Computational Biomedical Genomics

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Lecture WS 2023/24: Introduction to Data Structures and Programming for Life Scientists

Assignment No. 6 (10 + 1 points)

Hand out: Monday, December 11, 2023
Hand in due: Wednesday, January 10, 2024
Direct inquiries to stephan.ossowski@med.uni-tuebingen.de

1. Graphical User Interfaces with Swing (10P)

Implement a calculator using the Swing API. The calculator should look like a pocket calculator (see example images on page 2). The following sub-tasks are required to fully complete this exercise:

a) Implement a class 'Calculator' that uses a JFrame and Swing components (optimally instantiated in the Constructor) to provide a GUI resembling a pocket calculator. Closing the calculator window should end the program. (2P)

b) Use Layouts to make the calculator visually appealing. The layout of the calculator should at least contain the following elements: numbers 0 to 9, decimal point, operations +, -, *, /, a clear button, and a '=' button that starts the calculation (all implemented as JButton), as well as a display (JTextField or JLabel, which should not be used for user interaction, just displaying). (2P)

c) Implement the functionality of all buttons using ActionListeners. If numbers are pressed they should be displayed on the JTextField or JLabel and the value stored in a variable. If an operator is pressed the requested operation needs to be stored, and the program has to be prepared to receive the second number. The clear button should reset the program (clear display, clear stored numbers and the operation). (2P)

d) Implement the mathematical functions in the ActionListener of the '=' button. The minimum functionality should allow the user to perform an operation with exactly 2 numbers, e.g. 7 * 3 (therefore you need 3 variables for 2 numbers and one operation). The result will be displayed and afterwards the program is reset for the next operation (but the result of the last operation should for now remain on the display, until the user starts a new input). (2P)

e) Extend your calculator to allow for calculations with more than 2 numbers and more than one operator (use array, ArrayList, Vector, LinkedList or HashMap to store multiple numbers and operations). Add at least one extra operator (e.g. power) and allow for negative numbers (new buttons needed for that). (2 + 1 P)

Example Layouts of pocket calculators from MacOs and a browser application: