W. P. CAREY
SCHOOL of BUSINESS
ARIZONA STATE UNIVERSITY

CIS 494 – Business Systems Development with Java – Spring 2015

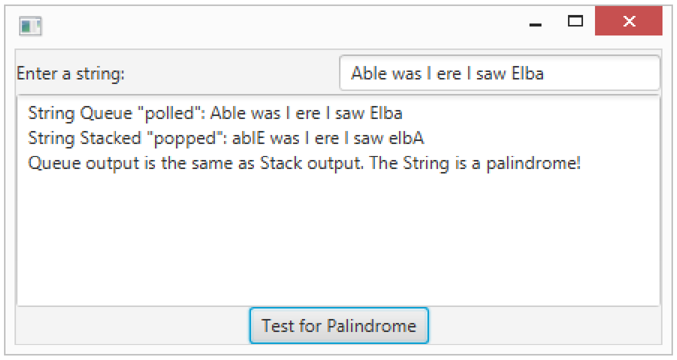
Assignment 4

Due on Blackboard: Friday, March 27 by 11:59 PM

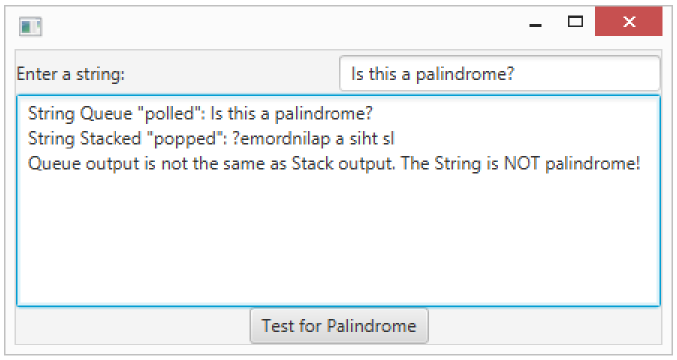
Skills Developed: Using Scene Builder, JavaFX, Queues, and Stacks. Utilizing the basic MVC framework of creating Controllers to process business logic.

Create a simple JavaFX application that checks if a specified string is a palindrome or not. A palindrome is a word or phrase that is the same when spelled in reverse, e.g. mom, wow, "A man, a plan, a canal – Panama!"

Sample Output

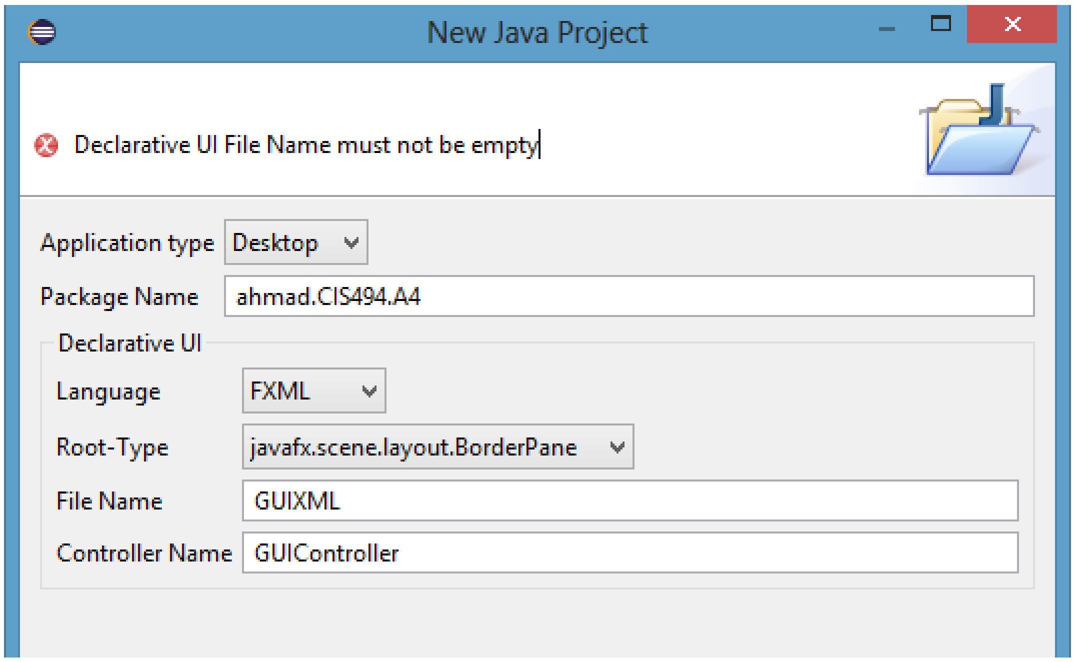


```
String Queue "polled": Able was I ere I saw Elba
String Stacked "popped": ablE was I ere I saw elbA
Queue output is the same as Stack output. The String is a palindrome!
```



```
String Queue "polled": Is this a palindrome?
String Stacked "popped": ?emordnilap a siht sI
Queue output is not the same as Stack output. The String is NOT palindrome!
```

1. Develop a JavaFX project. Utilize e(fx)clipse to generate a FXML document for the user interface. You may automate all generation of the GUI elements as well as the entire FXML document by specifying the FXML details directly to e(fx)clipse (just keep on pressing "Next" when you create the project to get to the options shown below).



New Java Project

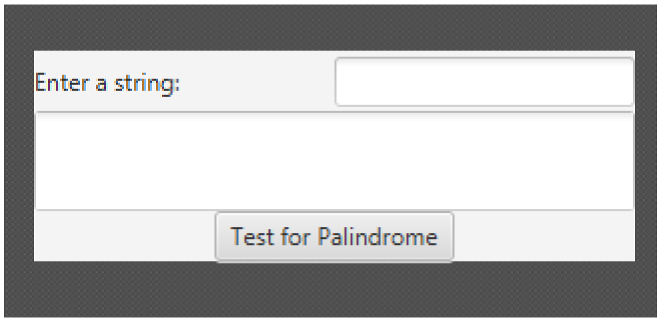
Declarative UI File Name must not be empty

Application type: Desktop
Package Name: ahmad.CIS494.A4

Declarative UI
Language: FXML
Root-Type: javafx.scene.layout.BorderPane
File Name: GUIXML
Controller Name: GUIController

On the other hand, you may do the FXML steps manually as shown in class if you choose to do so!

2. Setup the GUI so it roughly matches the sample. You should have an fxid for the TextField, TextArea and the Button. The Button should have an event handler that is specified within SceneBuilder (or directly into the FXML document). Make sure the Controller is specified properly in the document properties.



3. Make sure your Controller java file has appropriate references properly prefixed with @FXML for all elements that you want linked between the java file and the FXML document.

4. Within your controller class, create an initialize method that returns void. The initialize method should set the TextArea to not be editable by calling an appropriate method from the TextArea class.

5. In your event handler for the button,

Create 3 strings – one for the storing the content of the TextField (userString), one for storing the ouput of a Queue (queueString) and one for storing the Output of a Stack (stackString).

Store the text from the TextField in the first one and instantiate the other two strings using the "no-args" String constructor. Trim the userString of spaces!

Process the Queue

Declare and instantiate a Queue of datatype Character. Loop through the entire userString (use its length property) and add/offer each character of the string to the Queue.

Go through the entire queue and remove/poll each Character in the queue and add it to the queueString. This should be a loop and after the loop is completed, queueString should have in it a string, which is the output of the queue. *This means that all the characters put into the queue must come out in the same order. queueString should have in it at this point the same string as userString!*

Add the word as it came out of the Queue to the TextArea (see sample picture).

5. (Event handler method continued…)

Process the Stack

Declare and instantiate a Stack of datatype Character. Loop through the entire userString (use its length property) and add/push each character of the string to the Stack.

Go through the entire Stack and remove/pop each Character in the Stack and add it to the stackString. This should be a loop and after the loop is completed, stackString should have in it a string, which is the output of the Stack. *This means that all the characters put into the Stack must come out in the REVERSE order. stackString should have in it at this point the same string as userString but in REVERSE order.*

Add the word as it came out of the Stack to the TextArea (see sample picture).

Compare the stackString with the queueString (case insensitive). If they are the same, the word is a palindrome. If they are not the same, the word is not a palindrome. Print an appropriate message to the TextArea.

Optional Enhancement

Account for spaces and punctuation by ignoring only spaces and punctuation while comparing (or adding to Stacks or Queues).

When done properly, the following should be identified as proper palindromes:

> A Man, a Plan, a Canal – Panama!
>
> Was it a car or a cat I saw?
>
> A Toyota's a Toyota!

<u>Submitting Files</u>

Submission should be made using a zip file that contains the entire Eclipse project folder. You will need to ***zip the entire project folder***. The folder will automatically contain the class source files as well as the compiled .class files.

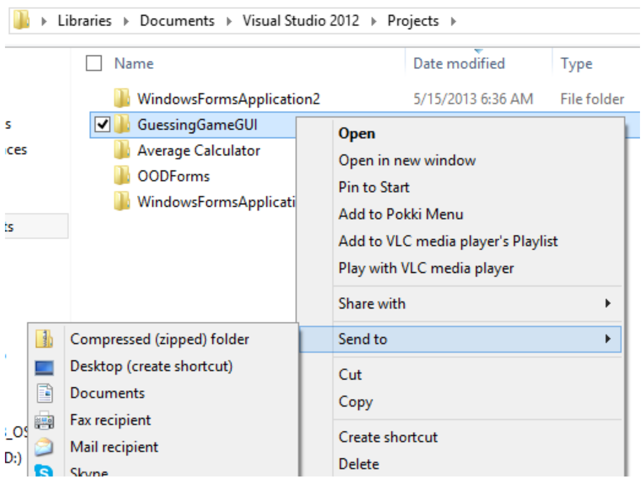
ZIP file should be named: **AX.zip or IC.zip**

> where X is the in-class assignment number, e.g., A1.zip is the submission file for Assignment 1 and IC1.zip is the submission file for InClass1.

Note: *The ZIP filename is <u>independent</u> of the Project name. Do not name your project A1.*

How to Properly ZIP and Submit Your Project Files:

- Go to the folder within {Eclipse Workspace}\
- ZIP the entire top-level folder for your project by right-clicking your project folder and selecting Send to | Compressed (zipped) folder.
- Finally, submit the ZIP file using the submission link on Blackboard by the due-date and time listed on the assignment. Upload the ZIP file.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Verify your files BEFORE and AFTER submission:

- Check for actual class files being present in the folder before you zip it.
- ***Ensure that you are not zipping a short-cut to the folder.***
- After zipping, check file size. A file size under 4K likely does not contain all the files.
- Unzip, extract all files, and verify you see actual files, not a solitary short-cut.
- Uncompress your zip file before submitting and verify that files are present.
- Make sure you have ***submitted*** your file and not just saved a draft on Blackboard. ***A blue clock indicates a submission in progress, i.e. a draft, not a submission. The draft is accessible only to you. You will get a ZERO if you only ever save a draft on Blackboard and never submit your files.***

- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.