

# Problem Solving Session

- The remainder of today's class will comprise the **problem solving session (PSS)**.
- Your instructor will divide you into **teams of 3 or 4 students**.
- Each team will **work together** to solve the following problems over the course of **20-30 minutes**.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.



Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

# Problem Solving Team Members



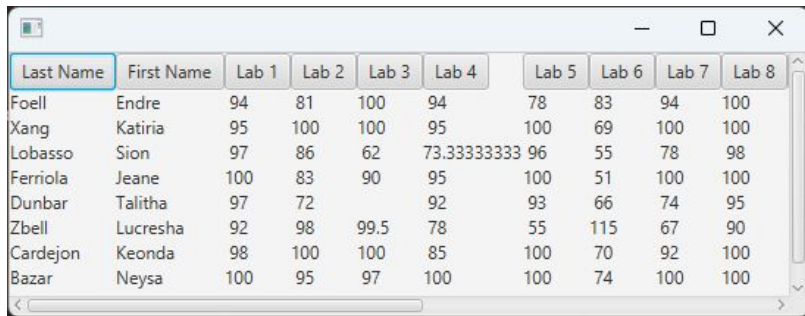
Record the name of each of your problem solving team members here.

Do not forget to **add every team member's name!**  
Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

Abigail Cawley
Dessa Shapiro
Nalin Cooper

# Filters

You have been provided with a GUI that will read data from a comma separated file and display it as a table using JavaFX.



Last Name	First Name	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8
Foell	Endre	94	81	100	94	78	83	94	100
Xang	Katiria	95	100	100	95	100	69	100	100
Lobasso	Sion	97	86	62	73.33333333	96	55	78	98
Ferriola	Jeane	100	83	90	95	100	51	100	100
Dunbar	Talitha	97	72		92	93	66	74	95
Zbell	Lucresha	92	98	99.5	78	55	115	67	90
Cardejon	Keonda	98	100	100	85	100	70	92	100
Bazar	Neysa	100	95	97	100	100	74	100	100

One of the goals of the assignment is to use the buttons to sort that columns in ascending order. To achieve this, you will need to sort the entire data set for a specific column.

The data is already stored as a `List` of `String` arrays. For example to sort by "Last Name" you would use column index 0.

Given this information, complete the `Comparator` on the right so that it will sort the data by "First Name".

```
private List<String[]> data;

public FirstNameComparator implements
    Comparator <String[]> {

    @Override
    public int compare(String[] a, String[] b) {
        return a[1].compareTo(b[1]);
    }
}
```

```
public FirstNameHandler implements
    EventHandler <ActionEvent> {

    public void handle(ActionEvent e) {

        System.out.println("First Name button
                            pressed");
    }
}
```

```
...
Button button = new Button("First Name");
button.setOnAction (

(e) -> {
    System.out.println("First Name button
                        pressed");
}
```

## EventHandler Lambdas

Currently the buttons in the header do nothing. Using the code on the upper right as an example, update the code in the lower right to create an `EventHandler` for the button as a lambda function.

To be clear, you should not call the `handle` function or create a `FirstNameHandler`. Instead, recreate the same functionality as a lambda expression.

# Streams

Currently the UI uses the code on the upper right to generate the data structure.

`BufferedReader` has a `lines` method that returns the reader as a stream of strings. Using this information refactor the provided code to use a stream to create/fill in the `data` list.

Some stream operations you might find helpful are:

`Stream.forEach(action)` - performs the specified action on each element in a stream.

`Stream.collect(Collectors.toList())` - adds each element in the stream to a collection and returns it as a list

`Stream.map mapper)` - applies the mapper 'function' to each element in a stream

*Note: It is not possible to use all of the above and you are welcome to use other `Stream` methods discussed in class.*

```
// Get the header
String line = fin.readLine();
line = fin.readLine(); // First data row
while (line != null) {
    String[] record = line.strip().split(",");

    // Store all the data in a list
    data.add(record);
    line = fin.readLine();
}
```

```
fin.lines()
    .map(line -> line.strip().split(","))
    .forEach(record -> data.add(record));
```

Round everything to the same point

Color coordinate - to make more readable

Fix the spacing a little/ add borders

Replace the background of the buttons with a solid color

Make the buttons and labels take up the full space available

Stylize the label buttons

## Make It Pretty

Currently the UI is fairly sloppy looking. In the box to the left, describe the changes you would perform to make the displayed data look more "professional".

You do not need to write code, simply list the changes you would make.

Last Name	First Name	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8
Foell	Endre	94	81	100	94	78	83	94	100
Xang	Katiria	95	100	100	95	100	69	100	100
Lobasso	Sion	97	86	62	73.33333333	96	55	78	98
Ferriola	Jeane	100	83	90	95	100	51	100	100
Dunbar	Talitha	97	72		92	93	66	74	95
Zbell	Lucresha	92	98	99.5	78	55	115	67	90
Cardejon	Keonda	98	100	100	85	100	70	92	100
Bazar	Neysa	100	95	97	100	100	74	100	100