The City College of New York
Department of Compute Science
CSc 221: Software Design Laboratory

Assignment 5 – Fall 2019

In this assignment we employee the Java techniques we have learned so far along with JavaFx and object serialization. This assignment counts towards 12% of your final grade.

Note: please do your own work, sharing and/or copying code and/or solution ideas with/from others will result in a grade of 0 and disciplinary actions for all involved parties. If you run into any problems and have done your best to solve them, please see me before/after class or e-mail me.

Problem Description:

For this assignment, you are being asked to finish the development of a small application for a saving a company's contacts using JavaFX and object serialization. The application's GUI is missing some important controls; please add them using your favorite JavaFX Scene Builder. The complete application's window is shown in Figure 1. Your solution will be graded for usability and error checking, so make sure your application shows and saves the correct and up-to-date information without any errors

Note: error message can be shown using the Alert class. The following statements display an error popup message:

$Alert\ alert\ =\ new\ Alert(AlertType.ERROR);$
$alert.setTitle("Invalid\ value");$
$alert.setHeaderText("Error\ message\ here");$
$alert.showAndWait();$

- **Initial state:**
  - When the application starts all controls are disabled except Load and Exit
  - Other controls are enabled based on the current state of the window. For example, once the load button is clicked, the $add(\ )$ button is enabled. When the button is clicked, $textfileds$ and $delete$ button are enabled …
- **Title:** *make the title of your window* "$Assignment\ 5 - <\ your\ name\ >$ "
- **Validation:** *You must use Regular Expression for validation*
  - **Name:** *Valid names consist of one or two words. Each word must start with an uppercase letter followed by at least two characters. Numbers are not allowed.*
  - **Company:** Valid d*epartment names consist of one or two words. Each word must start with an uppercase letter. A word can be a single uppercase letter. Numbers are allowed*
  - **Extension:** Valid e*xtensions start with 1, 2, or 3 numbers followed by a dash* " $-$ " *followed by 1 or 2 numbers*
- **Load:** use Java's $FileChooser$ class to prompt the user to select a file. Once the user selects a file, open the file and load the list. If the serialization fails, show an error message and create an empty list. Show the file's name next to this button. For the first time, you can create an empty XML file and select it via $FileChooser$.

  Note: just for this assignment, serialization will be written to this file whether deserialization succeeds or fails

- **Add button (+):** enabled once the file is loaded. It allows a new record to be appended to the list.
  - ✓ Don't allow a new record to be added unless the current record values are valid.
  - ✓ Before adding a new record, save the current one.
- **Remove button (–):** enabled if there is at least one record displayed. The button removes the current shown record.
  - ✓ Update the text fields with proper record. For example, if record #5 is removed, the application displays #4.
- **Next button (≫):** enabled if there is a record that can be displayed after the current one.
  - ✓ Once clicked, the successive record is shown.
  - ✓ Save the current record before moving on
- **Previous button (≪):** enabled if there is a record that can be displayed before the current one.
  - ✓ Once clicked, the previous record is shown.
  - ✓ Save the current record before moving on

- **Serialize:** the list will be serialized to a file using XML format. Use the $FileChooser$ to prompt for the file.
  - ✓ Save the current record before moving on
  - ✓ Once saved, the file's name is displayed next to the $Load$ button
- $\boldsymbol{Exit}$: exits the application.
  - ✓ The button will confirm that the user wishes to exit. If the user selects OK, exit the application
- **Current record label:** a label that shows the current record being displayed

Provided Files:

1. **Employee.java:** will contain the employee information displayed in the GUI's fields
2. **EmployeeList.java:** a container class to manage the different Employee instances. This class can be serialized (and deserialized) using $JAXB.marshal()$
   Hint: Create an instance of type $EmployeeList$ in your controller
3. **Directory.fxml**: an incomplete FXML GUI. The file assumes that the controller is called $DirectoryController.java$

Grades:

Part of the grades for each class will be dedicated for proper logic. For example, you should reuse code from the parent class when implementing certain methods. You should also implement methods in the right place to avoid duplicate code statements.

Item	Points
GUI:	
Title	4
Adding missing controls	10
Initially disable all fields except Load and Exit	5
Field validation with Regular Expressions including alert boxes	26
Load:	
$FileChooser$	5
$Deserialize$ list	5
Add Button:	
Validation	2
Reset fields	3
Remove Button:	
Validation	2
Update fields	3
Next/Prev Button:	
Validation	4
Update fields	6
Save fields	6
Current Record label	8
Serialize Button	
Validation	2
Serialize	3
Exit	6
	100

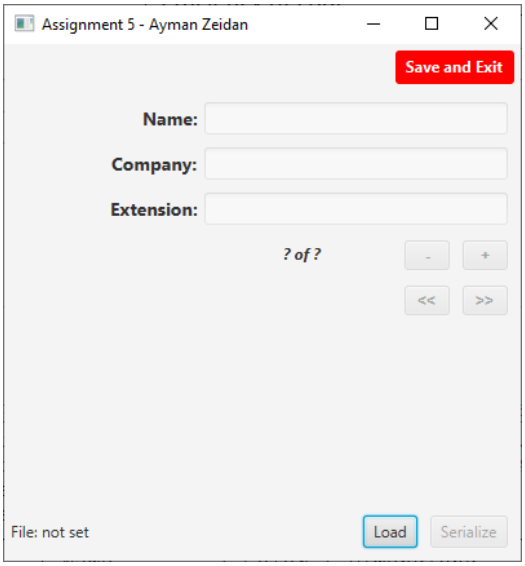

Figure 1: Initial Start

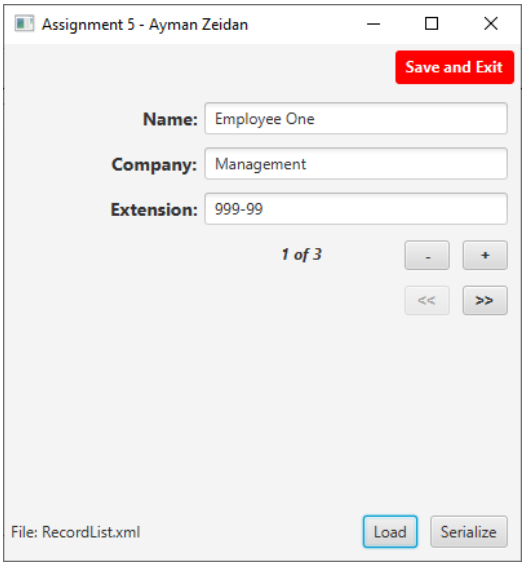

Figure 2: File Loaded

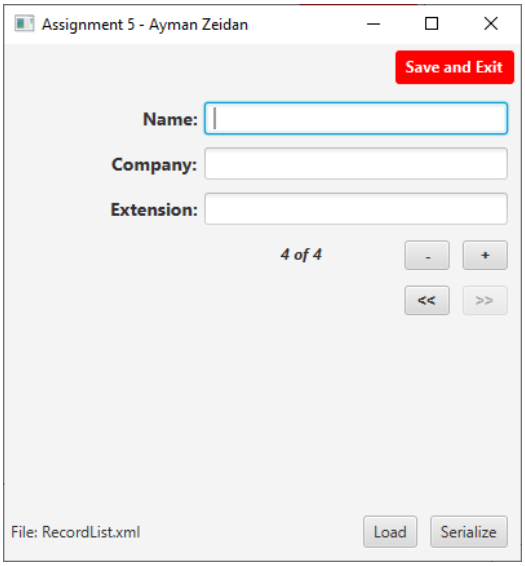

Figure 3: Add button clicked

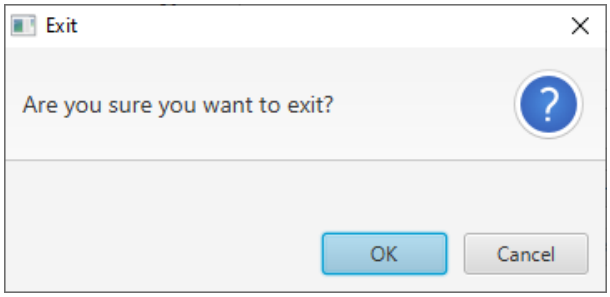

Figure 4: Exit confirmation message

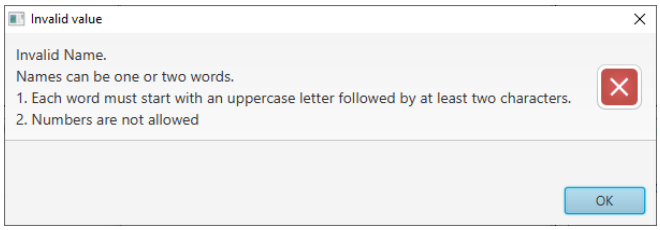

Figure 5: Name validation error message

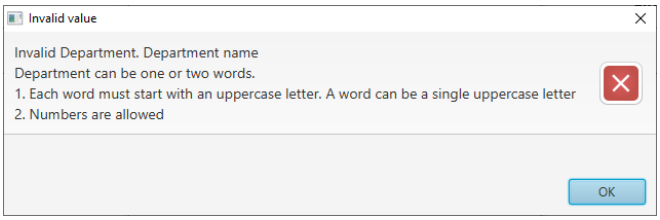

Figure 6: Department validation error message

Figure 7: Extension validation error message