CSCU9A2 Practical 5B (Week 7) Spring 2017
GUIs and Arrays 2nd March

If you get stuck or need help at any time during the practical, ask a demonstrator.

Remember: Week 4's checkpoints (sheets 3A and 3B) must be completed by the end of this week. You have until the end of week 9 to complete this week's checkpoints.

THIS WORKSHEET:

In this worksheet you will build an interactive GUI for a simple address book. Then you will carry out a series of array exercises.

A SIMPLE ADDRESS BOOK "DATABASE"

At this step you will build an interactive GUI for a simple address book.

1. First you need to create a new basic application from an existing file as you have done before:

   In a new folder in your CSCU9A2 working folder: Create a new BlueJ project using the file **Addresses.java** from **Groups on Wide (V:)\CSCU9A2\Java**

   Compile and run the program. Note that, as provided, the "database" contains only one name entry — you will add more data below.

   This is a simple database (an array) holding a fixed collection of **`Strings`**. There is also a global instance variable **`selected`** which will be used to hold the index of a "selected" item in the database.

   The entire name list is to be displayed in the window in a column down the left hand side of a graphics panel. In addition, the "selected" item is displayed to the right, that item being selected through use of the slider. [Using a slider to select an item guarantees that we cannot select an invalid array index by mistake — provided that the slider's range is correctly set up to reflect the valid index range for the array!]

   Some of the code for this functionality is present, and some is not. Carry out the steps following, gradually increasing the functionality of the program. Compile, run and test after each change.

2. **There is not much data in the "database" — just one item so far.** Add another *five items* to the array holding the data (so it holds six entries), like this:

   (a) Because the array will be larger, you will need to adjust all parts of the program that refer to the *number of elements in the array*: There are currently *two* numbers in the program that are determined by the array size: the 1 where the array *is created*, and the second 0 in the *slider constructor* (this is the *maximum valid index* for the array as originally created!).

   - Where the `names` array is *created* in `setUpData`, change the `1` to `6`.
   - So that further changes in the array size are easier to deal with, it is convenient to use `names.length-1` in place of the second `0` in the slider constructor – do that now.
     Remember to do the same thing wherever you need to mention the size of the array later on. Then the program is "self-customizing" when the array size is changed.

   (b) Add five assignment statements to the `setUpData` method to give values to the extra five elements of `names` – you make up the new names!

3. To display the new names on the screen add another five `drawString`s to method `displayList`, like the one that is there already, to display each of the new names in a column at y coordinates `y+15, y+30,` etc. Compile, run and make sure that it works.

4. Can you imagine how awkward the previous step would have been if there had been, say, 500 names! A better solution is to use a *loop*:

   Either delete or put comment bars to the left of your six `drawString`s in `displayList`. Then un-comment the outline `for` loop that you can see in that method.

   The loop body is already correct. You need to customize the *control part* of that `for` loop to display all the elements of `names`. Remember the roles of the three parts of the `for` loop control: at what value should the loop counter `i` *start*, what test do you need for continuing with the loop body, and how should `i` change at each repetition? Make sure that the loop test is "self-customizing" to the array size.

   Compile and make sure that it works.

5. As a result of the changes above, the slider should now have a selectable range of values (0 – 5), but as it is adjusted the "current selection" message that is

displayed does *not* change — it is always element 0 that is displayed. So the slider is not really "selecting", although the value in the global variable `selected` is updated.

The `displaySelected` method is responsible for displaying just the selected item. Look at it, and work out why it is always element 0 that is displayed – it is something quite simple, and not something complex.

Correct the fault, compile and check that it works properly by displaying different items as you adjust the slider.

6. Temporarily alter the slider constructor in `createGUI` to specify 100 (yes, one hundred) as the upper limit of the slider. This is, of course, well above the valid index range of the array of names.

   When you compile and run the program, all should be well until you adjust the slider one step beyond the sixth entry in the database: then BlueJ will pop up its Terminal Window, it will display the details of an `ArrayIndexOutOfBoundsException` in the window, and the program's graphical display will be faulty. You may need to scroll the text in the terminal window down to see the start of the exception report. Make sure that you can pick out from the exception report the *line number* in *your program* where the failure occurred (it is given at the end of the second line of the exception report), and check that that line in your program is the array element display step in the `displaySelected` method – this is where an invalid index was used to attempt to access an element of `names`.. [If you click on the second line of the exception report, BlueJ will open its editor window and highlight the relevant line.]

   The display is faulty – some or all of the expected information is missing – because when an exception occurs, the JVM immediately abandons the current event handling process and returns to waiting for the next event – the display is unreliable!

   Alter the slider limit back to its sensible value before continuing.

7. Now we have selectable names in a column, but no addresses nor telephone numbers! It's time to add them.

   Add a second and third array of `String`s: These are to hold the addresses and telephone numbers; each element `i` of the new addresses/telephone numbers array is to hold the address/telephone number of the person in `names[i]` . Set up the arrays in the same way as `names` (using the same `size`), and fill them with data in `setUpData` (you make it up!).

In `displaySelected`, add statements to show the address and telephone number below the selected person's name – so now the slider is used to choose a name, and then the details are shown.

Remember: Your code must be well formatted and clearly commented.

Checkpoint [Addresses]: Show a demonstrator your program correctly displaying the whole name list using the `for` loop, and displaying the name, address and telephone number selected by the slider. Answer any questions they ask you.

SOME ARRAY EXERCISES

In this exercise you will use a ready-built "array laboratory" to develop fluency with array manipulation.

8. First you need to create a new basic application from an existing file as you have done before:

   In a new folder in your CSCU9A2 working folder: Create a new BlueJ project using the file **Arrays.java** from **Groups on Wide (V:)\CSCU9A2\Java**

   This is a simple example that manages an array of 10 strings, rather like the Addresses database. It always displays the current contents of the entire array on the screen in a text area, with one "selected" item indicated by an asterisk *. The array is already set up in `setUpData` with the word "Alpha" in element 0, "Beta" in element 1, and so on. There is a slider already configured for choosing which item is to be "selected", and a button for causing some action to be taken (some modification of the contents of the array). When the button is clicked the `actionPerformed` method calls the `doSomething` method, where the action of interest is coded, following which the text area is updated to show the new/modified contents of the array. *The exercises below consist of programming different actions to be taken when the button is pressed.*

   Compile and run the application. Play with it a little. As given, the program initially displays "Alpha", "Beta", "Gamma", etc, and the action taken when the button is clicked is to assign the string "Hello" to all the array elements, and the screen is updated to show the new array contents.

   Read the program code: it is quite standard in structure. The array of strings is called `items`. Three "exercises" are already solved in the `doSomething` method ***near the end of the program code*** (two of them are "commented out").

Here are various changes for you to try out. ***All of the exercises*** just consist of putting a new short algorithm into the body of the `doSomething` method, so that the action happens when the program's button is pressed (**you should not change any other parts of the program**).

After each exercise you should "comment out" that part of the code in the body of `doSomething` *(rather than deleting it), leaving it for reference, and for copying and pasting later if need be. BlueJ's editor makes it easy to "comment out" a section of program: highlight the lines, and then select* ***Edit*** *menu/****Comment*** *(or press F8) – there is an "Uncomment" too.*

Compile and run after each change, testing carefully to make sure that the results displayed are correct, especially with the slider set at and near the extreme ends of its range. Make sure that you avoid getting `ArrayIndexOutOfBoundsException`s. None of the exercises requires more than a few lines of Java.

9. Comment out the lines of Java in `doSomething` that set all the array elements to `"Hello"`, and uncomment the single line that just sets element 2 to `"Hello"`. Compile, run and make sure you understand what happens when you click the button — in this case the slider setting has no effect on the action, and although the second and subsequent buttons clicks do actually assign `"Hello"` to the correct array element, of course no further change is apparent on the screen.

10. Now comment out that exercise again, and uncomment the exercise that sets the *selected element* to "Hello". Again, compile, run and make sure you understand what happens when you click the button — in this case you can click, adjust the slider, click again, and so on to change different elements of the array to "Hello".

11. Now comment out the previous exercise, and add statements to the `doSomething` method so that when the button is clicked *all the array elements are set to "Omega"*. Note: when the program starts up, the array should still contain "Alpha", etc, and the "Omega"s should *only* appear *after you click the button*!

12. Similarly, now alter `doSomething` so that when the button is clicked all the array elements from index 0 up to and including the selected one are set to "Stirling".

13. Similarly, now set all the array elements from index 0 up to but *not including* the selected one to "Glasgow".

14. Similarly, now set all the array elements from the selected one up to the end of the array to "Paris".

15. Similarly, now set all the array elements from the one immediately *after* the selected one up to the end of the array to "Berlin".

16. Now set the one array element immediately after the selected one to "Cottrell" — no `for` loop is required for this. This is the first exercise where you need a special check to make sure that your program avoids the ArrayIndexOutOfBoundsException that could happen in one precise situation — the program should just do nothing if the exception would occur.

17. Now, ***additionally to the previous step***, set the one array element immediately preceding the selected one to "Pathfoot" — again, no `for` loop. [And again, the program should do nothing if an exception would occur.]

18. Now *copy* the string in the selected element into *both* the immediately preceding and the immediately following elements — again avoiding exceptions.

Remember: Your code must be well formatted and clearly commented.

SBJ March 2017