SED : Exercise 7 : Interactive Application - Work in Pairs

Using the Java Swing library of GUI components, build an application that implements a Reverse Polish Calculator, where the user can click on the buttons and see the results of calculations.

Look at pages 2 onwards of this spec for details of how to get the skeleton code, test it, and submit.

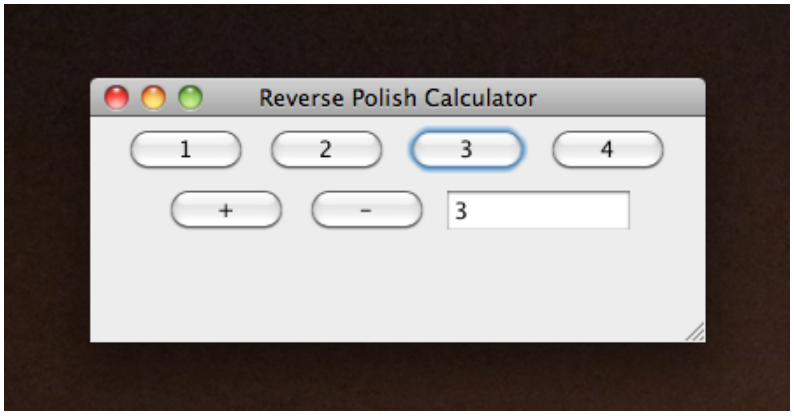
The Reverse Polish Notation

Reverse Polish notation (RPN) is a mathematical notation wherein every operator follows all of its operands. For example, if we want to add 3 and 4, we write 3 4 +

So our Reverse Polish Calculator should allow us to click 3 (and we see 3 in the display), then 4 (and we see 4), and then +, which performs the calculation and shows the result 7 on the display. The calculation is most easily implemented using a stack (perhaps a `java.util.Stack`).
http://en.wikipedia.org/wiki/Reverse_Polish_notation

The User Interface

Create a simple GUI with a window, some buttons for numbers and operators (you could add more than in this example screenshot), and a text field to show the current output.



Add an ActionListener to each button, following the Observer pattern to do something in response to clicks. If you want to tune the layout, Google for "swing layout managers".

Unit Tests

Which parts of your code can you write unit tests for? Running your unit tests should not cause any GUI components to display on the screen.

Design Considerations

Is your code good quality?
Do you have a clear separation of concerns? Can you separate out a Model, View and Controller?
Do you have any duplication? Can you reduce it?
Are there any error cases to consider?

Getting The Skeleton Code

Get the skeleton from GitLab:

```
git clone https://gitlab.doc.ic.ac.uk/lab2021_spring/70055_ex7_login.git
```

Project Structure

When you clone from GitLab, you should find a similar structure to what we had in the first exercises, but there is no starting code. You are free to create whatever classes you need.

```
├── build.sh
├── config
├── build.gradle
└── src
    ├── main
    │   └── java
    │       └── ic
    │           └── doc
    └── test
        └── java
            └── ic
                └── doc
```

Do follow the existing structure of directories and packages for code and tests, as per previous weeks. Delete the README.txt files once you have added your code.

Running the Build

This is the same as previous weeks, you can just type `./build.sh`

Make sure you run `./build.sh` before you submit, as this is what the autotest will run.

The coverage report this week does not have a minimum percentage to meet, but you should use your judgement to unit test your code as you think best.

If you have completed the required functionality, the build passes, and you are happy with your code then you are ready to submit.

Submission

When you have finished, make sure you have pushed all your changes to GitLab (source code only, not generated files under *target*), test on LabTS and submit your token to CATE.

Deadline

There is an online lab session timetabled for Monday 10:00-12:00 UK time where you can talk to us about the exercise. This is not intended to be a large exercise, so it should not take a lot of time.

The deadline for submission to CATE is **7pm UK time on Wednesday 3rd March**.

Only one person (whoever cloned the repository originally) needs to submit from LabTS to CATE. Once this is done, add your partner as a group member on CATE. The other person should sign the submission on CATE to confirm.

Assessment

The markers expect that your submission passes the automated tests and checks:

- Code compiles
- Tests pass
- Style check passes

If you pass these automated checks then the markers will review the design of your code and award marks based on:

- 10% Effective implementation of calculator
- 10% Effective separation of concerns following the Model View Controller pattern
- 10% Effective testing of relevant parts of the code
- 10% General code quality, clarity, freedom from duplication etc
- 10% Bonus marks for particularly good code or design (at the marker's discretion)

Marks may also be deducted for poor quality code, errors, or for not meeting the requirements.