# CS-207: Programming II

# Northeastern Illinois University

# Research Lab: Forms & Strings

# Due: [insert date]

## Goal:

The goal of this research lab is to modify the given HTML and Java files to practice and investigate how to validate information in String formats submitted through an HTML web form and through a JavaFX application and to compare and contrast the difference between web application and server-side applications. As this is a "research" lab, you will need to analyze the code that has been provided for you in order to be able to modify it using the materials presented in class.

## The Problem:

You have been provided with an HTML file named simple\_form.html, and SweepstakesJavaFX.java, which contains the code needed to display a simple sweepstakes entry form. You will need to add the necessary text validators to the HTML code to create a functioning form with validation. Once the HTML code is complete, you will need to transfer this validation to the JavaFX form.

## Instructions:

* You should work in groups of 2-3 individuals. Groups of more than 3 are not permitted.
* Each group should submit ONE lab write-up. It is the responsibility of each group member to ensure that their name is on the write-up.
* The lab write-up should be typed! Type each question (and the question number) followed by your group’s answer. Convert your lab write-up to a .pdf.
* You should use complete sentences and proper grammar in your write-up. Use spell-check! This counts as part of your grade.
* Do not copy/paste directly from your sources for your answers (this is called plagiarism). Instead, you should re-word the information in your own words.
* Submit the PDF and Java files in a ZIP file to D2L by the specified due date.
* Required: Submit a group selfie! This should not be taken in our Programming II classroom or in a hallway, but should either be a group selfie via Google hangouts (See this for an example: http://i.imgur.com/WLLxEEO.png) or a group selfie of all of you sitting in a common area and working. Your entire group must be in the picture and you should submit it as a .pdf, .jpg, or .png file.
* Each member of the group must turn in a readable digital copy of the peer assessment to an individual Dropbox by the assigned due date and time. The peer assessment counts as a significant part of your grade and you will receive a zero for that portion of the research lab grade if you do not turn it in.

## Part A: Getting Started

Download the project files from D2L. Once again, you are being presented with code you did not write! You need to finish the HTML form so that it includes data validation for each field.

Double-click to view simple\_form.html with the browser of your choice.

## Question #A.1

Now, open the HTML file in a text editor program (such as Notepad, TextEdit, or Notepad++) and examine the code. You can see HTML tag elements, which are enclosed by angle brackets, < and >. The first word inside the brackets indicates the type of tag. Examine the code and list 5 different types of tags (do not choose the “html” or “input” tags).

## Question #A.2

Often HTML attributes follow after the tag – the number of attributes can vary, depending on the type of tag. Attributes typically have the format attributename=“attributevalue” or attributename. For each input tag, examine the type attribute– list 3 different types that you find.

## Question #A.3

Return to your web browser. Test the code by filling out the form with obviously invalid input data. Submit your form – do you receive an error message? If so, what does it say?

## Question #A.4 + coding

This form has no validation yet, and can be submitted without typing in any values! You can add the required attribute to an input tag, which means that the user will have to enter something into that field before the form can be submitted. Go back to your text editor and add this attribute to the end of every input tag – just before the >.

Save your code, and then refresh the page in your web browser. Test your code by leaving every field blank and trying to submit. Did you receive an error message? If so, what error message(s) did you receive? Do you receive one error message, or multiple error messages?

## Section B – Validating Input

## Question #B.1

Now each field is required, but you can enter any information into the fields – you need to add data validation to each input field, so that we know the submitted information will be correct. The validation rules for each field are listed below.

If you add the pattern=“” attribute to your input tags, you can validate using regular expressions, which you learned about in class! You will put your regular expressions between the quotation marks, just like you do in Java. **(Note: In HTML there are other attributes we can use to validate fields, such as selecting type=“tel” for a phone number, type=”date” for a date, or “max=” and “min=” attributes for numbers/dates. Only use the pattern attribute for this research lab!)**

**Validation requirements**

Every field is required

**First Name** and **Last Name**: at least 2 characters, and only contains uppercase and lowercase letters

**Phone Number**: Format is ###-###-####, where #’s can only be digits

**Email Address**: Format is xxx@xxx.yyy, where xxx and yyy stand in for any number of characters (at least one). yyy’s can be any letter or digit. xxx’s can be any letter, digit, or special characters: . \_ or -

**Lucky Number**: Can be any number between 1 and 100.

**Date of Birth**: Format is mm/dd/yyyy or m/d/yyyy, where m, d, and y’s can only be digits. The first two values for yy must be either 19 or 20.

## Question #B.2:

Draft the regular expressions you will use for each field:

First Name

Last Name

Phone Number

Email Address

Lucky Number

Date of Birth

Write these out by hand, and list them in your research report.

## Question #B.3: coding only

Test out each regular expression, one at a time – this is very important, and will make it easier to troubleshoot! Follow this process for each input tag:

Implement one regular expression attribute. Save your work and refresh the page. Test by inputting both valid and invalid data.

## Part C: Creating the Java Validation

Great work, you were able to create a functioning HTML form! Now you need to create this same form in Java. You have been given the JavaFX code for the layout; you will just need to write the code that does the validation.

## Question #C.1 – reading only

JavaFX is a set of packages which creates applications and event-driven programs. Applications are laid out differently than the types of programs we have been writing so far. Look at the main method – there is only one line, which calls the Application.launch method. This method then calls the start method, which is directly below main. Read through the beginning of this method (lines 28-65), to get a general sense of what is happening in this section of code.

There are no attributes in Java to accomplish data validation, like in HTML. Instead, you need to create validation methods, which will utilize the String methods.

## Question #C.2

As there can be multiple errors, you will need to accumulate your error messages. In other words, if there is more than one error, you should see ALL applicable error messages. In this program, what type of object is error? Look up this class in the JavaFX 8 documentation. What method can retrieve the current value stored in error? What method allows you to update the value in error?

## Question #C.3 - coding only

At the top of the handle method, you will see six Strings which correspond to the six form fields. Each validation method takes a String parameter and does not return anything.

Create the first method, checkFirst, which validates the first name. If the String parameter is empty, the method should update the text of the label named error to read "First name required". Otherwise, if the first name is invalid, the method should update error to read "First name invalid".

Inside the handle method, call this method on fName. Compile and run your code. Test and fix any errors.

## Question #C.4 - coding only

Create the method checkLast, which validates the last name. If the String parameter is empty, the method should update the text of the label named error to read "Last name required". Otherwise, if the last name is invalid, the method should update error to read "Last name invalid".

In the handle method, call this method on lName. Compile and run your code. Test and fix any errors.

## Question #C.5 - coding only

Create the method checkPhone, which validates the phone number. If the String parameter is empty, the method should update the text of the label named error to read "Phone number required". Otherwise, if the phone number name is invalid, the method should update error to read "Phone number format: ###-###-#### ". Inside the handle method, call this method on inPhone. Compile and run your code. Test and fix any errors.

## Question #C.6 - coding only

Create the method checkEmail, which validates the email address. If the String parameter is empty, the method should update the text of the label named error to read "Email address required". Otherwise, if the email address is invalid, the method should update error to read "Email address invalid". Inside the handle method, call this method on inEmail. Compile and run your code. Test and fix any errors.

## Question #D.7 - coding only

Create the method checkLuckyNum, which validates the lucky number. If the String parameter is empty, the method should update the text of the label named error to read "Lucky number required". Otherwise, if the lucky number is invalid, the method should update error to read "Lucky number must be between 1 and 100". Inside the handle method, call this method on inLuckyNum. Compile and run your code. Test and fix any errors.

## Question #C.8 - coding only

Create the method checkDob, which validates the date of birth. If the String parameter is empty, the method should update the text of the label named error to read "Date of Birth required ". Otherwise, if the date of birth is invalid, the method should update error to read "Date of Birth format: ##/##/#### or #/#/#### ". Inside the handle method, call this method on inLuckyNum. Compile and run your code. Test and fix any errors.

## Part D: Final Summary

Whenever you complete a project, it is important to assess what you think went well and what you need to improve on.

## Question #D.1

What was the most challenging part of this research lab for your group?

## Question #D.2

What did your group learn/find the most useful by doing this research lab?

## Question #D.3

Did your group find validation with the HTML web form easier to implement or validation in the JavaFX file easier to implement? Why?

## Question #D.4

What topics covered in your Programming II class (either in lectures or textbook reading) did you use in this lab?

## Question #D.5

What was the most fun aspect of doing this research lab?