CS 1110, LAB 13: FORMATTING LOCALES

Name: _____ **Net-ID**: _____

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab13.php

You may wish to use that version of the instructions.

The goal of this lab is to introduce you to the concept of writing a listener to a GUI application. In addition, you will get some experience with "localization", how to customise an application to different cultures. This matters when you are writing to a larger user base.

People in the US sometimes think that they dominate the world, but they are in the minority when it comes to many issues, even formatting numbers and percentages. For example, here is how people in three parts of the world format decimal numbers, currencies, and percentages:

US	English	Russian
1,500,012.253	1.500.012,253	1 500 012,253
$ 1,500.23	£1.500.23	1 500,23
100.075%	100,075%	100 075%

In this lab you will work with other forms of localization.

Requirements For This Lab. There are two files necesssary for this lab, and they are all available from the online version of these instructions at the course web page. You should create a new directory on your hard drive and download the following five files into this directory:

- `GenericGUI` (http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab13/GenericGUI.java)
- `LocaleController` (http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab13/LocaleController.java)

Despite the importance of these files, there is no code to turn in for this lab. Everything that you need to do corresponds to some output for you to write down on this sheet. When you are done, you should show your instructor the contents of the sheet. If you do not finish the lab, then you need to finish it **by the beginning of lab next week** and show it to your instructor at that time.

Starting the Application

As we have discussed several times in class, `LocaleController` is a controller class. It launches the application and provides the methods that give functionality to the various GUI components. To start up this program, compile it and type the following in the Interactions pane:

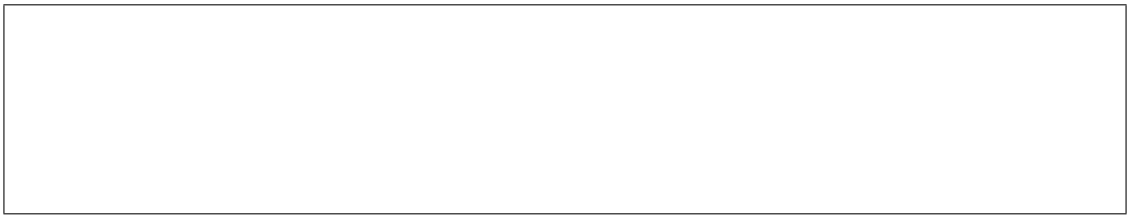
```
run LocaleController
```

This will execute the method called `main`.

Adding a Listener. Currently, the application does nothing when you press the "Ready!" button; that is because it does not have a listener. The class `LocaleController` implements `ActionLister` and can function as listener, but it must be added to the view. To add an instance of `LocaleController` as a listener, you must modify the constructor, which is where the view is created. Add the following line:

```
view.getButton().addActionListener(this);
```

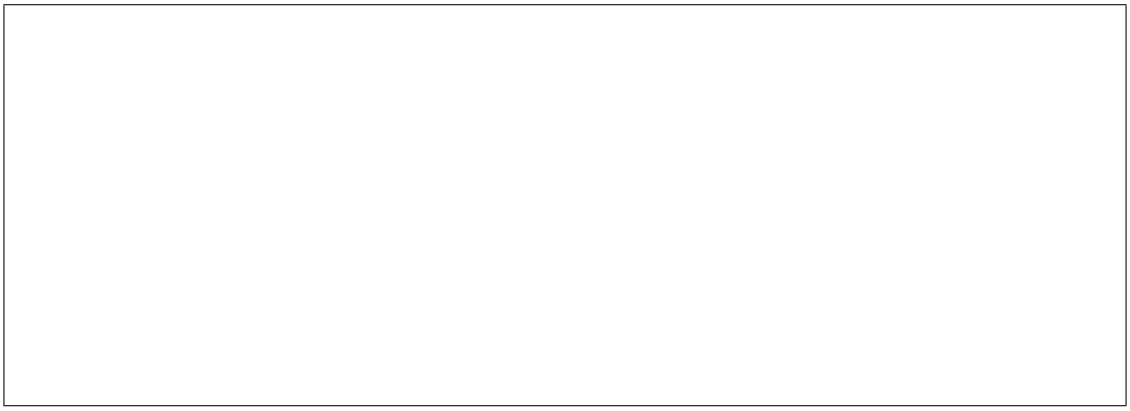
Once you have done this, recompile and hit the "Ready!" button. The number will be printed in three ways in the String fields – number format, currency format, and percentage format – all for the chosen locales. Write what you see below.



Testing the Locales

Take a look at the method `actionPerformed` and `print3` of class `LocaleController`. You will see how a number like 5000.365 (placed in `double` field 0 of the GUI) is printed in the decimal format, currency, and percentage format of the locales selected in the pulldown box at the top of the GUI. The default locale is English (U.S.), but you can use the pulldown box to select one of over 100 locales.

Execute the program and write down the number 5000.365 in the formats for four different locales. First, English (U.S.), then English (United Kingdom), then Chinese (Taiwan), then one of your choice. Be sure to hit the ready button after selecting a locale. You may get a question-mark instead of a currency symbol for some locales; that is because your computer software may not know how to display that particular symbol.



Understanding Locales

A locale consists (essentially) of a language and a country. Class `Locale`, in package `java.util`, contains information about over 100 locales (but not all possible locales in the world). The English (U.S.) locale is the default in the U.S., but you can probably set the default to what you want on your computer.

To display the default locale of your computer in `String` field 3 of the GUI, Add the following to method `actionPerformed`, just before the `return` statement:

```
setStringField(3, "Default locale: " + Locale.getDefault());
```

Note that method `getDefault` is a `static` method of class `Locale`, so it can be called without creating an instance of class `Locale`. Method `getDefault` does exactly what it sounds like; it gets the default locale for this computer. See the **Java API** for more information on class `Locale`. ¡P¿

Now when you run the program, the displayable form of the default locale for your computer is placed in String field 3, showing you the language and country, each in two letters. For example, if the default locale for your computer was French (Canada), field 3 would contain:

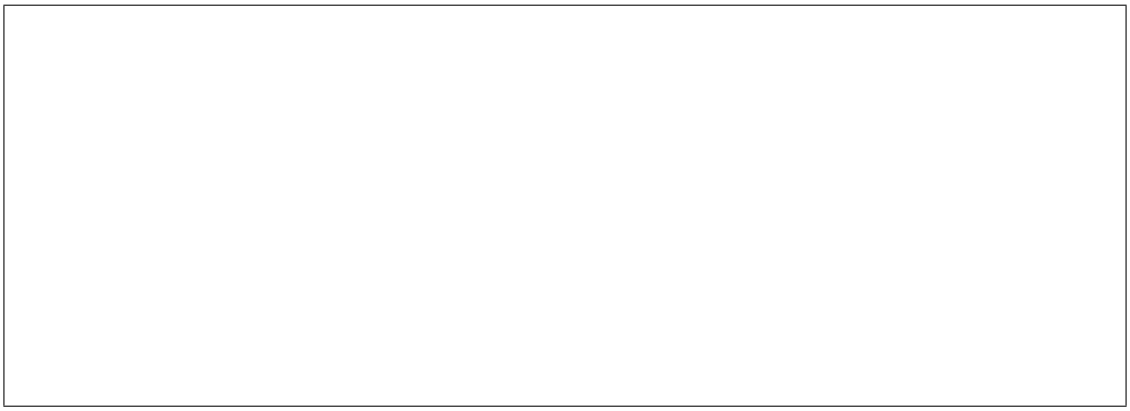
Default locale: fr_CA

Now add code to print the displayable form of the locale that you selected in the GUI by adding the following statement to method `actionPerformed`, just before the `return` statement:

```
setStringField(4, "Selected locale: " + view.getLocale());
```

Note that method `getLocale` is in the view class `GenericGUI`. The view is where the user chooses the locale, and so the controller must get that information from the view.

Once you have done this, write the display form of three different locales below. Show what was in the locale pulldown box as well as the language and country codes.



Decimal Number Format

Now suppose you want to format some decimal numbers using the format of `Locale` whose name is stored in a variable `loc`. To do this, you first need an instance of the class that does such formatting, which you obtain using the following:

```
NumberFormat f = NumberFormat.getInstance(getLocale());
```

You then call a method to format the number that is in `double` field 0:

```
String s= f.format(getDoubleField(0));
```

When done, you can put the result in String field 5:

```
setStringField(5, s);
```

Place these three statements just before the return statement in method `actionPerformed`; do not delete any of the statements written so far. Then execute the program. You will see that the values in String field 0 and 5 are the same, because we use the same technique to convert the number into decimal format.

Place the following numbers in `double` field 0, hitting the ready button (with locale US selected, please) each time, and write the resulting numbers that appear in String field 5:

Field 0	**Field 5**
1E6	
1E-6	
3.1459467	
3.1459467E3	
3.1459467E15	

Given this results, give an English description of the standard US format for decimal numbers.

Currency Format and Percent Format

Above, you used `NumberFormat.getInstance(getLocale());` to get an instance of a class that would help you format numbers in the decimal format of the selected locale. To get an instance of a class to format numbers in a currency, use the statement:

```
f= NumberFormat.getCurrencyInstance(view.getLocale());
```

Similary, for a percentage, use

```
f= NumberFormat.getPercentInstance(view.getLocale());
```

Try them and repeat the exercise from the previous step.

Field 0	Currency	Percentage
12		
1E3		
1E-3		
3.1459		
3.1459467E3		

Getting the Available Locales

Not all possible locales are implemented by Java, but over 100 are. Take a look at the declaration of instance variable `locales` in class `GenericGUI`. Its righthand side is a call on a `static` method in class `Calendar`, which returns an `array` of all locales that have been implemented. Variable `locales` is used later in method `createChoiceBox` to provide a list of all available locales.

It might be useful to have a listing of what the decimal, currency, or percent format of a particular number would look like for all locales at once. Method `printInAll` in class `LocaleController` will print a list of all possible decimal formats of a `double` variable, when `int` parameter `j` is greater than zero. Write a call to this method inside method `actionPerformed`, so that it will print a list of all possible currency formats for the value entered in `double` field 0.

The output will appear in the Java console, because of the use of the `System.out.println` statements. Write down the first six lines of the printout below