Programming: Week 20 Lab (GUI)

Learning objectives:

- Developing simple GUI programs using the Java Swing classes.

Resources

- Week 20 Lecture Slides – look at the **Greeter** application
- **Greeter.java** (Moodle, Week 20).

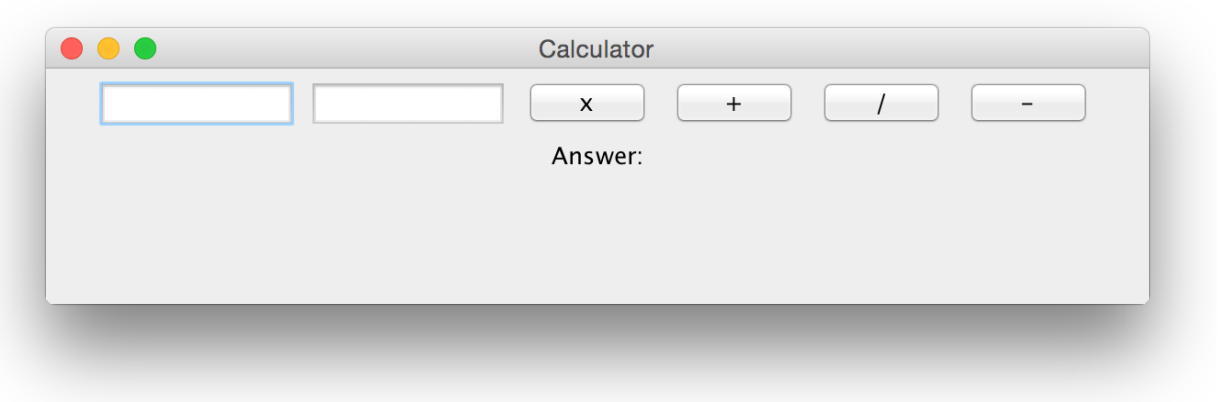
Exercise

Write a simple calculator program which allows the user to enter two decimal numbers, and compute their sum, difference, product or quotient.

You should have two **JTextField** components that allow the user to input two numbers (treat them as floats), and four **JButton** components which are labelled x, +, /, -. The result of the calculation is displayed using a **JLabel**.

Part 1

Create the application framework, and add all the components. Use the **Greeter** application from the Week 20 Lecture as an example. You will need to derive a class from **JPanel**, and add two **JTextField** components, four **JButton** components and a **JLabel** for the result. You can add the **actionPerformed** method, but for now it should be an empty method.



When this stage is complete, you should have a window that looks like the picture above. You should be able to type text into the text fields, and press the buttons, but the inputs will have no effect.

Part 2

Now add the functionality. You will need to respond to a button press as follows:

1. Read the strings from the **JTextField** objects and convert them to floating point numbers.
2. Calculate the result by applying the appropriate operation, and write the output to the **JLabel** (or an error if the input cannot be converted to a number).

Hints:

1. Make sure all the buttons send events to the **JPanel** class (you should call **addActionListener** on each one – see **Greeter.java**).

2. Convert the strings to floats using the **Float.parseFloat(String)** method. This returns a **float**, but throws an exception if it can't do the conversion. You can use the code below in your solution:

```
try
{
    // get the operands
    float val0 = Float.parseFloat( input1.getText() );
    float val1 = Float.parseFloat( input2.getText() );

    float result = 0;
    // calculate result here...
    answer.setText( "Answer: "+result);
}
catch (Exception e)
{
    answer.setText("invalid input");
}
```

What this does is *try* to execute all the code in the **try** block. If it succeeds, the **catch** block is ignored. If an exception is thrown by any of the code in the **try** block, the **catch** block *is* executed. The details of the exception can be accessed using the local variable **e** – in this case we don't query this and just give out a blanket error message. We will cover exceptions in more detail next week.

3. Think carefully about how you use the code in hint **2.** Would it be a good idea to cut and paste this four times into the **actionPerformed** method, once for each button? Probably not. Much better would be to write a single method which reads the input and performs the calculation. You could define an enumeration for the operations, and then pass that into a calculation method. Something like:

```
enum Operation
{
   TIMES, PLUS, MINUS, DIVIDE
}

public void actionPerformed( ActionEvent event )
{
    if ( event.getSource() == buttonTimes )
        Calc( Operation.TIMES );
    else if ( event.getSource() == buttonPlus )
        Calc( Operation.PLUS );
    else if ( event.getSource() == buttonMinus )
        Calc( Operation.MINUS );
    else if ( event.getSource() == buttonDivide )
        Calc( Operation.DIVIDE );
}

void Calc( Operation operation )
{
    // ... do stuff. A switch statement may be useful here...
}
```

Refer to week 11 for a refresher on enumerations. Test your code with a range of inputs – how does it respond to a division by zero?

Portfolio Exercise

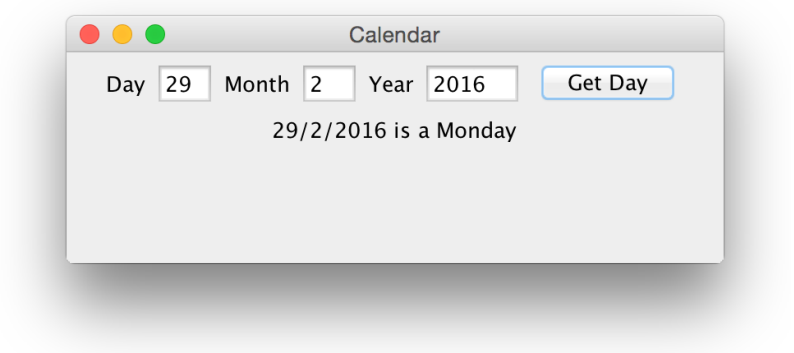
The portfolio task is to write a day of the week calculator which allows the user to input a date, and which calculates and outputs the day of the week on which that date fell (or will fall). Download **Doomsday.java** from Moodle. This is a class which implements John Horton Conway's 'doomsday' algorithm for calculating the day of the week for any date. The class has a static function

```
public static int GetDayOfWeek( int day, int month, int year )
```

This returns -1 if the date entered is invalid. For valid dates, it returns 0 for Sunday, 1 for Monday, and so on. Call it like this:

```
int dayOfWeek = Doomsday.GetDayOfWeek( 29, 2, 2016 );
System.out.println( dayOfWeek );
// outputs '1' - 29/2/2016 was a Monday.
```

Your task is to write a GUI for this class. It should look something like the picture below.



The program should include the following features:

- The day should be given by name, not the integer returned from the function.
- It should output an error if the input cannot be converted into integers.
- It should output a different error if the input can be parsed but is not a valid date.
- The output should include the date.

Example outputs:

'Invalid input'
'45/32/2020 is not a valid date'
'29/2/2016 is a Monday'

Don't change anything in Doomsday.java. You should include it in your project, but you don't need to modify it.

Hints

- Convert **String** to **int** using **Integer.parseInt(String s)**.
- Use a similar exception handling mechanism to the calculator application to catch invalid (non-integer) inputs.
- You could use a **switch** statement in your code to convert an integer day to a string.

Extension Exercise

Find out how to get today's date in a Java program. Use this to modify the code so that the output takes the current date into account – for example:

'29/2/2016 was a Monday'

'29/2/2020 will be a Saturday'