
Java Fundamentals End Assignment

Hogeschool Inholland
Teachers: Mark de Haan & Wim Wiltenburg
Submission deadline: 25-10-2020 11:59 PM

Introduction

In order to complete the course of Java Fundamentals students need to complete two tasks:

1. An end assignment
2. A classroom exam session, extending the functionality of the end assignment with three new features.

Before you is the document detailing the requirements for this end assignment.

General requirements:

- The application must be written in Java 11 and JavaFX
- SceneBuilder and FXML are NOT to be used
- Libraries like Apache Commons are forbidden.

Assignment submission requirements

- The application must be submitted as a zip file with only source code.
- The name of your zip file must be: [your-name]-[your-student-number]-[end-assignment].zip, e.g. **wim-wiltenburg-123456-end-assignment.zip**
- Before creating your zip file, do a code reformatting on all your files, and optimize your imports!
- Zip files with no source code will not be graded and the whole project will get no points – check your file before you submit!
- You don't need to add additional libraries. The JavaFX library is already part of the development environment of the reviewer.
- If you have specific usernames/passwords that are needed to log into the application, please share them in a README.md file in the root of the project.
- Your application must be able to start up in the IDE of the reviewer.

Coding standards:

Code should be executed by computers but read by humans. That means that you should write code in a clear and purposeful way that makes it easy for other people to understand, especially the teachers that are reviewing your code. Therefore, apart from the functionality of your application, you will be graded on the following:

Naming

Classes, fields and methods must be properly named, so that a reader understands exactly what the purpose of said class, field or method is.

Packages

The code must be structured into packages that are sensible and make it easy to navigate the code.

Object Orientation

The application must adhere to the Object Oriented paradigm. The DRY principle should apply as much as possible and use of inheritance is encouraged.

Encapsulation

Classes should not expose data without proper control. Make your fields private, with public getters and setters.

Responsibilities

Classes and methods should adhere as much as possible to the Single Responsibility Principle: a class or method should be concerned with one thing. Side effects should be kept to a minimum.

Short methods

Methods should not be too long. A rule of thumb is that a method should not exceed 30 lines. It's even better when methods are broken up into smaller methods. Robert C. Martin has written a very good book in which this is one of the principles: Clean code. It's suggested reading material.

Static

Static constants and methods are allowed, as long as they don't represent state. The Singleton pattern is not allowed as an alternative. State should be maintained by passing objects.

Edge cases

Make sure to think about ways that your code can fail, e.g. when you apply negative numbers or null where these are not expected, and handle these cases gracefully.

Formatting

Make sure you format your code in a way that makes it easy to read, both horizontally (indentation, lines that are too long), as vertically (logical ordering of your class members). A line is considered too long when it exceeds 80 characters. Before you submit your code make sure you format your code using your IDE. You can use IntelliJ standard (4 spaces) or Google Java Formatting. The latter is preferred. There's a plugin for that.

Comments

When writing code, you are free to insert comments, but it should be sensible. Strive towards your code being self-explanatory, e.g. by naming your variables in a way that makes sense. Comments should not be about what you are doing, but about explaining your decisions in solving a problem.

Profanity

Inserting profanity into your code will lead to demerit points. If the transgression is serious enough this could lead to obtaining no points at all. Be respectful to readers of your code.

Functional requirements

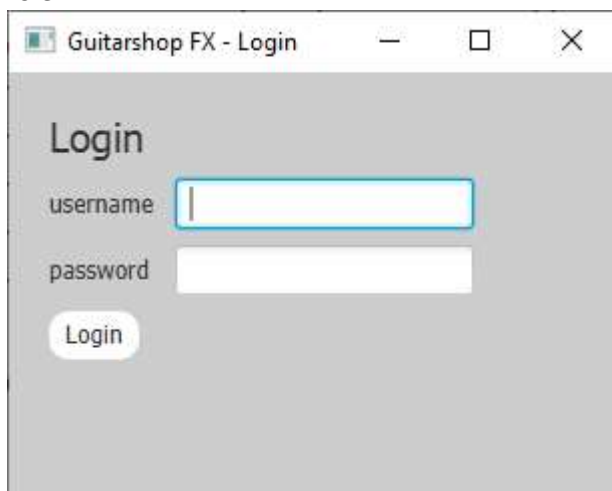
Your task is to build an application for a guitar shop. This application will have **two types of users**: a SALES person, and a MANAGER. Each type of user has access to a specific set of functionalities, which can overlap. You need to implement at minimum:

- A login screen
- A personalized menu according to type of user
- Different views:
 - A personalized dashboard
 - A screen for creating an order – accessible only to the sales person
 - A screen for seeing the order list – accessible to both sales and manager
 - A screen for stock maintenance – accessible to the manager
- Each view must have a window title detailing: The name of the shop (Guitarshop FX) and what the purpose of that view is, e.g. create order
- Each view (except the login screen or popup windows) will have the personalized menu at the top of the screen.

In the following and final section, the various parts of the application are explained in more detail. The screenshots in this screens are indicative. Feel free to arrange and design the GUI to your own taste, as long as the required functionality is there and the application is user-friendly.

Login Screen

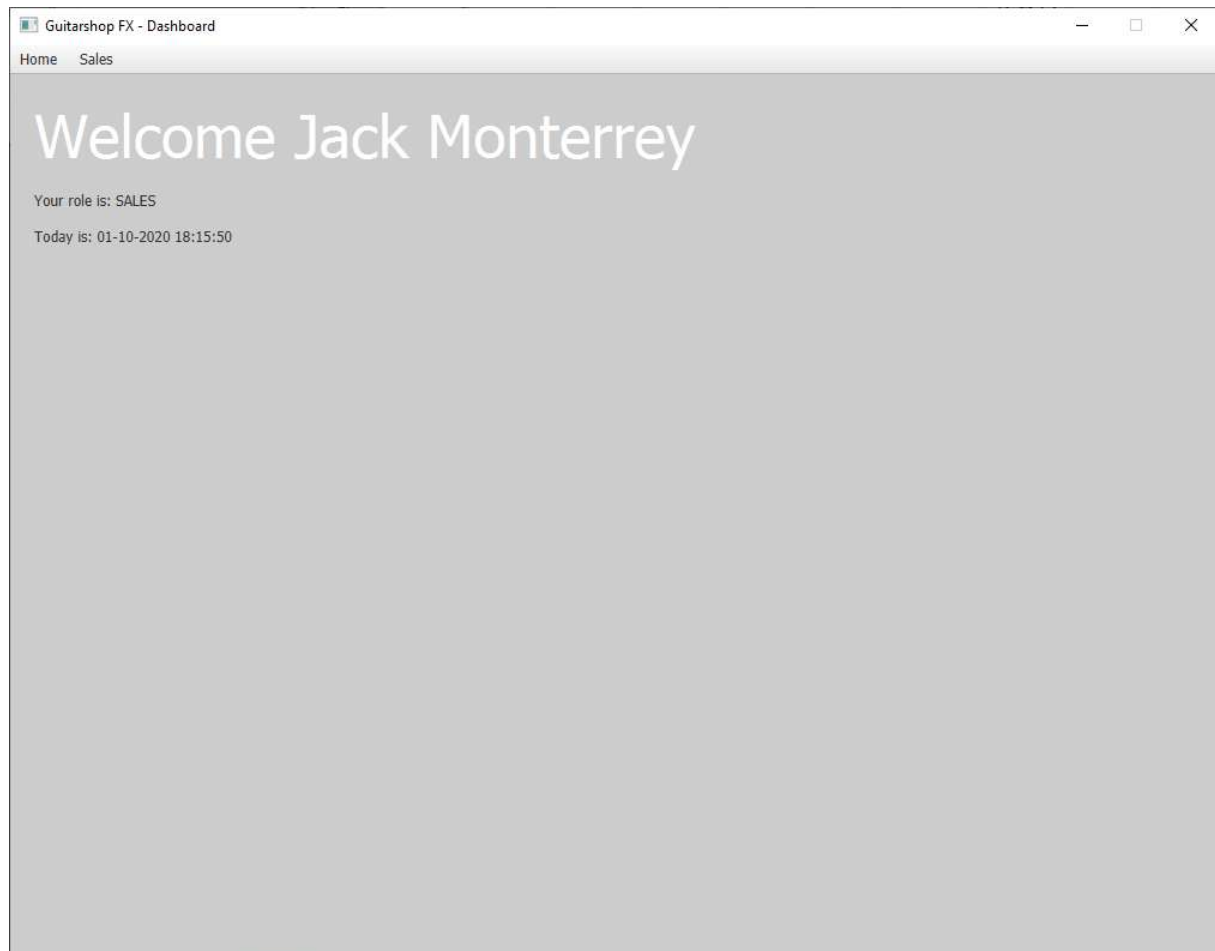
The login screen is a basic screen with username and password. You need to have a proper mechanism in place to validate the input and grant the user access to the application with the proper role.



The user must get a proper message if something goes wrong during the login process.

Dashboard

This dashboard welcomes the user, showing their name, the role they perform, and the current date and time:



On top of the screen is a menu. The following menus and menu items are available to the specific roles:

Role	Menu	Menu item
SALES	Home	-
SALES	Sales	Order
SALES	Sales	List Orders
MANAGER	Home	-
MANAGER	Sales	List Orders
MANAGER	Stock	Maintain

Order

The job of a SALES person is to create an order, and possibly to view the order list that's available.

For creating an order you need to implement a number of views:

The screenshot shows a web application window titled "Guitarshop FX - Create an Order". It has a navigation bar with "Home" and "Sales" links. The main heading is "Create Order #1000002". Below this, there is a "Customer" section with a search input field and a "Search" button. To the right of the search field is a form for customer details: "First name:", "Street address:", "Phone Number:", "Last name:", "City:", and "Email address:". Below the customer section is an "Articles" section with a table header: "Quantity", "Brand", "Model", "Acoustic", "Type", and "Price". The table body is empty, displaying "No content in table". At the bottom of the form are four buttons: "Add", "Delete", "Confirm", and "Reset".

- A search bar makes a pop-up window appear in which to select customers:

The screenshot shows a pop-up window titled "Guitarshop FX - Search customer". It has a heading "Customer List" and a table with the following data:

First Name	Last Name	StreetAddress	City	Phone #	Email
Wim	Wiltenburg	Stentorstraat 90	Amsterdam	06-123456789	wim@email.com
Jack	Traven	Dorpsstraat 10	Arnhem	06-87654321	jack@email.com
Jenny	Gump	Churchillallee 141	Den Haag	06-14253648	jenny@email.com

- The customer information is displayed on the order view

- When you click the Add button, a pop-up window appears in which you can select an article and specify how many of that article the customer wants:

Brand	Model	Acoustic	Type	Price	
Fender	Telecaster	false	REGULAR	1079.79	
Fender	Precision	false	BASS	1300.49	
Simon Patrick	Pro Flame Maple	true	REGULAR	1290.7	

11 Add Cancel

Not enough in stock for Simon Patrick Pro Flame Maple. Only 1 remaining

If the article is not in stock, or there are not enough in stock to complete the purchase, then an error should appear:

- When an article is added, the order view is updated:

Home Sales

Create Order #1000002

Customer

customer name Search

First name: Jack
Street address: Dorpsstraat 10
Phone Number: 06-87654321

Last name: Traven
City: Arnhem
Email address: jack@email.com

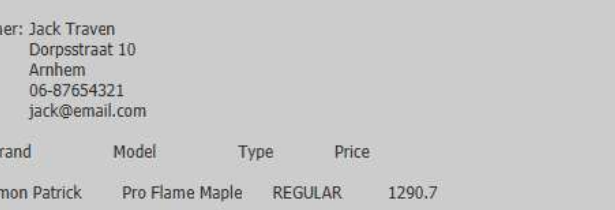
Articles

Quantity	Brand	Model	Acoustic	Type	Price
1	Simon Patrick	Pro Flame ...	true	REGULAR	1290.7

Add Delete Confirm Reset

- You can also delete an article from this list. Think about what this does to the amount in stock.

- When you have completed your order, you can confirm the order by clicking the confirm button. A new pop-up window should appear that displays the details of the order. When you confirm the order, the sale is made, and the order is added to the list of orders.



Customer: Jack Traven
Dorpsstraat 10
Arnhem
06-87654321
jack@email.com

Qty	Brand	Model	Type	Price
1	Simon Patrick	Pro Flame Maple	REGULAR	1290.7

Total price: 1290.7

Confirm

When you navigate away from the order view before the order is confirmed or you press the reset button, the order is lost.

List Orders

The view for listing orders is a Master/Detail view of orders. Orders are final and cannot be changed, so order editing functionality is not needed:

[illegible]

Stock Maintenance

The MANAGER is the only role that can do stock maintenance. In order to do this, a view is available with items and quantity in stock. In here the MANAGER can add or subtract a number from the quantity in stock.

NOTE: normally reducing the number happens only by creating orders. It should be possible to reduce the number in this window but that should take a conscious extra step for the user, not just make the number negative. An example is below, where a checkbox has to be checked to be able to reduce the amount in stock.

[illegible]