Dear Students,

The objective of this lab is to cover Event-Driven Programming in Java.

First, please review and practice the concepts we discussed in the last lecture, where we went over code examples implementing the three recommended types of listener interfaces. You can find the relevant code examples in your slides.

Once you have completed the practical examples provided in the slides, please attempt all of the following Exercises (1-4).

Exercise 1: Dynamic Font Resizer Event Controlling

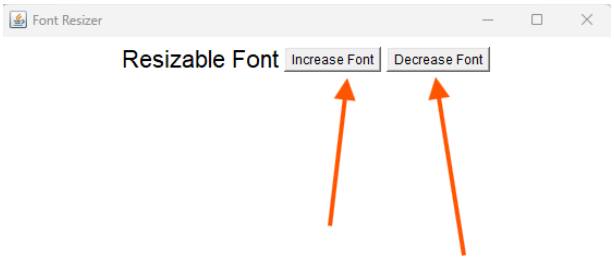
1. Create a GUI with a **Label named as** label and two buttons: **Increase Font (btnIncrease)** and **Decrease Font(btnDecrease)**.
2. Add functionality so that:
   - Clicking **Increase Font** increases the label's font size.
   - In the increase font Event Source use the following code or use your customized source

```
Font currentFont = label.getFont();
label.setFont(new Font(currentFont.getName(), currentFont.getStyle(),
currentFont.getSize() + 2));
```

   - Clicking **Decrease Font** decreases the label's font size.

```
Font currentFont = label.getFont();
int newSize = currentFont.getSize() - 2;
if (newSize > 8) {
    label.setFont(new Font(currentFont.getName(), currentFont.getStyle(), newSize));
 }
```

Sample Demo:



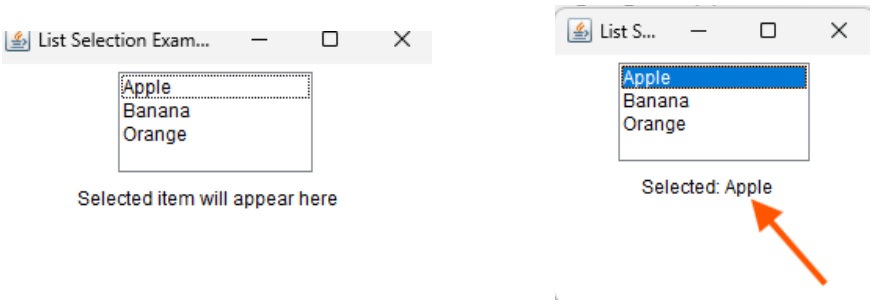
Exercise 2: Mouse Events

1. Design a GUI application with a Panel using AWT.
2. Implement mouse events to:
    - Change the panel's background color to **yellow** when the mouse is clicked inside it.
    - Change the color to **green** when the mouse enters the panel.
    - Revert the color to the original **light gray** when the mouse exits the panel.

Exercise 3: List Selection Event

1. Create a GUI with a List component containing items: **Apple, Banana, Orange**.
2. Implement an event listener to display the selected item in a Label when the user clicks on any list item.

    Sample Output Looks like the following



Lab Exercise 4: Handling Key Events with TextField and Label

Create a Java GUI application using AWT that includes:

1. A TextField for user input.
2. A Label that displays the text typed in the TextField.

Use a KeyListener to listen for key events and update the label as the user types in the TextField.

Instructions:

1. Create a Frame to hold the GUI components.
2. Add a TextField with a capacity of 20 characters.
3. Add a Label below the TextField that will display the typed text.
4. Use a KeyListener (specifically KeyAdapter) to monitor key events in the TextField.
5. When the user types in the TextField, update the Label to show the text typed so far.
6. Ensure the program properly handles the closing of the window.

Sample Output



Add the following event handler to you GUI

```
textField.addKeyListener(new KeyAdapter() {
          @Override
          public void keyReleased(KeyEvent e) {
              label.setText("You typed: " + textField.getText());
         // this code will update the label with typed text
          }
      });
```