F27SB Software Development 2

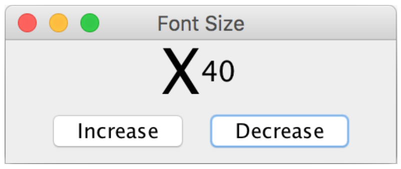
GUI Coursework 2: Buttons and Listeners

Write a program to display font sizes

Layout
It should have:

- In the top:
    - label for letter “X”
    - label for font size value
- In the bottom:
    - button to increase font size
    - button to decrease font size

Desired final look:



Functionality
Initially, the X label should display an 18 point “X”, and the font size label should display the value “18”. Every time the “increase” button is selected, the font size of the “X” and the value in the font size label should be increased by 1. Every time the “decrease” button is selected, the font size of the “X” and the value in the font size label should be decreased by 1. Make sure that this does not allow negative font sizes.

(Correct layout: 0.25P,
Class extends JFrame: 0.25P,
Class follows OOP principles: 0.25P,
Global set-up in main method: 0.25P)

(Class implements ActionListener: 0.5P,
Works as required: 0.5P)[1]

[1] Detailed explanation of points at the bottom of the document.

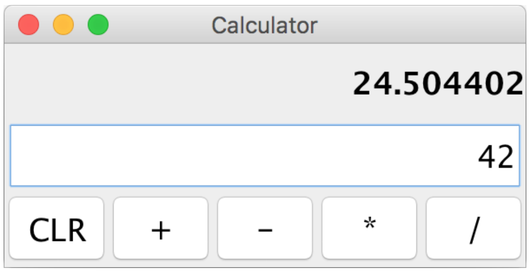
Create a simple calculator

Layout

The calculator has:

- a display label at the top which will be used to show the calculation results
- a text field for the user to input an integer/double in the middle
- buttons for “clear”, “+”, “-”, “*” and “/” in the bottom

Desired final look:



Functionality

- Create an *accumulator* state variable that keeps track of the current overall value of the calculation.
- Clicking the “clear” button, set the *accumulator* value to 0
- Clicking the “+” button, add the text field value to the *accumulator* value
- Clicking the “-” button, subtract the text field value from the *accumulator* value
- Clicking the “*” button, multiply the *accumulator* value by the text field value
- Clicking the “/” button, divide the *accumulator* value by the text field value
- After a button has been pressed, update the display label with the new *accumulator* value and empty the text field.

(Correct layout: 0.25P,
Class extends JFrame: 0.25P,
Class follows OOP principles: 0.25P,
Global set-up in main method: 0.25P)

(Class implements ActionListener: 0.25P,
Uses accumulator variable: 0.25P,
Works as intended: 0.25P,
Input checks to prevent errors: 0.25P)[2]

[2] Detailed explanation of points at the bottom of the document.

Marking

In each coursework, you can achieve a total of 4 points. Each question awards different amounts of points for different parts of the question. Partial completion of a task will award the partial points listed underneath. Most descriptors of points are self-explanatory in context of the task. Where further clarification is needed, you can find that below.

Your work will be checked by a lab helper during your assigned lab slot. Once all tasks are checked, the points will be used to calculate your marks. Please, understand that the lab helpers are not marking your work but are checking the completion of subtasks. As part of this check, you will need to explain how you solved the given task. **Only successfully completed sub-tasks will award points.**

The marks will be released on Canvas after the marking deadline has passed. This is not an automatic process so please, be patient. Once the marks are released, you will be notified via Canvas. Please, **make sure to check your marks as soon as possible**. If there are any issues, please contact your teaching team immediately.

Collusion and Plagiarism

As mentioned above, you will need to explain your work during the demo session in the lab. If you are not able to explain how you arrived at the solution to the task, we need to assume that you did not do the work yourself. We do, however, know that you might be anxious or nervous during the session. Please, rest assured that this is not an interrogation and you can take all the time you need to explain your solution.

If there is reasonable doubt that you solved the given problems yourself, you will not get any points for this task. If there are concrete indications that you copied your answer or colluded with other students, we might also start an official investigation.

Please, make sure to fill and sign the Declaration of Student Authorship form for each coursework and upload it to Canvas. If you do not upload the form, we will not be able to give you any marks for this coursework.

If you feel unjustly accused of plagiarism or collusion, please contact your teaching team.

Coursework submission

Unless stated otherwise, all code parts of your work need to be committed and pushed to your GitLab fork. You need to upload the Declaration of Student Authorship to Canvas, and you need to present your solution to a lab helper. If you fail to do any one of these steps, you will not be awarded any marks.

The deadline for submission can be seen on Canvas. You will need to **present your work to a lab helper any time before the deadline during your allocated lab slot**.

If you do not manage to present your work before the deadline, you can do so at the first lab after the deadline but will incur a 30% late penalty. If this late submission was caused by issues out with your control, you can apply for mitigating circumstances to have this late penalty removed.

If you also fail to present your work at the lab following the deadline, we will not be able to give you any marks for your work. Similarly, if this was caused by circumstances out with your control, you should apply for Mitigating Circumstances.
Please, note that we are not allowed to give individual extensions. If you cannot submit your work on time, you will need to apply for Mitigating Circumstances.

Explanation of Points

Correct layout: Where an image or text explanation of the layout is given, points are awarded for exactly achieving the required layout. You are free to choose a different layout that you find more appealing as long as all the same components are used. This means you might want to choose a different layout manager or different positioning but you should still make sure to have the same amount of labels, panels, buttons, etc. If an example of a layout is given, you are required to match this layout as closely as possible.
Class extends JFrame: The class that declares and initialises all the GUI components should be a subclass of JFrame and correctly use the methods and variables provided by its super class to achieve the desired layout.

Wrong	Wrong	Right
`public static void main() {` `JFrame f = new JFrame();` `f.add(new JLabel());` `…`	`public class A {` `public A() {` `JFrame f = new JFrame();` `f.add(new JLabel());` `…`	`public class A extends JFrame {` `public A() {` `add(new JLabel());` `…`

Class follows OOP principles: You should endeavour to reduce coupling and increase cohesion in your code. That means you should check that every class is responsible for its own components and data and that you do not provide unnecessary public variables or methods. Always think about how someone else could use your class in a different project. Refer to all the guidance given in the first half of the course and make sure to highlight this to the lab helper when explaining.
Global set-up in main method: When you think about someone using your code in their larger GUI, do you think they are happy that the name, size, position, and visibility is defined inside the class where it cannot easily be changed? Make sure to only define these things in the main method where you create a new instance of your class as we do in the lectures.
Class implements ActionListener: To increase cohesion, the class that declares and initialises the components that can trigger action events should be the one that is the action listener. Make sure that this class implements the ActionListener interface and overrides the actionPerformed method.
Input checks to prevent errors: Make sure that the calculator doesn't break regardless of what the user does. What would happen, for example, if someone types in a letter instead of a number? Make sure to highlight what you have done to the lab helper.