BIRZEIT UNIVERSITY

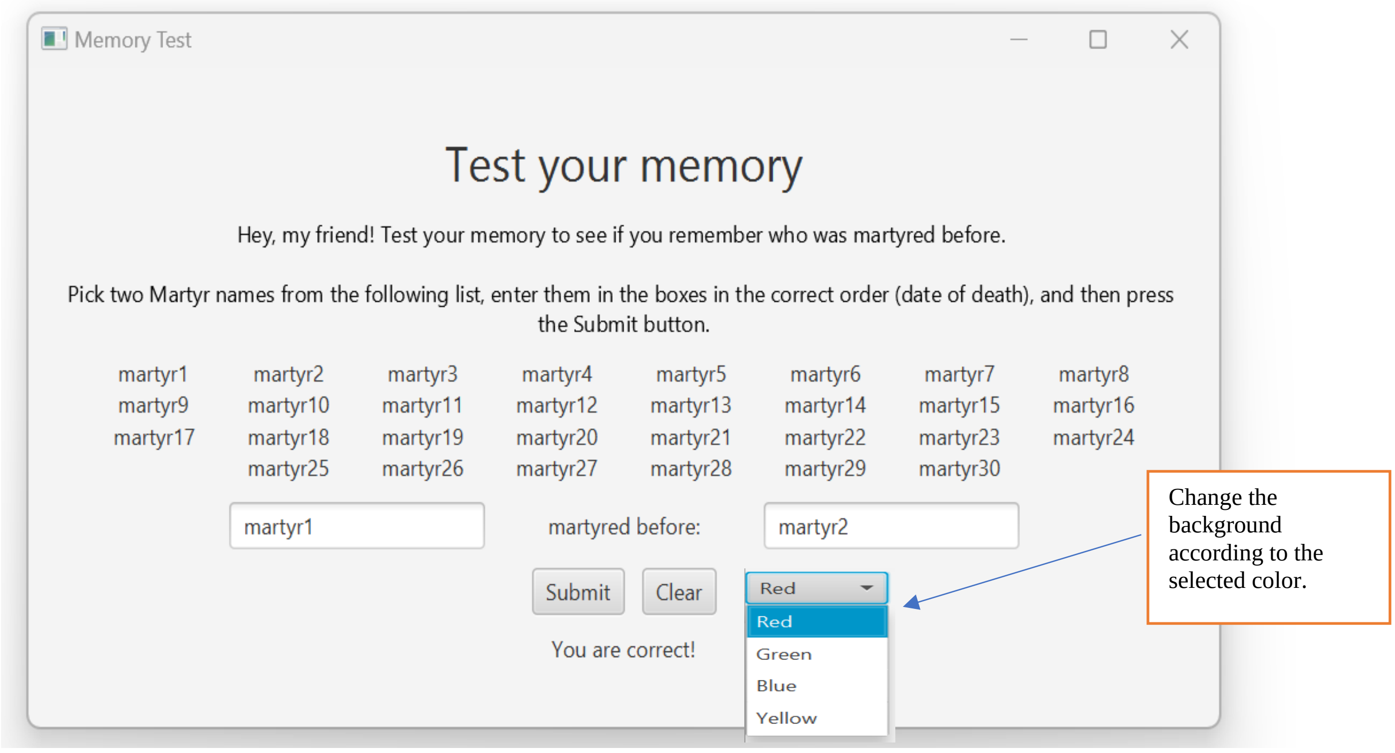
Faculty of Engineering and Information Technology
Computer Science Department
COMP2311 Project (Phase 3)

Project title: The War on Gaza – Memory test application

Project objectives: 1) To create GUI using JavaFx framework
2) To apply the concept of Event-Driven Programming

Project Description:

Create a simple computer program that provides the Palestinian people with a way to remember their martyrs based on their date of martyrdom. Your program should display a list of martyr names and two text boxes. The user is required to type two names into the provided text boxes and then press a Submit button. The program ensures that the entered names come from the name list and are arranged in the correct order of the date of martyrdom. Here is what the display should look like after the user enters the names "martyr1" and "martyr2":



Memory Test

Test your memory

Hey, my friend! Test your memory to see if you remember who was martyred before.

Pick two Martyr names from the following list, enter them in the boxes in the correct order (date of death), and then press the Submit button.

martyr1	martyr2	martyr3	martyr4	martyr5	martyr6	martyr7	martyr8
martyr9	martyr10	martyr11	martyr12	martyr13	martyr14	martyr15	martyr16
martyr17	martyr18	martyr19	martyr20	martyr21	martyr22	martyr23	martyr24
	martyr25	martyr26	martyr27	martyr28	martyr29	martyr30	

martyr1 martyred before: martyr2

You are correct!

Figure 1: Memory Test Window

If the user clicks the **Clear button**, the two names in the text boxes should **disappear**, and the user should be able to repeat the exercise. Notice that the **"You are correct!"** message on the interface should initially be **empty**. If the user does not follow the instructions, different messages should appear instead of **"You are correct!"**, as follows:

- If there is nothing in either of the two text boxes, show:
  ```
  Enter names in both boxes. Then press Submit.
  ```
- If neither of the two entered names is in the list, show:

```
Neither entry is in the name list.
```

- If the first name is not in the list, show:
  ```
  First entry not in name list - check spelling.
  ```
- If the second name is not in the list, show:
  ```
  Second entry not in name list - check spelling.
  ```
- If both names are from the list but are the same, show
  ```
  You entered the same names. Try again.
  ```
- If both names are from the list, but are in the wrong order, show:
  ```
  Wrong. Try again.
  ```

To make the user interface more attractive, add a **combo box** with different colors, allowing the user to adjust the theme of the interface. For example, selecting the color red will change the theme of the interface to red. You can add different colors based on what you find more attractive. You have to use **JavaFX CSS to update the style of the interface**.

Instructions:

All martyr names must be retrieved from the binary file named **"MartyrsList.dat"**. Modifying the martyr names in the binary file will lead to changes in the displayed list of martyrs on the interface. For instance, adding a martyr's name to the binary file will result in that specific martyr being appended to the visible Martyr list on the interface. To read the martyrs from the binary file, you first need to create a GUI window that allows you to add martyrs to this file. Afterward, you can proceed to read them. To do that, please create the following interface.

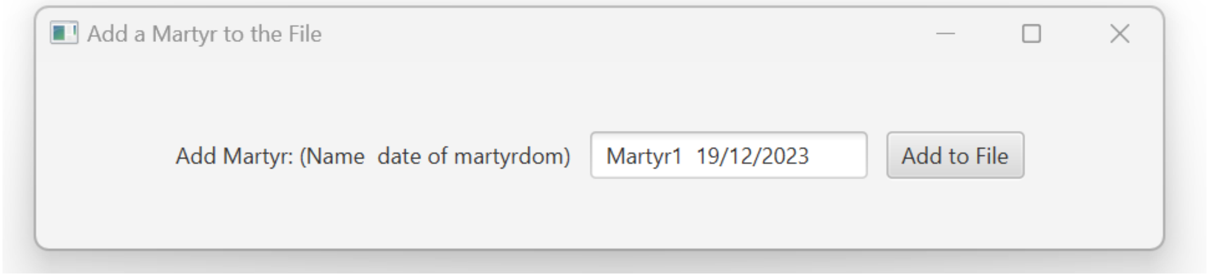


Figure 2: Add Martyr to the File.

You may need an initial window to prompt the user to choose whether they want to add a martyr to the binary file or proceed to the memory window if the file is already created. To achieve this, create the following interface.

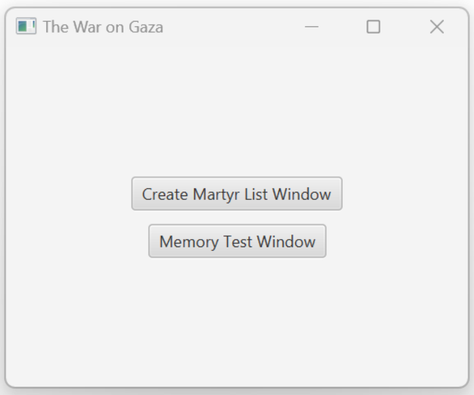
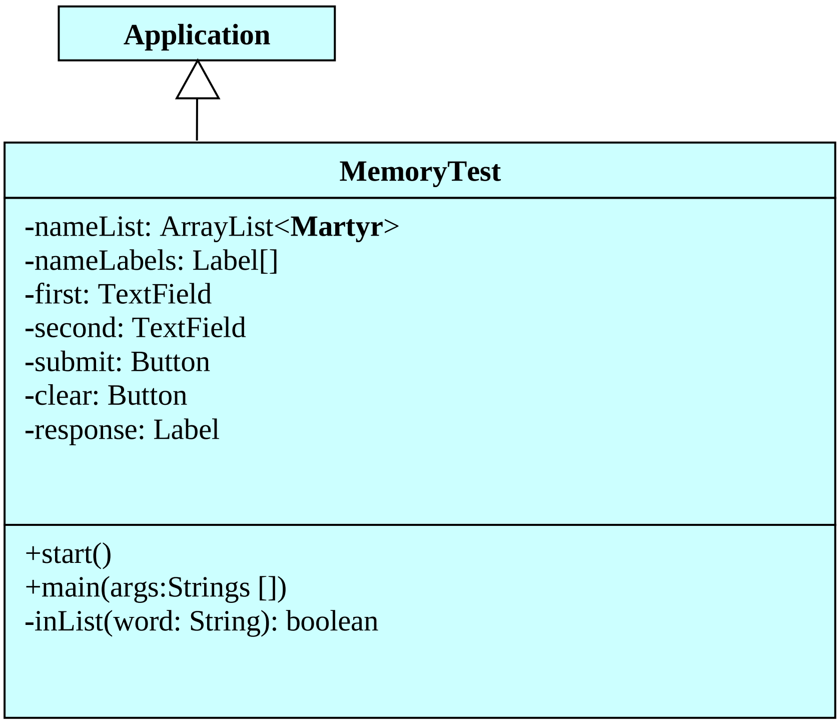


Figure 3: initial window

If the user selects the “Create Martyr List Window”, one action must be performed, which is to display the interface shown in Figure 2 above. The main objective of this window is to append martyrs to the binary file if it already exists. If the file does not exist, the goal is to create the file and add martyrs to it. After closing this window, it implies that the file should be updated and ready for reading.

If the user selects the “Memory Test Window”, two actions must be performed: first, reading the martyr names from “MartyrsList.dat”, and then running the “Memory Test Window” as shown in Figure 1.

To assist you, we have provided the UML class diagram along with additional details.



inList(name: String): to check whether the given name is in the name list.

The **Martyr class** may have:
- martyrName: String
- dateOfMartyrdom :String
+ Setter and getter methods.

Your file should include both the names of martyrs and the dates of their martyrdom. Using these dates, the program can determine who was martyred first. Please note that each line in the file will represent a martyr and their date of martyrdom in binary format, not in text format (you may use readUTF():String / writeUTF(s:String))

<u>Please note the Followings:</u>

1. This is an **individual assignment**. Disciplinary action will be taken against those who **cheat**. Additionally, the use of **AI tools** for generating solutions or **copying from websites** is strictly prohibited. Students found in violation of these policies will face severe consequences. It is crucial to ensure that all work submitted is **your own** and adheres to the **guidelines provided for this assignment**.
2. Make sure you submit your project through **ITC** before the deadline (Submit all **Java files (.java) and binary files (.dat))**.
3. Due date is Friday 26/01/2023, at 11:00 pm.

"Success is the sum of small efforts, repeated day in and day out."

Robert Collier

<u>All the Best</u>