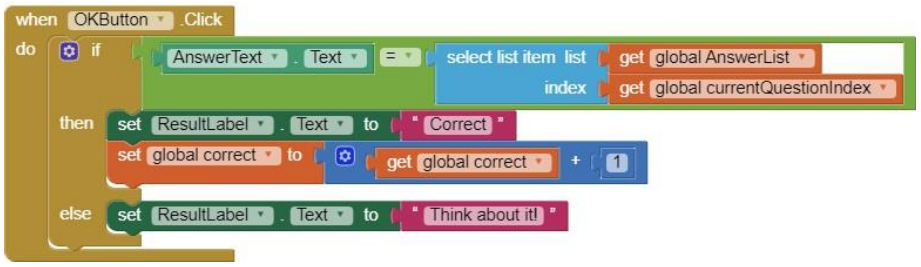
```
initialize global correct to 0
```

- At the start of the application the *QuestionLabel* has to display the first element of the *QuestionList* and also the belonging image.



```
when Screen1 .Initialize
do  set QuestionLabel . Text to  select list item  list  get global QuestionList
                                                   index  1
    set Image1 . Picture to  select list item  list  get global PictureList
                                             index  1
```

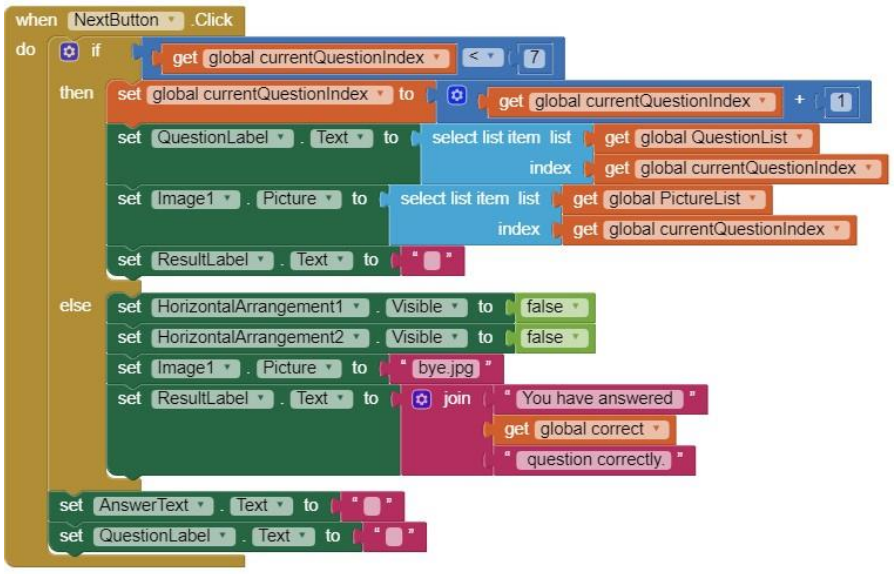
- When the *OKButton* is clicked, it has to analyze if the entered text is in accordance with the current element of the *AnswerList.*
    - If yes, the *ResultLabel* has to show the correct answer and increase the number of the correct answers by adding plus one point.
    - Else, it has to show the *"Think about it!"* text.



```
when OKButton .Click
do  if   AnswerText . Text  =  select list item  list   get global AnswerList
                                                 index  get global currentQuestionIndex
    then set ResultLabel . Text to  " Correct "
         set global correct to  get global correct  +  1
    else set ResultLabel . Text to  " Think about it! "
```

- When the *NextButton* is clicked, it has to analyze if the number of the question is smaller than the index of the last element of the *QuestionList.*
    - If yes, it has to increase the number of the questions by adding one more, so the *QuestionLabel* has to show the next element from the *QuestionList* and also the belonging picture.
    - Else, it has to turn off the *AnswerTextBox* by setting the *HorizontalArrangement1* and *HorizontalArrangement2* (which contain the buttons) to invisible.

      ✦ It has to show the *"BYE"* image and the number of the correct answers. ✦ And it has to set the *AnswerText* and the *QuestionLabel* to empty.



```
when NextButton .Click
do  if   get global currentQuestionIndex  <  7
    then set global currentQuestionIndex to  get global currentQuestionIndex  +  1
         set QuestionLabel . Text to  select list item  list   get global QuestionList
                                                     index  get global currentQuestionIndex
         set Image1 . Picture to  select list item  list   get global PictureList
                                                 index  get global currentQuestionIndex
         set ResultLabel . Text to  " "
    else set HorizontalArrangement1 . Visible to  false
         set HorizontalArrangement2 . Visible to  false
         set Image1 . Picture to  " bye.jpg "
         set ResultLabel . Text to  join  " You have answered "
                                          get global correct
                                          " question correctly. "
    set AnswerText . Text to  " "
    set QuestionLabel . Text to  " "
```