CIS 494 – Business Systems Development with Java – Spring 2015

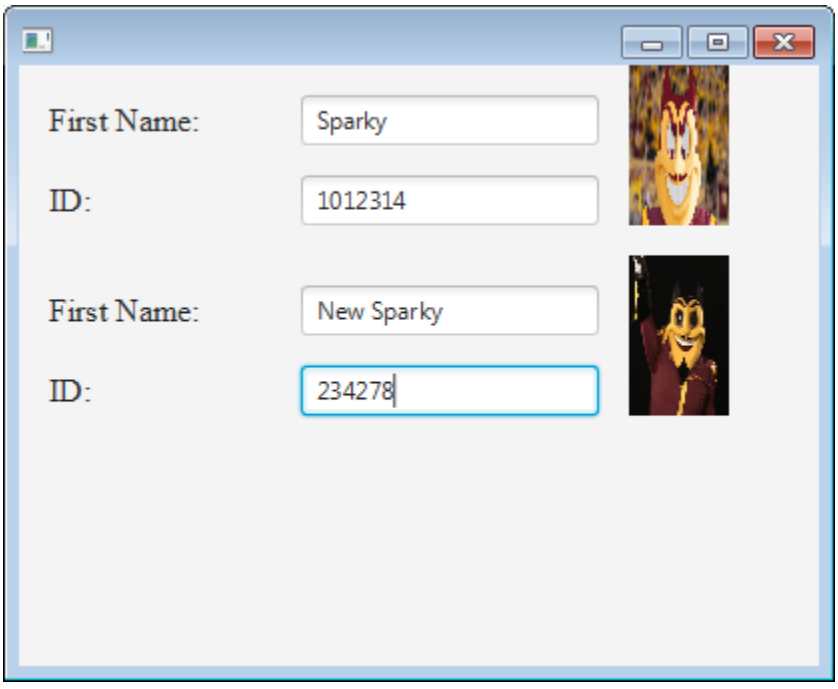
Assignment 3

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

Skills Developed: Implementing JavaFX, developing logic to create repetitive graphical elements, reading from text files, and developing GUI elements dynamically at runtime.

Implement a JavaFX application, which reads student data from a text file and for each line of data in the file, creates a "record" in the GUI. Each graphical record consists of Labels and TextFields for name and ID. Each record also has a photo displayed based off a URL from the text file.

Sample Output



- Create a new Eclipse project for Assignment 3.
- Download the Students.txt file to your Eclipse project folder.

  This file should have data lines in the following format:

  Name,ID,URL

  Each line represents one student record, which needs to be converted into a series of graphical elements in JavaFX.

Method Descriptions

Note: There are detailed method descriptions given here. However, if you choose do so, you may create your own method logic. Any methods you write should be properly modularized and reusable within the context of *this* application.

If you use the logic provided here, please note that you do need to develop your skills to the point where you are able to create logic like this on your own!

1)
Method Name: buildUserLine
Purpose: This method should build one "line" or record for a given user. The line consists of labels, textfields, and an image. It should then return the line, which is a HBox object.
Parameters: 3 Strings, name, id, and url.
Return Type: HBox

Logic:

- Declare and create a new instance of GridPane called recordPane.
- For recordPane
  - set a Padding with an Inset of 15 on all sides.
  - Set an Hgap of 50 pisels and a Vgap of 15 pixels.
- Declare and create a new Font called standardFont and specify that you want the Times New Roman font, sized 16 points.
- Declare and create a new Label called nameLabel and make it display "Name" using the constructor.
  - Set the Font of the nameLabel to standardFont.
- Declare and create a new TextField called nameTextField.
  - Set the text of the TextField to show the parameter *name*.

- Declare and create a new Label called idLabel and make it display “ID” using the constructor
    - Set the font of the idLabel to standardFont.
- Declare and create a new TextField called idTextField.
    - Set the text of the TextField to show the parameter *id*.
- Declare and create a new Image called image. Pass the url to the Image constructor.
- Declare and create a new ImageView called imageView. Pass the image to ImageView constructor.
    - Using the setFitHeight and setFitWidth methods, set the height and width to 80 * 50 pixels.
- Add the nameLabel, nameTextField, idLabel, and idTextField to the ***recordPane***. They should appear in a grid using 2 columns and 2 rows. Specify column first!
- Declare and create a new HBox called ***outputPane***.
    - Using getChildre(), add the recordPane and imageview to outputPane.
- Return the finished outputPane. *This has now create ONE record pane per line of user info.*

2)
Method Name: processLine
Purpose: This method should accept one line of input as a String, which represents the data of one student. This line needs to be parsed and one student record needs to be created from it. The student record will be in the form of a graphic HBox object.
Parameters: One String called line.
Return Type: HBox.

Logic:

- Declare an array of Strings called splitLine. Call the split method of the *line*, which you received as a parameter. Pass it a comma as an argument. This should split the string into an array delimited by commas.
- Declare String variables for name, id, and url.
- The splitLine array has 3 elements in it from index 0 to 2. Assign one element each to name, id, and url.
- Declare an HBox called linePane. Call buildUserLine and pass to it your name, id, and url. Assign the return value of buildUserLine method to linePane.
- Return linePane.

3)
Method Name: loadStudents
Purpose: This method will do File I/O and read the data file that has the details of each student in it. For each line representing one student, it will call the processLine method. It will then add the panes for each line to the overall VBox pane to develop a vertical list of students. The method then returns the VBox.
Parameters: None
Return Type: VBox

Logic:

- Declare a String called inputLine.
- Declare and create a VBox called pane.
- Start a try-with-resources block.
    - Within the parantheses, declare and create a Scanner object called inputFile. The argument for the Scanner constructor ought to be an instance of a File. Call the constructor of the File object using the new keyword and supply the File constructor with "Students.txt" as the name of the file.
- Within the try block,
    - Use a while loop. **While,** inputFile has a new line to read (use hasNext()),
        - Call the nextLine method of the Scanner inputFile and assign it to inputLine.
        - Declare a new HBox called tmpHBox. Call processLine and pass it inputLine as a parameter. processLine returns an Hbox – assign its return value to tmpHBox.
        - Add the tmpHBox to pane using the getChildren() method.
- Catch two exceptions: FileNotFoundException and NoSuchElementException.
- After the try-catch has ended, return pane.

4)
Method Name: start
Purpose: The start method is called automatically and is the “start” of the JavaFX Application.
Parameters: One parameter of type Stage called primaryStage.
Return Type: void

Logic:

- Declare a new VBox called pane. Call the method loadStudents and assign its return value to pane.
- Declare and create a new Scene called scene. Pass to its constructor, the pane, and the size 400 * 300.
- Set the scene of primaryStage to scene.
- Show the stage.

5)
Method Name: main
Logic:

- Call the launch method of Application class. Pass it the args parameter as an agument.

Submitting Files

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.

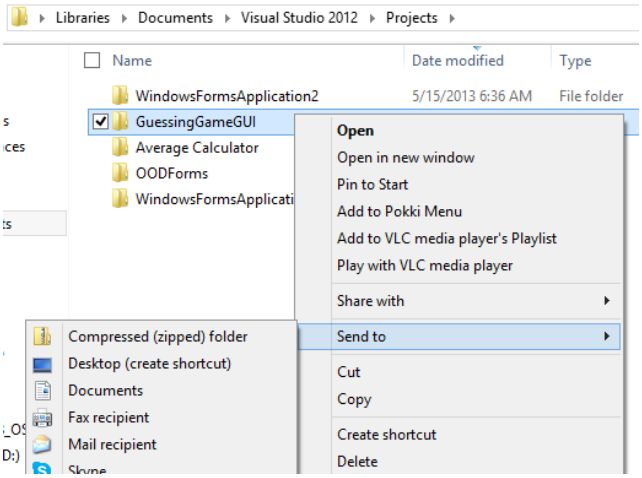
ZIP file should be named: **AX.zip or IC.zip**

> 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 independent 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.