Code Challenge 3

Search a Database

Description

Create an app that allows the user to search a database of information and view their results. The database should contain names and descriptions of famous people, places, movies, music, events or anything else you want. Make sure you have at least three entries in your database.

See our example app here: Technovation Challenge #3

Try to figure out how to complete the Coding Challenge on your own before looking at the instructions. The instructions show just one way of how this challenge can be solved. Happy coding!

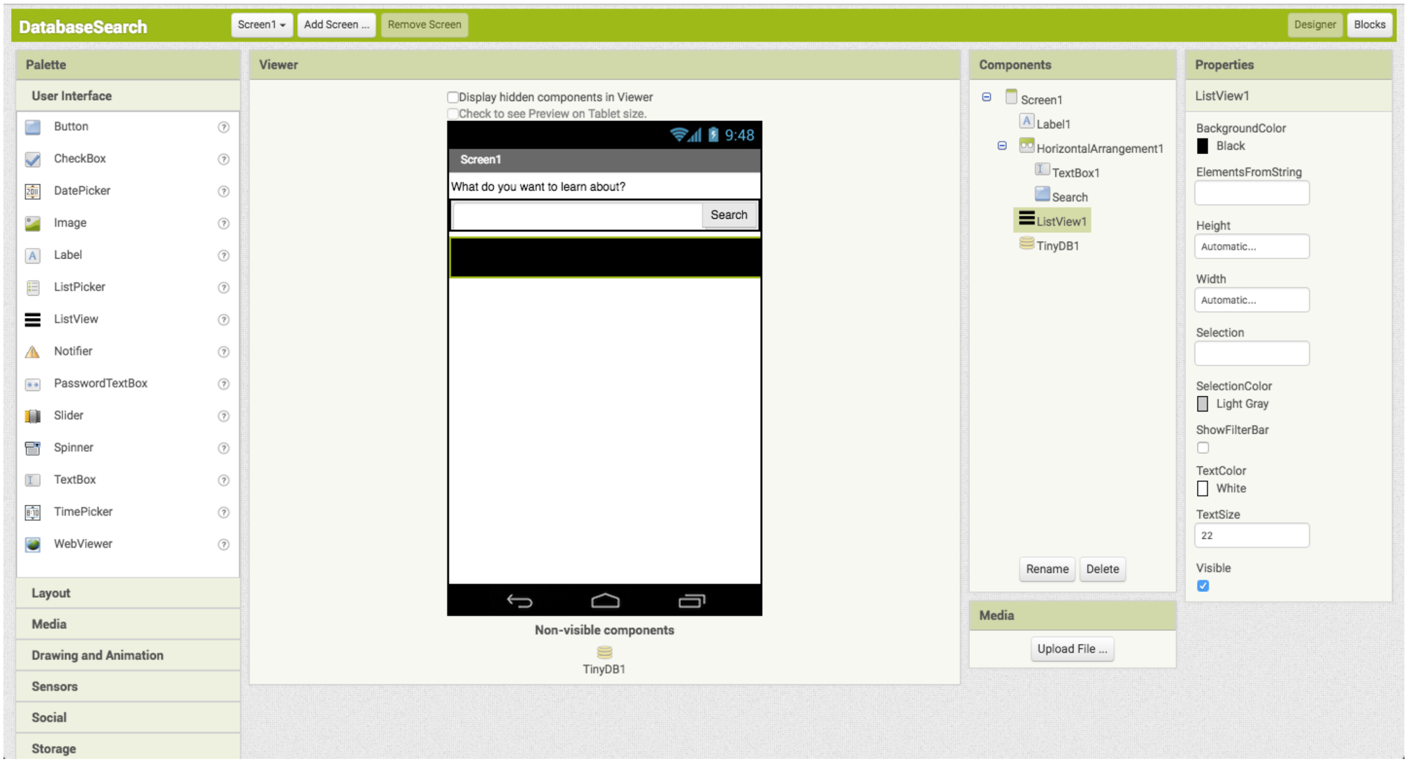
We decided to tackle this coding challenge by storing our data in a TinyDB. We stored the description of the scientists with tag names of her name. After that we made two lists, one with the person's name, and one with the descriptions of her work. The user will enter text and the app will search each item in both lists for a match. If there is a match, the woman's name will appear in a **listview**. The user can then select which scientist they want to learn more about and a new screen will open with the description of her work. For this app, we had to use a **TinyDB** to store the data so that we could use it on separate screens.

Name (Tag)	Description
Ada Lovelace	Ada was an English mathematician and writer, who lived in the 1800s and is known for her work on Charles Babbage's early mechanical general-purpose computer, the Analytical Engine. Ada is often credited as being the first programmer ever because she wrote the first algorithm intended to be carried out by a machine.
Marie Curie	Marie was a physicist and chemist who was born in Warsaw, Poland in 1867. With her husband Pierre, she discovered 2 new radioactive elements. Marie won the nobel prize in physics in 1903. She also discovered that radium gas could be used for cancer treatment.
Alice Ball	Alice was a chemist born in Seattle in 1892. She was the first African American and the first woman to graduate from the University of Hawaii. At only 23 years old, Alice developed a cure for leprosy, which prior to her work seemed like a hopeless disease.
Sau Lan Wu	Sau Lan is a particle physicist who was born in the early 1940s during the Japanese occupation of Hong Kong. She earned her PhD at Harvard. She led the team that

	discovered gluon. She is one of the most important particle physicist in her field and has made many groundbreaking discoveries.
Patricia Bath	Patricia is an ophthalmologist and inventor who was born in 1942 in Harlem, New York City. She finished high school in only 2 and a half years and knew she wanted to be a doctor. In 1985 she finished an invention that removes cataracts and restore people's sight around the world.

Designing the Screen

We want a user to be able to search a database, so we are going to add a textbox for the user to enter their search into a button for them to click once to search. We know we are going to store our data in a database, so we dragged a **TinyDB** onto the screen. We also know that we'll be using lists so we added a **listview** to the screen.



ListView1

BackgroundColor: Black

ElementsFromString:

Height: Automatic...

Width: Automatic...

Selection:

SelectionColor: Light Gray

ShowFilterBar: □

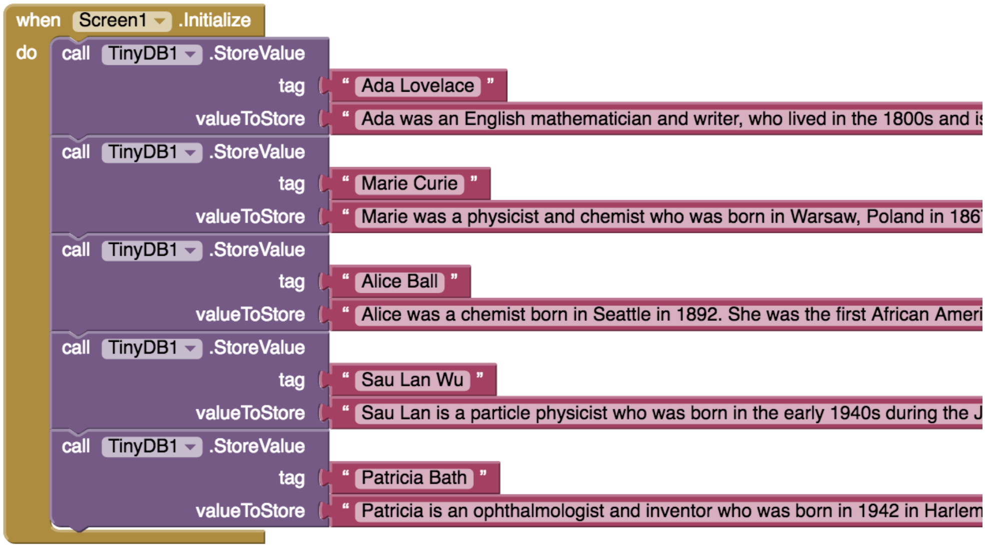
TextColor: White

TextSize: 22

Visible: ☑

Adding Data to the Database

We decided to add the scientist's name as the tag and the description of her work as the value to the database. We copied and pasted the information we wanted to add to the database like this:

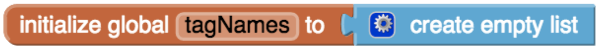


```
when Screen1 .Initialize
do  call TinyDB1 .StoreValue
                   tag  " Ada Lovelace "
          valueToStore  " Ada was an English mathematician and writer, who lived in the 1800s and is
    call TinyDB1 .StoreValue
                   tag  " Marie Curie "
          valueToStore  " Marie was a physicist and chemist who was born in Warsaw, Poland in 186
    call TinyDB1 .StoreValue
                   tag  " Alice Ball "
          valueToStore  " Alice was a chemist born in Seattle in 1892. She was the first African Ameri
    call TinyDB1 .StoreValue
                   tag  " Sau Lan Wu "
          valueToStore  " Sau Lan is a particle physicist who was born in the early 1940s during the J
    call TinyDB1 .StoreValue
                   tag  " Patricia Bath "
          valueToStore  " Patricia is an ophthalmologist and inventor who was born in 1942 in Harlem
```

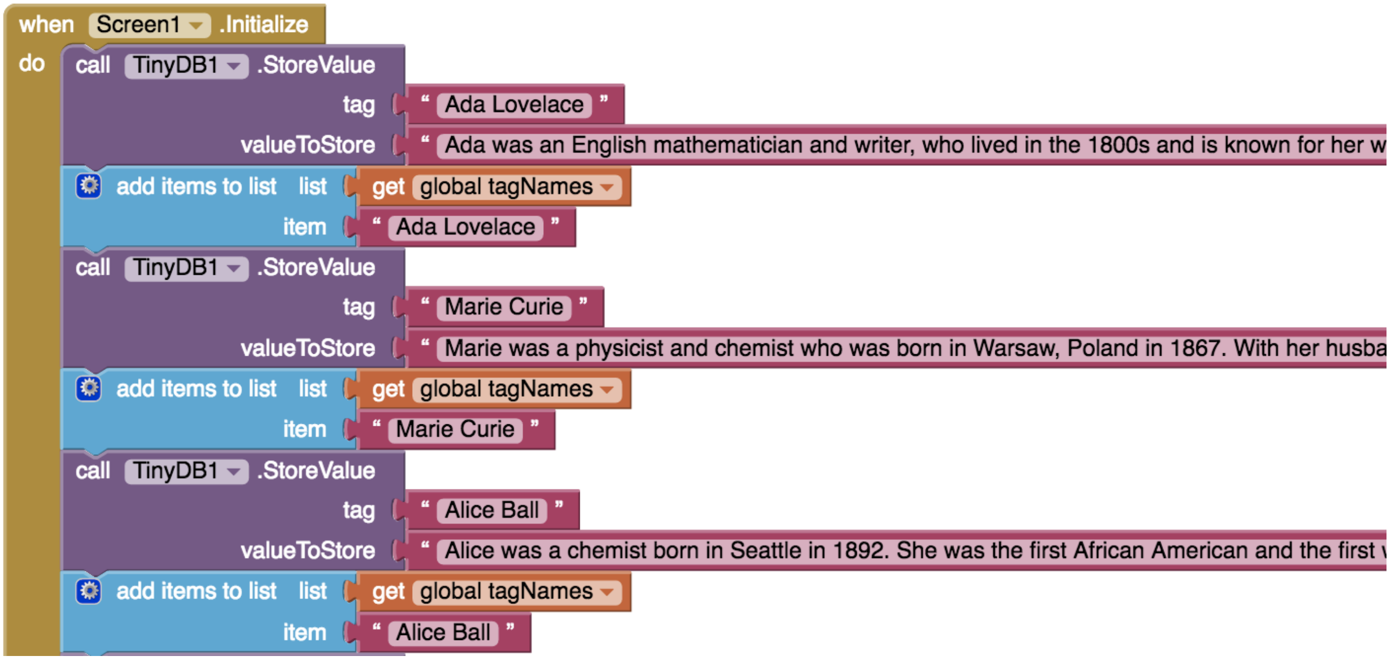
You should store your information into the database when your screen initializes so it is the first thing your app does. You'll need to use a database to store this information if you are planning on using it in any other screens.

Creating lists of tag names and database values

We created a list to store the tag names in. First we created an empty list called "TagNames". Next we added blocks to add every one of our tag names to this list. We made sure that the tag names being added to the list matched exactly the tag names in the database, including all capital letters.

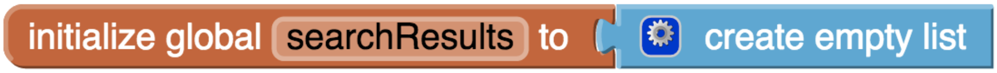


```
initialize global tagNames to  create empty list
```

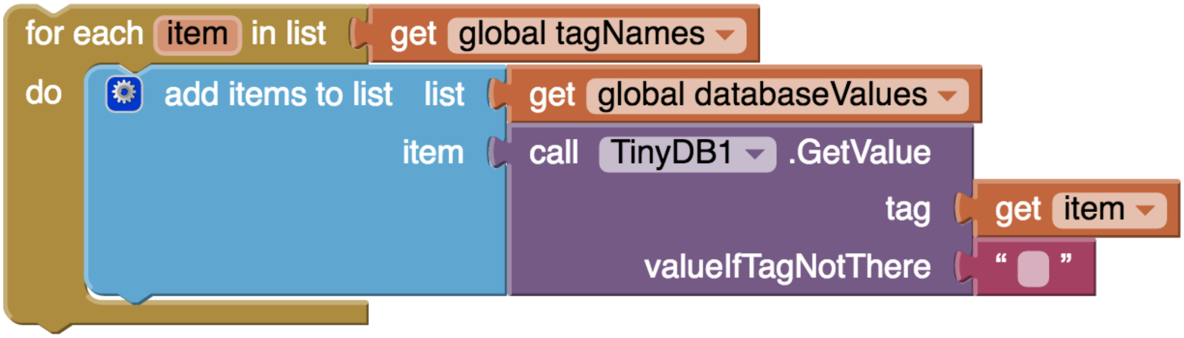


```
when Screen1 .Initialize
do  call TinyDB1 .StoreValue
                   tag  " Ada Lovelace "
          valueToStore  " Ada was an English mathematician and writer, who lived in the 1800s and is known for her w
    add items to list  list  get global tagNames
                       item  " Ada Lovelace "
    call TinyDB1 .StoreValue
                   tag  " Marie Curie "
          valueToStore  " Marie was a physicist and chemist who was born in Warsaw, Poland in 1867. With her husba
    add items to list  list  get global tagNames
                       item  " Marie Curie "
    call TinyDB1 .StoreValue
                   tag  " Alice Ball "
          valueToStore  " Alice was a chemist born in Seattle in 1892. She was the first African American and the first w
    add items to list  list  get global tagNames
                       item  " Alice Ball "
```

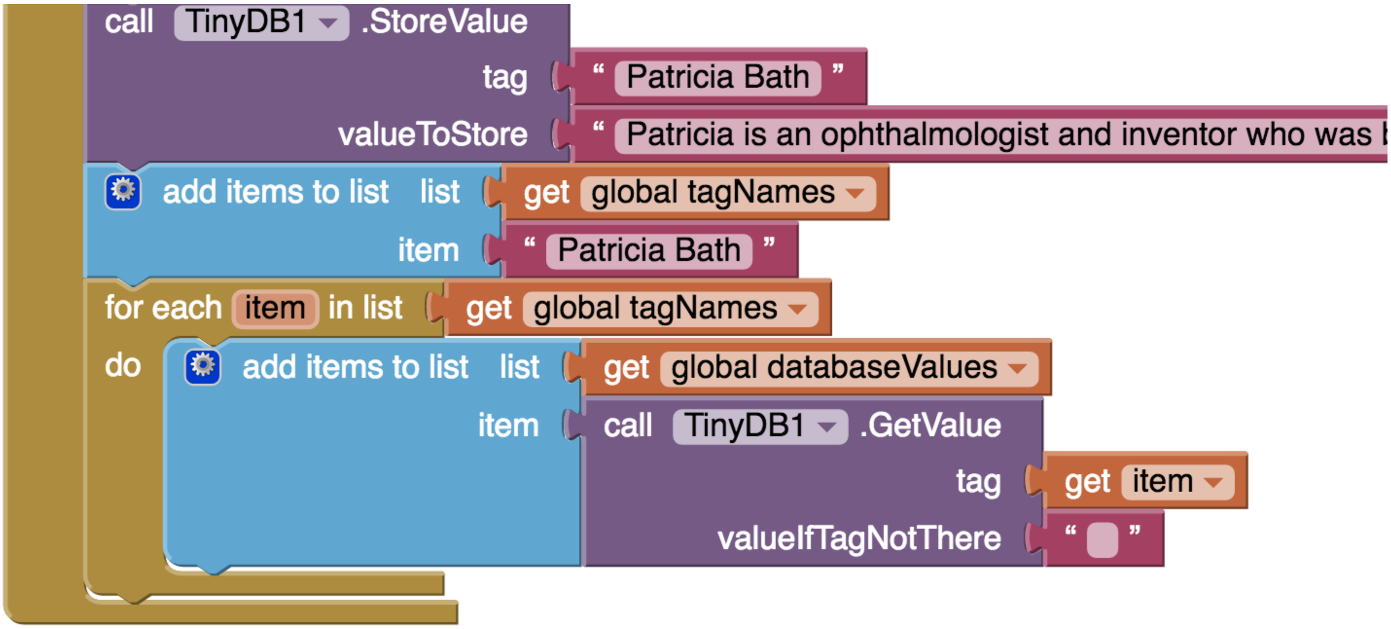
Now that we have a full list with all of our tag names, we need to create another list containing all of the values in the database in the same order as the tag names. First, we created an empty list to hold all the values from the database called "databaseValues". Next, we added each value to the list in the same order as the tag that corresponds to it by using a for loop. For each item in the "tagNames" list, we added the value from the database to the "databaseValues" list. We then added this for loop to the **when Screen1.Initialize** event handler.



```
initialize global searchResults to create empty list
```



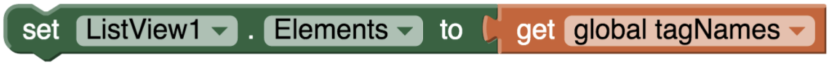
```
for each item in list get global tagNames
do  add items to list  list get global databaseValues
                       item call TinyDB1.GetValue
                                 tag get item
                                 valueIfTagNotThere " "
```



```
call TinyDB1.StoreValue
     tag " Patricia Bath "
     valueToStore " Patricia is an ophthalmologist and inventor who was b
add items to list  list get global tagNames
                   item " Patricia Bath "
for each item in list get global tagNames
do  add items to list  list get global databaseValues
                       item call TinyDB1.GetValue
                                 tag get item
                                 valueIfTagNotThere " "
```

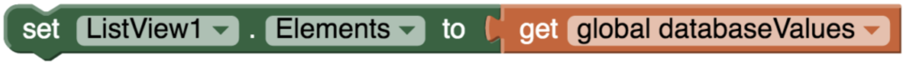
Check point #1

At this point you should have all your data in your database as well as one list with your tag names and one list with your database values. Make sure that each of your lists contains the values you want! Since your app doesn't do anything yet, you can check your work by displaying your lists in your ListView and making sure everything is there. Add this block to your code underneath your for loop in your **when Screen1.Initialize** event handler to check that your “tagNames” list is accurate.



```
set ListView1 . Elements to get global tagNames
```

Add this block to your code underneath your for loop in your **when Screen1.Initialize** event handler to check that your “databaseVales” list is accurate.



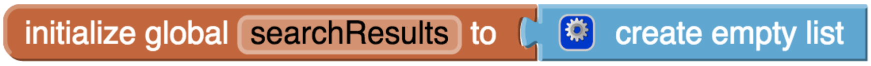
```
set ListView1 . Elements to get global databaseValues
```

After you’ve made sure your lists are accurate, makes sure to delete these blocks!

Programming the search button

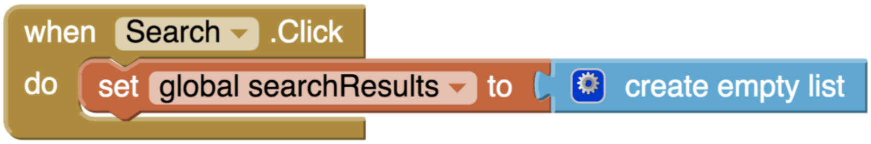
We want our user to be able to search our database by entering text in the textbox and pressing the “search” button. To do this, we are going to put an event handler on the search button that will look through each item in both the “tagName” list and the “databaseValues” list for strings that match what the user entered in the textbox. If there is a match, we’ll add the scientist’s name to the list view. Here’s how we did it.

First,we created a variable called “searchResults” and set it to be an empty list:



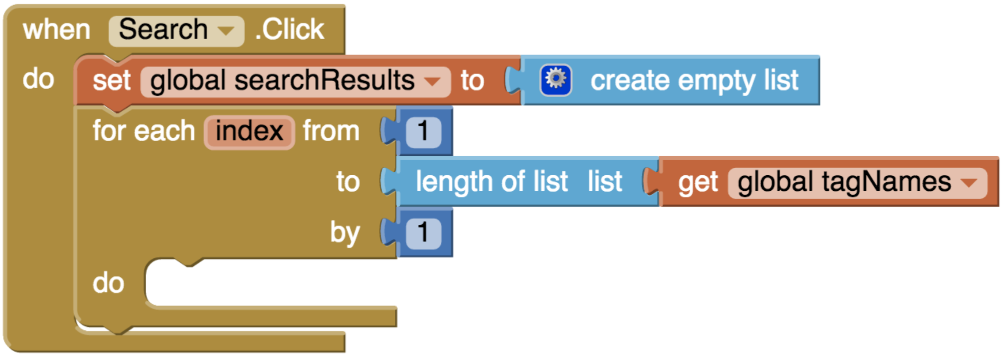
```
initialize global searchResults to create empty list
```

Next we added an event handler to the search button. Inside of it, we added a block to set the search results back to being an empty list. This is because each time the user hits “search” we want to clear out any items that may have been added to “searchResults” in the last search.



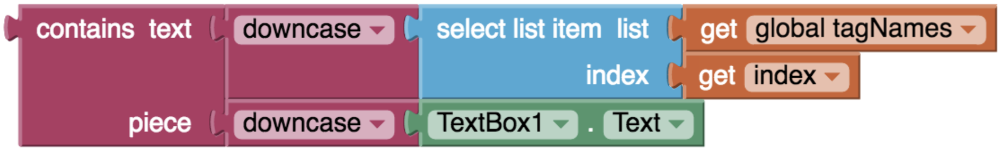
```
when Search .Click
do  set global searchResults to create empty list
```

Next, we set up a **for loop** that searches each item in the “tagName” list and in the “databaseValue” list. We didn’t use the for *each item* in list loop because we wanted to use a counting variable in the **for loop**. We started the **for loop** at "1" and ended it at the number of items in the list. We changed the name of the variable to index.



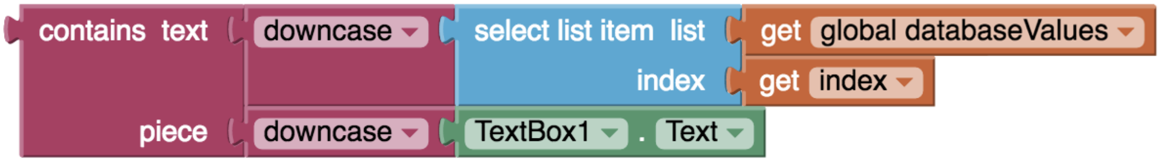
```
when Search .Click
do  set global searchResults to  create empty list
    for each index from 1
                    to  length of list  list  get global tagNames
                    by  1
        do
```

Here's the block we used to compare the text that the user searched for in the textbox to list items in "tagNames". The variable "index" increases by 1 each time the loop repeats so it will eventually compare the textbox text to every item in the list "tagNames". We choose to make everything downcase to allows the search to not be case sensitive. The "contains text" block returns true if the *piece* appears in the *text*.



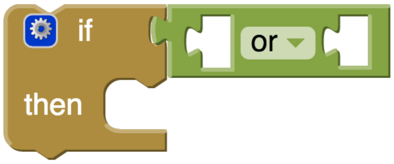
```
contains text  downcase  select list item  list  get global tagNames
                                           index get index
         piece downcase  TextBox1 . Text
```

To compare the user's search to "databaseValues", we used the same blocks except for changing the list name.



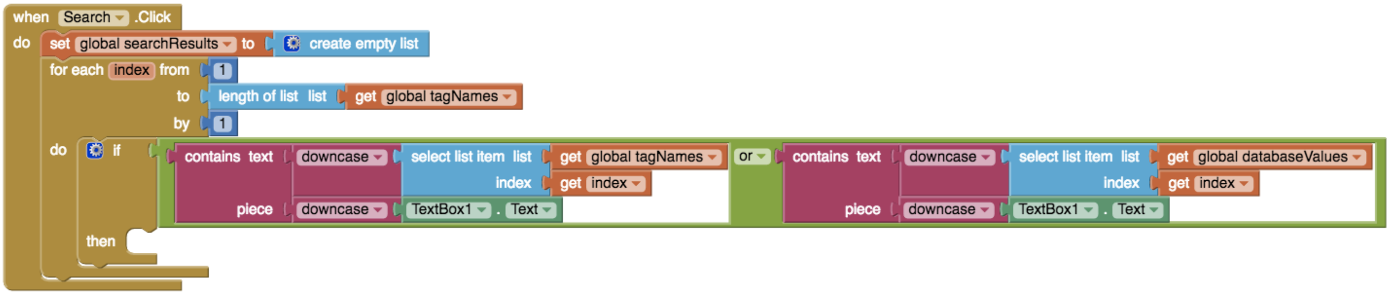
```
contains text  downcase  select list item  list  get global databaseValues
                                           index get index
         piece downcase  TextBox1 . Text
```

We want to show the user the women scientist whose name or description matches the user's search. To do this used these last two blocks with a conditional and a logic **or** operation.



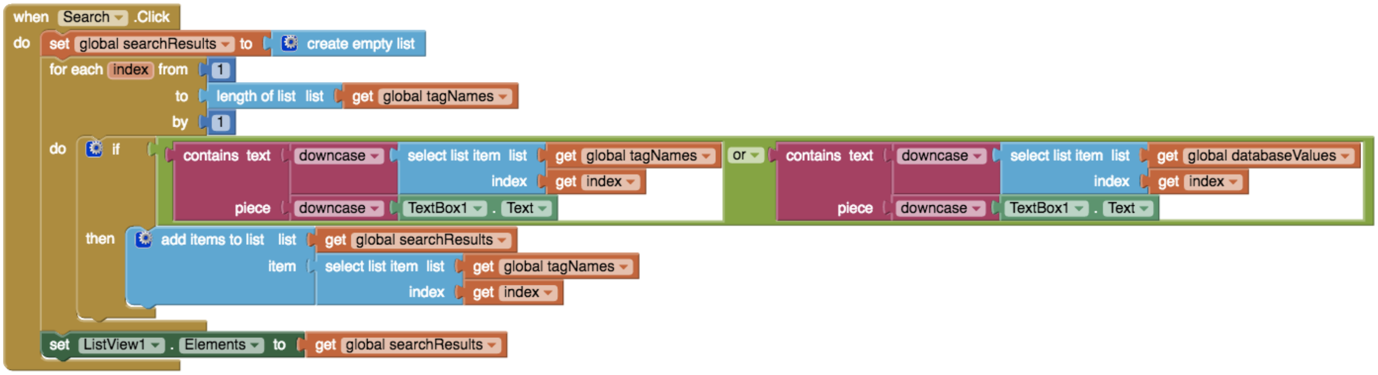
```
if    [ ] or [ ]
then
```

Here is what or code looks like with the the **or** operation filled:



```
when Search .Click
do  set global searchResults to  create empty list
    for each index from 1
                    to  length of list  list  get global tagNames
                    by  1
        do  if   contains text  downcase  select list item  list  get global tagNames    or   contains text  downcase  select list item  list  get global databaseValues
                                                            index get index                                                               index get index
                          piece downcase  TextBox1 . Text                                           piece downcase  TextBox1 . Text
            then
```

We want our code to add the name of the women scientist to the list “searchResults” if either her name or her description matches the user’s search terms. To do this, when there is a match, we added the name from tagNames using the loop variable “index”. We didn’t add an **else** to our conditional statement because if the name or description isn’t match we want our code to do nothing. We then added the items in the list “searchResults” to the ListView. Because this block is outside of the for loop, our code only updates the ListView after for loop has ended.



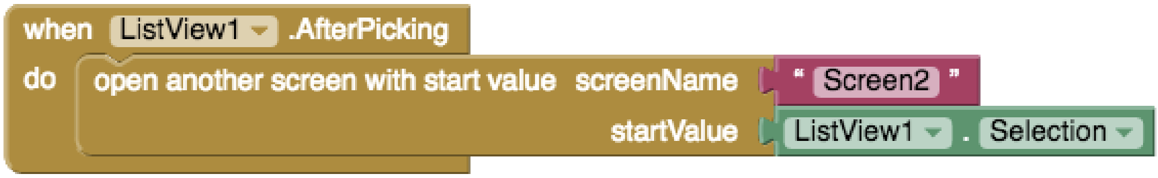
```
when Search .Click
do  set global searchResults to  create empty list
    for each index from 1
                   to  length of list list  get global tagNames
                   by 1
    do  if  contains text  downcase  select list item list  get global tagNames
                                                     index  get index
                       piece  downcase  TextBox1 . Text
            or
            contains text  downcase  select list item list  get global databaseValues
                                                     index  get index
                   piece  downcase  TextBox1 . Text
        then  add items to list list  get global searchResults
                                item  select list item list  get global tagNames
                                                     index  get index
    set ListView1 . Elements to  get global searchResults
```

Check Point #2

You just wrote a lot of code! Check your code to make sure you can search for something. Try searching for a name, like “Alice” or an occupation like “chemist”.

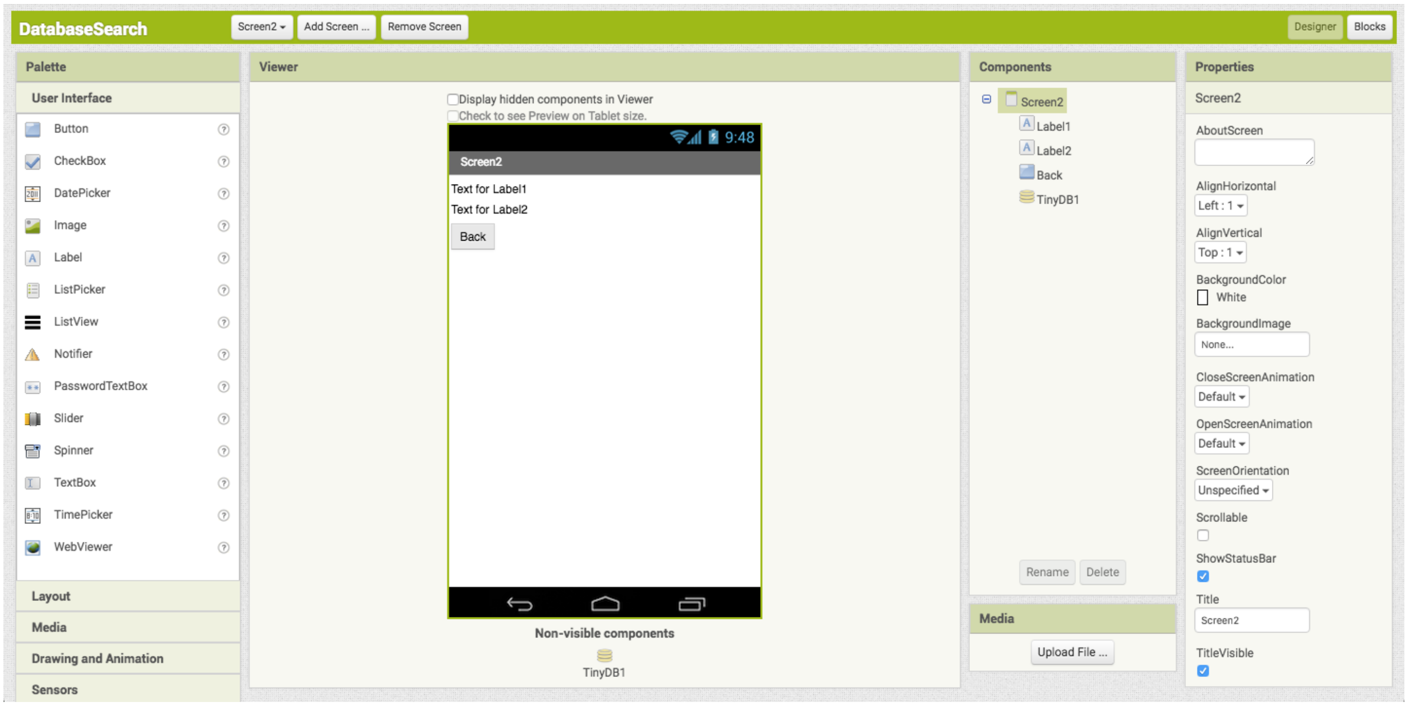
Displaying the search results on a new screen

We decided our app would be better if the user could select the women scientist that showed up in the search results to learn more about her. To do this, when the user picked something from the list, we opened a new screen with a **startValue** of the user’s selection. Remember, the only two ways to pass data between screens is by using a database or a startValue block!



```
when ListView1 .AfterPicking
do  open another screen with start value  screenName  " Screen2 "
                                          startValue  ListView1 . Selection
```

We designed our second screen with two labels. One to display the scientist's name and one to display her description. We also added a back button to return to **Screen1**. Don't forget that you also need to add a **TinyDB** to this screen!



□Display hidden components in Viewer
□Check to see Preview on Tablet size.

Screen2 (9:48)
Text for Label1
Text for Label2
Back

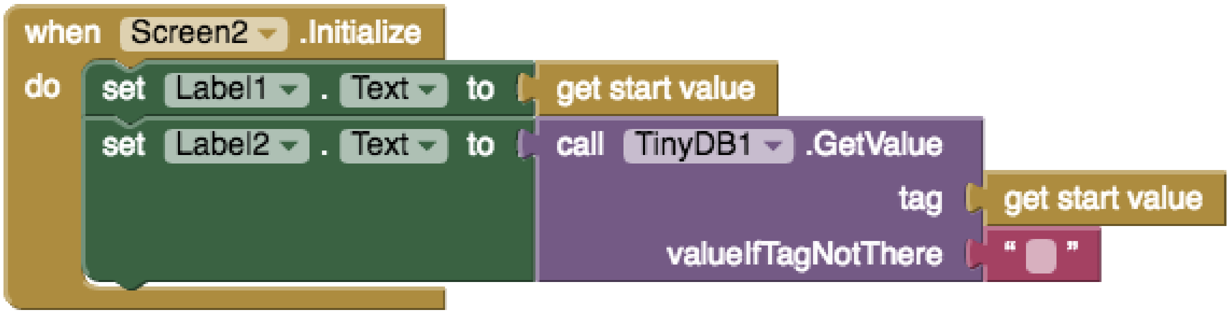
Non-visible components: TinyDB1

Components:
- Screen2
  - Label1
  - Label2
  - Back
  - TinyDB1

Media

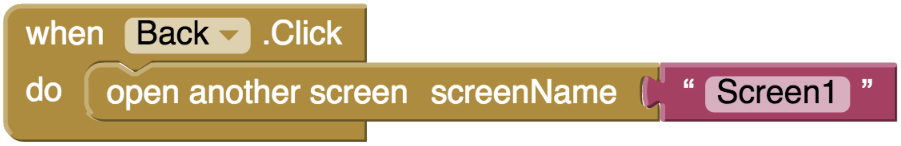
Properties: Screen2
- AboutScreen:
- AlignHorizontal: Left : 1 ▾
- AlignVertical: Top : 1 ▾
- BackgroundColor: □ White
- BackgroundImage: None...
- CloseScreenAnimation: Default ▾
- OpenScreenAnimation: Default ▾
- ScreenOrientation: Unspecified ▾
- Scrollable: □
- ShowStatusBar: ☑
- Title: Screen2
- TitleVisible: ☑

Since the start value is the name of the scientist, we displayed the start value in **Label1** and also used it to retrieve the scientist's description from the database to display in **Label2**. You can find the *get start value* block in the *control* category.



```
when Screen2 .Initialize
do  set Label1 . Text to  get start value
    set Label2 . Text to  call TinyDB1 .GetValue
                                              tag  get start value
                                valueIfTagNotThere  " "
```

Don't forget to program your back button to bring you back to the first screen!



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

Final Check Point

Make sure everything in your app is working! Does selecting an item in the ListView bring you to Screen2 and display the tag name and database value?

Still stuck? Download our source code. Here are **instructions** on how to download and use source codes.

DOWNLOAD THE SOURCE CODE

English