CSC 211 — Introductory Programming and Design

Laboratory Assignment 07: GUI Time Converter

Monday, March 10, 2008

Due date — Section 1: Wednesday, March 26, at 1 PM
Section 2: Tuesday, March 25, at 1 PM.

1 About this Assignment

1.1 Objectives of the assignment

The main objectives of this assignment are for you to

- start learning about layouts and other GUI elements;
- learn to handle action events;
- start using more complex arithmetic expressions in your code.

You are going to write a simple converter between a total number of microseconds and centuries, years, weeks, days, hours, minutes, seconds, microseconds. For example, if the input is 97428438420480000782 as the total number of microseconds, your program should determine that this corresponds to 5 centuries, 24 years, 17 weeks, 5 days, 14 hours, 51 minutes, 23 seconds, 256 milliseconds, and 782 microseconds.

Read very carefully the text of this assignment before rushing to type code. Read it completely through once. Do not stop at the first unclear sentence you encounter; sometimes things are explained and detailed a bit later. Then you should start all over again. This time stop to ask questions when a point remains fuzzy, code your solution, and move on to the next section.

1.2 Handouts

The sole handout this week is the text of this assignment as a pdf file.

2 What to Do, Part I: Learn about Layouts

2.1 About the Swing “trail” tutorial

This semester we are going to keep learning elements of Swing (a part of the Java API that deals with the graphical user interface). Luckily, Java developed an excellent tutorial on Swing. I have
posted a link to this tutorial on the course’s web page (“favorite links” on the right side). This tutorial gives a list of all the types of GUI widgets supported by Swing (radio buttons, tabbed panes, sliders, layouts, etc), with pictures of examples of these widgets. It also provides sample code demonstrating how to create, set up, and use these widgets. Using this resource, combined with the regular Java API documentation, you should be able to build increasingly sophisticated interfaces for your applications.

From now on, you will be expected to go check these resources to learn about any new Swing widget that is needed in an assignment.

### 2.2 GridLayout

Layouts are ways to specify how your application’s GUI widgets should be arranged. A GridLayout organizes your widget into a rectangular grid. All elements will have the same size. The following code sample creates a JFrame object and assigns to it a grid layout of 3 rows and 2 columns.

```java
JFrame frame = new JFrame("my new frame");
frame.setSize(400, 600);
frame.getContentPane().setLayout(new GridLayout(3, 2));
```

As you can see, we don’t assign the layout directly to the frame but to its “content pane”. This is one of the small oddities of the JFrame class that will make more sense (hopefully) if you invest more time learning about Swing classes.

Having assigned a layout to my frame, I can simply add GUI widgets to it:

```java
frame.add(new JButton("hello"));
frame.add(new JButton("world"));
frame.add(new JLabel("bye!"));
frame.add(new JLabel("later"));
frame.add(new JTextField("type something"));
```

### 2.3 BorderLayout

Another type of layout that is commonly used in applications is BorderLayout. You can for example use the “NORTH” part of your frame to display some text information, the “Center” part for graphics, and the “EAST” part for a grid of buttons.

What is really interesting is that you can embed layouts within each other. Now try to do the following:

- Create a new frame and assign to this frame a border layout
- Create JLabel objects with content “north”, “east”, “south” and add them to your frame at the corresponding locations.
- Create a new JPanel object, assign to this JPanel a $3 \times 2$ grid layout.
- Add to your JPanel the widgets from the sample code above (or your own).
- Add the JPanel to the “CENTER” part of your frame.
- Admire your work.
2.4 Design a layout for your application

For this application you are going to need two JPanel: One in the Center part of your frame will contain text fields and labels that allow the user to enter numbers of centuries, years, weeks, days, etc. and view the corresponding total number of microseconds, or inversely, to enter the total number of microseconds and view the corresponding number of centuries, years, etc.

The second panel, to appear either in the East or West part of the frame (your choice) should contain at least a “Quit/Exit” button, a “Convert” button to compute the transformation, and possibly a combo box to select a different time system (see extra credit section)

3 What to Do, Part II: Action Events

3.1 ActionListener

Last week we learned about MouseEvent. We saw that we had to declare that our JFrame object implements MouseListener and that this implied implementing some methods we predefined names and also adding our frame to a list of MouseListener objects. Things are fairly similar with action events, that is, when events that are triggered by action on GUI elements such as buttons, menu items, check boxes, etc.

The first thing to do is to say that our Lab07 class implements ActionListener. After you do this you will see that you must now implement a method named actionPerformed. This is the method that the JVM invokes when one of your GUI widgets triggers and event.

3.2 Define your frame as the listener for your widgets

In the first part of this assignment, we saw how to add widgets to a layout. This takes care of the appearance of your widget in the frame, but it does not connect it to the logic of your application. To do this, you must attach an ActionListener for your widget. Any object that belongs to a class implementing ActionListener (and therefore contains a method actionPerformed can be defined as the listener for your widget. At this point, because our application is so simple, we are better off defining our frame as the listener for all our widgets. So you can simply declare

```java
JButton b = new JButton("hello"); // creates a new button
add(b); // adds the button to the layout
b.addActionListener(this); // declares that this frame can // handle action events for the button
```

Now you should be ready to complete the assignment

4 Other Implementation considerations

4.1 Beware of overflow

When dealing with arithmetic calculations, one concern that you should always have on the back of your mind is the risk of overflow, that is, of attempting to store in a variable of a given type a
number that is larger than the capacity of this data type. The other very serious (and in fact more common) risk with numerical computing is that of accuracy of floating point calculations. For today’s assignment, only the former is a real issue.

4.2 What to do with the results of the conversion

Converting numbers of centuries, years, etc. into a total number of microseconds is fairly easy. It is just a long expression, and since what is produced is a single number, it is simple enough to write a little method to take care of the conversion.

The reverse conversion is trickier. Besides the fact that it requires more steps, the problem is that the result is not a single number but a number of centuries, a number of years, a number of weeks, etc. A method can only return one value, so what can we do?

The simplest solution here is to create a storage class, whose sole purpose is to store together a set of values (possibly of different types), and to have your method return a new object of that class.

4.3 Months, weeks, and leap years

You certainly noticed that we don’t extract a number of months. This is because not all months have the same number of days. Maybe you will think of a way around that. Another thing you should be aware of is that a year is not made of exactly 52 weeks. Finally, there is the issue of leap years, so that a year does not always count 365 days.

5 For Extra Credit

5.1 Take care of leap years (5 pts)

This is not particularly hard.

5.2 For frequent travelers

For those who travel frequently away from good old planet Earth, it might be convenient to modify the converter to work with different year and day durations.

6 What to Hand in

6.1 End-of-session evaluation

You are not expected to complete the assignment by the end of the lab session, but you are definitely expected to have done some work during that session. Try to use the lab session to make sure you understand everything about the assignment. Ask questions; try things; ask more questions.
You should *not* leave the lab before your work has been evaluated. This first evaluation is worth 10 pts out of 100 for the complete assignment. If you leave before you have been evaluated, these points are lost with no chance of a later evaluation.

### 6.2 Your project

A cleaned-up and properly named project folder containing all your source files and your report should be uploaded to your CSC211 submit folder. Do *not* include the documentation produced by javadoc to this folder.

### 7 How You Will Be Evaluated

#### 7.1 Point distribution

The maximum number of points is 100, but extra points could be awarded for excellent aspects of the project or report. The point distribution for this assignment is as follows:

<table>
<thead>
<tr>
<th>Execution evaluation</th>
<th>10 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-of-session evaluation</td>
<td></td>
</tr>
<tr>
<td>Execution of the project handed in</td>
<td>30 pts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier names</td>
<td>5 pts</td>
</tr>
<tr>
<td>Good indentation and general readability</td>
<td>5 pts</td>
</tr>
<tr>
<td>Judicious comments within the code</td>
<td>10 pts</td>
</tr>
<tr>
<td>Javadoc comments</td>
<td>10 pts</td>
</tr>
<tr>
<td>Follows lab specifications</td>
<td>10 pts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Report</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion</td>
<td>10 pts</td>
</tr>
<tr>
<td>General quality of the writing and presentation</td>
<td>10 pts</td>
</tr>
</tbody>
</table>

If you submit a project late, then it is your responsibility to notify the TA (with CC. to the instructor) that the project is finally available in your submit folder. If you fail to do so, then the “late penalty clock” will keep ticking until the TA gets around to checking your folder and notices your project. Unless specifically asked to do so, do *not* mail your project folder as an attachment.