Application Assignment 2

Start Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a website url

Overview

Using IntelliJ and Gradle, you will create a GUI-based desktop application to allow a user to track their personal inventory.

The program should allow you to enter an item, a serial number, and estimated value. The program should then be able to display a tabular report of the data that looks like this:

```
Value     Serial Number   Name
$399.00   AXB124AXY3      Xbox One
$599.99   S40AZBDE47      Samsung TV
```

The program should also allow the user to export the data as either a tab-separated value (TSV) file, or as a HTML file. When exported as an HTML file, the data should be stored inside of a table structure to mimic the displayed appearance.

You will be responsible for both the design (UML diagrams) and implementation (production and test code) of this application

In addition to the assigned readings and videos, you will find the following links useful for this assignment:

- **https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm (https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm)**
- **https://openjfx.io/javadoc/16/ (https://openjfx.io/javadoc/16/)**
- **https://fxdocs.github.io/docs/html5/ (https://fxdocs.github.io/docs/html5/)**
- **http://tutorials.jenkov.com/javafx/index.html (http://tutorials.jenkov.com/javafx/index.html)**
- **https://www.youtube.com/playlist?list=PL4h6ypqTi3RR_bhBk6PtLfD83YkaJXXxw**

(https://www.youtube.com/playlist?list=PL4h6ypqTi3RR_bhBk6PtLfD83YkaJXXxw)

You will also find the following links useful if you need to hand-tune your UI:

- https://openjfx.io/javadoc/16/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html (https://openjfx.io/javadoc/16/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html)
- https://openjfx.io/javadoc/16/javafx.graphics/javafx/scene/doc-files/cssref.html (https://openjfx.io/javadoc/16/javafx.graphics/javafx/scene/doc-files/cssref.html)

Directions

Your application **shall** satisfy the following requirements:

1. The user shall interact with the application through a Graphical User Interface
2. The user shall be able to store at least 100 inventory items
    1. Each inventory item shall have a value representing its monetary value in US dollars
    2. Each inventory item shall have a unique serial number in the format of XXXXXXXXXX where X can be either a letter or digit
    3. Each inventory item shall have a name between 2 and 256 characters in length (inclusive)
3. The user shall be able to add a new inventory item
    1. The application shall display an error message if the user enters an existing serial number for the new item
4. The user shall be able to remove an existing inventory item
5. The user shall be able to edit the value of an existing inventory item
6. The user shall be able to edit the serial number of an existing inventory item
    1. The application shall prevent the user from duplicating the serial number
7. The user shall be able to edit the name of an existing inventory item
8. The user shall be able to sort the inventory items by value
9. The user shall be able to sort inventory items by serial number
0. The user shall be able to sort inventory items by name
1. The user shall be able to search for an inventory item by serial number
2. The user shall be able to search for an inventory item by name
3. The user shall be able to save their inventory items to a file
    1. The user shall be able to select the file format from among the following set of options:

1. The user shall be able to select the file format from among the following set of options: TSV (tab-separated value), HTML, JSON
    1. TSV files shall shall list one inventory item per line, separate each field within an inventory item using a tab character, and end with the extension .txt
    2. HTML files shall contain valid HTML and end with the extension .html
    3. JSON files shall contain valid JSON and end with the extension .json
2. The user shall provide the file name and file location of the file to save

4. The user shall be able to load inventory items from a file that was previously created by the application.
    1. The user shall provide the file name and file location of the file to load

You **shall** create a single IntelliJ project (using *Gradle*) and GitHub repository and ensure that your solution code has been pushed to the appropriate repository prior to the submission of this assignment.

- Your GitHub repository must follow the naming convention of **<lastname>-<last 4 numbers of your UCFID>-a5**.
    - As an example, I would create a GitHub repo named **hollander-1234-a5**.
    - Your github history will allow us to show the evolution of your solution. You should commit and push your code each time you complete a requirement (including successful testing).

You **shall** use a **.gitignore** file to ensure that your **.gradle** and **build** folders are not stored within git.

- Your repository should contain the following files and directories (with all expected subdirectories and files):

```
/.idea/
/uml/
/gradle/
/src/
/.gitignore
/build.gradle
/gradlew
/gradlew.bat
/settings.gradle
```

You **shall** place your code within a package named **ucf.assignments**.

- As an example, my main java file would be located in **/src/main/java/ucf/assignments/App.java** and the file itself would start with the statement **package ucf.assignments**;

You **shall** include the following comment at the top of *each* of your *.java files (with the appropriate name substituted in place):

```
/*
 *  UCF COP3330 Summer 2021 Assignment 5 Solution
 *  Copyright 2021 first_name last_name
 */
```

You **shall** create a Class Diagram for your application using PlantUML. This diagram must be pushed to your Git repository along with your java code.

- Each class within your diagram must list both the attributes and the methods defined within that class.
- You must store your *.puml files in a `uml` folder at the root level of your project (you will need to create this folder yourself).

You **shall** create your application Graphical User Interface (GUI) using FXML and store the associated *.fxml files within your project's `resources` folder.

You **shall** create a file named `readme.md` that contains a user guide for your application. This file must be placed within the root directory of your project.

- This guide should describe how to perform each operation specified in the requirements.

You **shall** follow good programming practices, as outlined in the course notes for Module 2. For example, use meaningful variable names, break your solution up into classes and functions when it makes sense, and keep your classes and functions short. You can find other guidelines

when it makes sense, and keep your classes and functions short. You can find other guidelines for writing "clean" code within the pages of Module 2 (see **04 - The Anatomy of a Class** and **05 - Class Design Guidelines**).

- You must decompose your solution into multiple classes, each of which will possess a single responsibility and will contain a series of highly cohesive methods (with each method having a single responsibility). You will receive no credit if your entire solution is placed within the **main** method.

You **shall** write your Java code in accordance with either the Oracle or Google Java coding standard: **https://www.oracle.com/java/technologies/javase/codeconventions-contents.html (https://www.oracle.com/java/technologies/javase/codeconventions-contents.html)** or **https://google.github.io/styleguide/javaguide.html (https://google.github.io/styleguide/javaguide.html)**

You **shall** use JUnit 5 to incorporate automated unit testing into your solutions.

- You must create a unit test for each behavior specified in the application requirements.
- You must include pseudocode in each test.

Your application and unit tests **shall** be completely implemented and run without errors on the grader's machine.

- This means that any paths should be relative to the working directory of your program.
  - Do not place user-oriented data files within your package structure.
  - Do not assume directories exist.

You **should** write detailed pseudocode for each of your methods prior to writing your test and production code.

Submission

You will submit a link to your GitHub page. Please ensure that your repository is private until 12 hours after the due date, at which point you may make it public. If you need myself or a TA to look at your code, you may share your repository with us - just ask us for our GitHub user names.

- It should go without saying that if we suspect you have copied code from another student, you will be referred to the office of student conduct and receive a Z grade for the course.

If we cannot access your repository, or if you provide an invalid link, you will receive a 0 - *please double check your submission once it has been made*.

Late submissions will receive a 0 as per the syllabus.

Grading Criteria

Points will be allocated according to the attached rubric. Full points will only be awarded if your implementation is: based on your design, satisfies the requirements, passes your test cases, and abides by the rules of clean code as outlined in the module notes. Elements that are entirely missing or woefully incomplete will receive no credit (for instance, only having a single class in your solution).

Scores may be reduced in the event of incomplete submissions or failure to follow directions.

Bonus Credit

The requirement to save/load an inventory list in JSON format is optional. Implementation of this requirement will yield an extra 2 points on this assignment.

Assignment 5					
Criteria	**Ratings**				**Pts**
Each requirement is implemented in the application.	**25 pts** **Full Marks**	**20 pts** **Most Marks**	**10 pts** **Partial Marks**	**0 pts** **No Marks**	25 pts

Each requirement is tested with a unit test.	**25 pts** **Full Marks**	**20 pts** **Most Marks**	**10 pts** **Partial Marks**	**0 pts** **No Marks**	25 pts
GitHub repository is correctly configured All expected directories exist. All files are in the correct location. Files were pushed to GitHub from a Git client and not drag and dropped into the browser via the Upload button.	**5 pts** **Full Marks**		**0 pts** **No Marks**		5 pts
GitHub activity shows multiple commits over a multi-day period. You must have pushed at least three separate commits of actual code over at least a 2 day period. Example: You push 2 code updates on Saturday and 1 code update on Sunday; or, 1 update on Monday, 1 update on Friday, and 1 update on Sunday.	**9 pts** **Full Marks**		**0 pts** **No Marks**		

				9 pts
UML Class diagram reflects the structure of the application.	**9 pts** **Full Marks**	**5 pts** **Partial Credit**	**0 pts** **No Marks**	9 pts
GUI design reflects good design principles and practices. Your GUI must present a layout that is easy to use and professional looking. It must be of sufficient quality that you would show it to a potential job recruiter.	**9 pts** **Full Marks**	**5 pts** **Partial Credit**	**0 pts** **No Marks**	

				9 pts
User guide is complete and well-written. The user guide must be free of spelling and grammatical errors, and should reflect a university education.	**9 pts** **Full Marks**	**5 pts** **Partial Credit**	**0 pts** **No Marks**	9 pts
Source code is well-written, easy to follow, and embraces clean-code principles.	**9 pts** **Full Marks**	**5 pts** **Partial Credit**	**0 pts** **No Marks**	9 pts
				Total Points: 100