System Guide and Maintenance

for

DTCC Toner Inventory System



Team Coffee House

4/9/2019

CS214

Delaware Technical Community College – DE

**Table of Contents**

**Introduction**..................................................................................................  **1**

**Installation** .................................................................................................... **2**

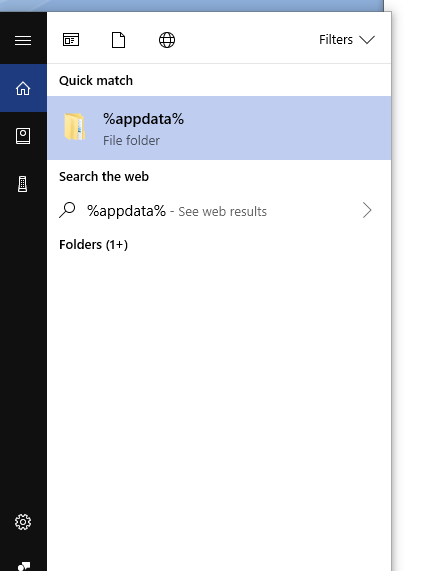
**Maintenance** ................................................................................................. **3**

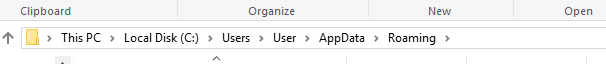
System Installation

The DTCC Toner Inventory System is an application designed on the Windows 10 operating system. This means, that all further installation and setup with be done through the Windows 10 operating system. A critical assumption being conducted is that the computer being used for installation has administrative access in order to comply with the dependency installation requirements.

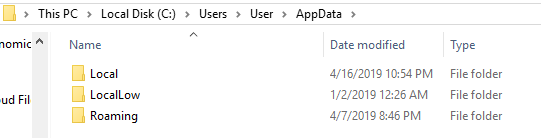
The DTCC Toner Inventory System was developed using the Eclipse Photon IDE environment version no. 4.8.0. To begin the installation, Eclipse Photon should be installed on the computer being used in the first step. NOTE: A copy of Eclipse Photon can be downloaded from Eclipse’s official website: <https://www.eclipse.org/downloads/download.php?file=/oomph/epp/photon/R/eclipse-inst-win64.exe>. (*After the download is complete, users can run the program and go through a series of actions that will install the Eclipse IDE to the selected computer).* A helper tool by the name of “Scene-builder” was used during this development in order to format within .fxml files for the GUI portion of this project, therefore you may want to have this tool for code alteration, this tool can be downloaded from Gluonhq’s official website: <https://gluonhq.com/products/scene-builder/>. (*After the download is complete, a series of instructions*

[**Scene-builder installation process:** *first download scene builder from the site linked above. Next after the installation is complete, navigate to your computers task bar (typically at the bottom left of your screen) where you will then type in “%appdata%”, this will bring up a folder named “roaming” or “%appdata% depending on your system. Click this folder. After you open this folder, you will have to navigate from “roaming” to “Appdata” by clicking the previous folder at the top of your window where it shows the file path. After doing so, you will then see a series of folders, one of which is labeled “Local”, navigate to that folder and open it. You will then see a list of folders from your system, within this folder there should be a folder named “SceneBuilder, the file path at this point should look something similar to “C:\Users\computer\_name\AppData\Local\SceneBuilder\” within your window. Next right click “SceneBuilder” and select the option “Cut”. Then navigate to “This PC” typically to the left of your window, then select your C: drive. Next right click anywhere within the C: drive that is not a folder or file and select “Paste”. After doing so your Scene-builder is now in the correct directory and you can begin working on the project.]*



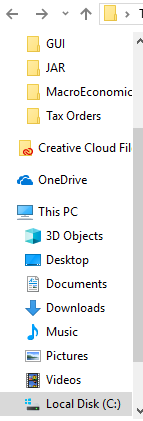


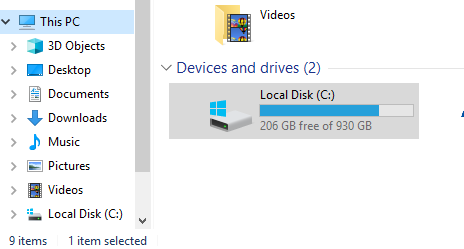
https://i.gyazo.com/37b50102ac554867c629c32fb66566df.png

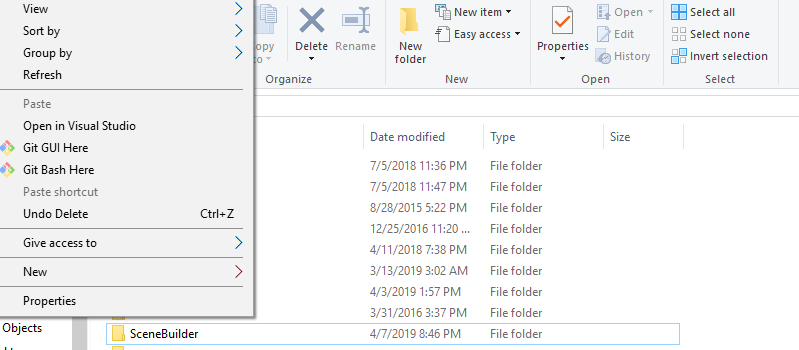


https://i.gyazo.com/9f09d1b556c4a6ded8bd2ae37ccc6fb2.png

https://i.gyazo.com/732c1fdc8bc50baa9f5e764208470073.png

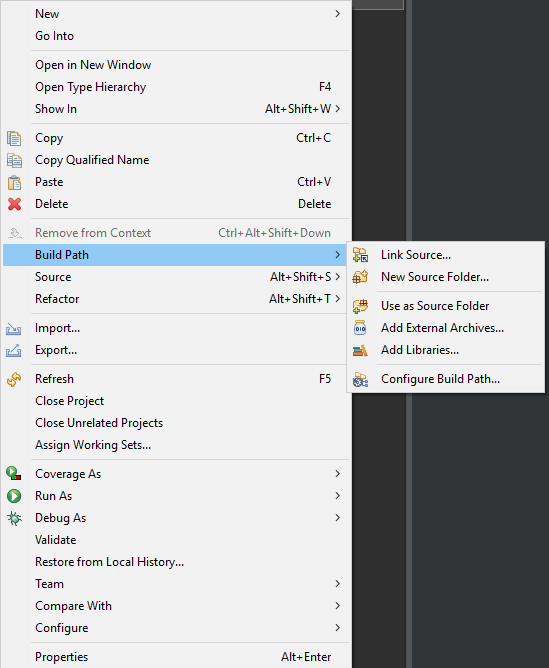


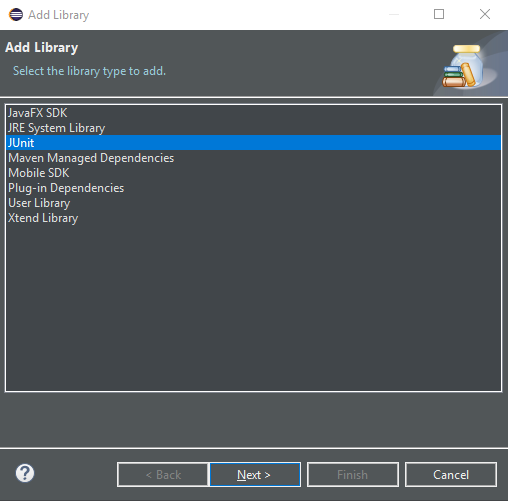


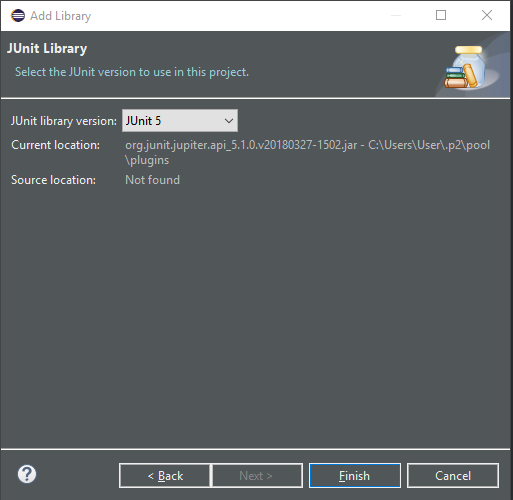


Various build dependencies were used during the creation of this project, these dependencies consist of JUnit 5 for testing, and (JavaFX SDK or e(fx)clipse) for the GUI application. Please see below on installation instructions.

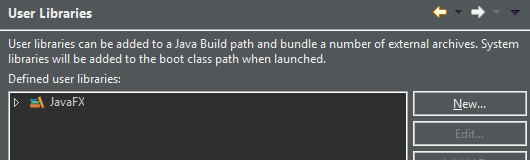
[***JUnit 5:*** *Included in Eclipse Photon, in order to include in your project, right click your project, hover over “Build Path”, and select “add libraries”. After, you should see a list of dependencies, one of which is JUnit. Select JUnit and click “Next”. Select JUnit library version as “JUnit 5”, then click finish.]*

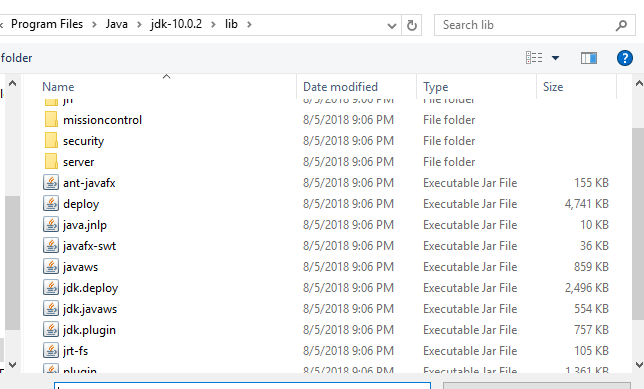




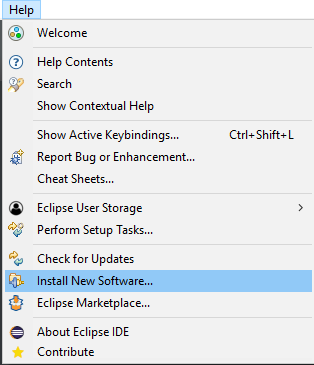


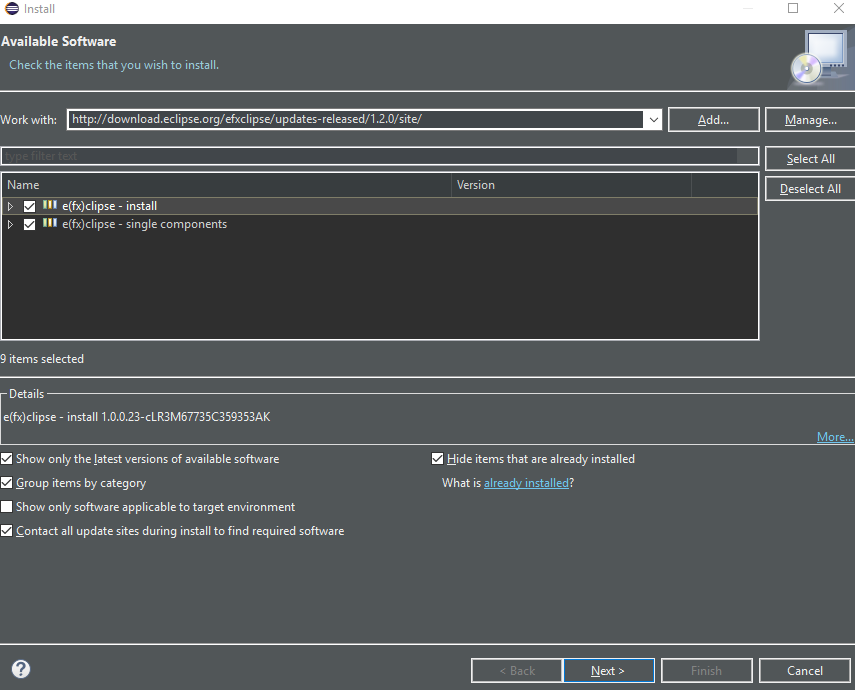
*[****JavaFX SDK:*** *In your Eclipse Photon there is a menu bar at the top. Select Window then scroll until you see preferences. After selecting preferences you will have a new window pop up with various options. Scroll down until you see the option “Java” and double click that opening more hidden options. Look for “Build Path” and double click that option, next you will see another set of options. Select “User Libraries”. You will see a window which should be empty or contain Java “JRE System Library”. Select this and click the remove button to the right. Next, select “New” and create a new library named “JavaFX” (the naming is optional but for the purposes of installation we will call it JavaFX). After the library is created select the library and click the button to the right that says “Add External Jars”. Clicking this button will open up your file system and you will now have to navigate to your JDK folder under “Java” (this is typically under your C: drive within the “program files” section of your file system). After you get to your JDK file, navigate to the JRE file and then LIB. In the LIB file you will see multiple .jar files, select the one named “jfxswt.jar” and click the Open button.]*





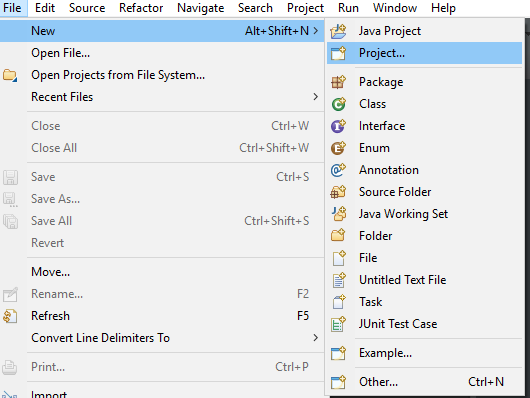
*[****(e(fx)clipse):*** *In your Eclipse Photon there is a menu bar at the top. Hover over “Help” and select “Install New Software”. Within the “Work with:” box of text, delete the current text and paste in* [*http://download.eclipse.org/efxclipse/updates-released/1.2.0/site/*](http://download.eclipse.org/efxclipse/updates-released/1.2.0/site/)*. Next name your repository (for this installation we will name it e(fx)clipse). After naming the repository, click OK. This will bring up a new window and you will see a few options within the installation option box, under the installation box you will see a button called “Select All”, click that button then to the right of the window click “Next”. On the following window you will see installation details, we will not be concerned with this, click “Next”. The following window will bring you to the terms and conditions of the plugin/dependency, first read the terms and conditions, then tick the box labeled “I accept the terms of the license agreements”, and click “Finish” at the bottom of the window. After the installation you will need to restart your Eclipse IDE]*

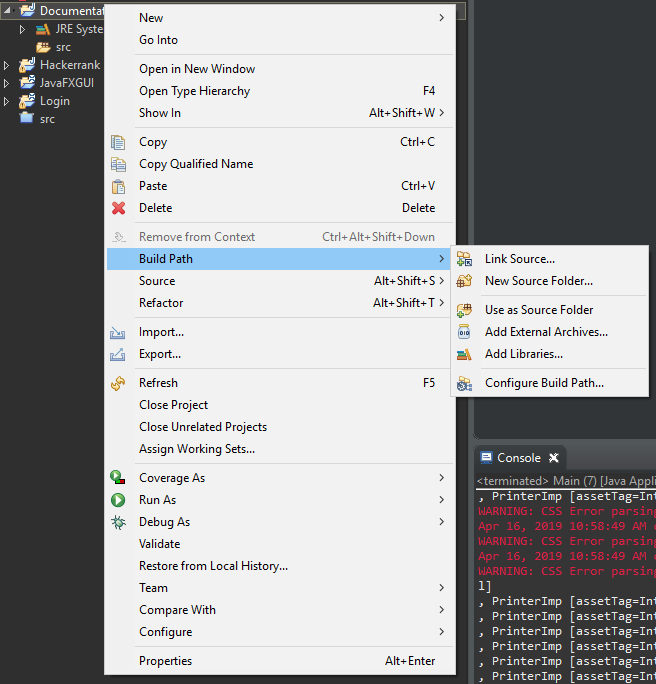


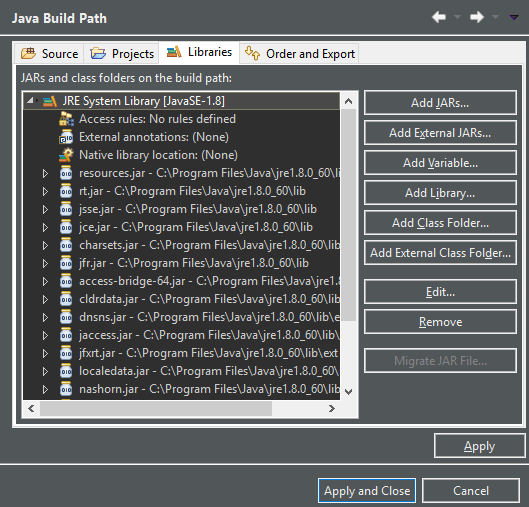


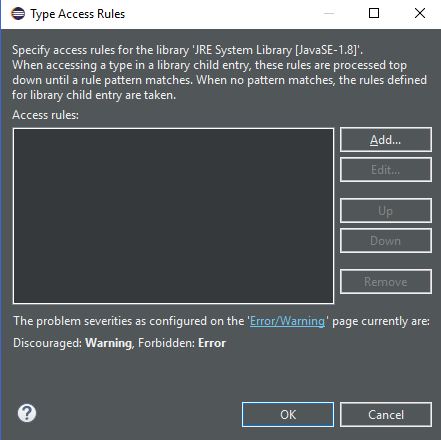
*[****IF none of the JavaFX installations worked correctly, there is a workaround, please follow the following steps in this occurs]***

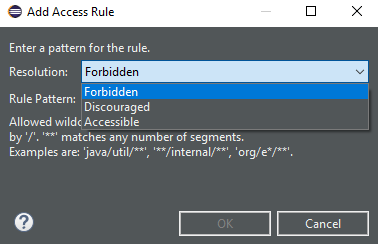
Alternative method: Create a normal java project by clicking file at the top left of your Eclipse IDE, hovering over “New”, and selecting “Java Project”. Name this project, for this example we will use “DocumentationExample”. Next navigate to the project in the “Package Explorer” and right click the project, this will bring up a menu with various options. Hover over the option that says “Build Path”, after doing so you will see another set of options, select “Configure Build Path”. After this you will see a set of buttons and a table of your “Libraries”. In your libraries there should a library named “JRE System Library”, click the arrow to the left of the library, this will cause a drop down of .jar files to show within the library. Within the library there should be a folder labeled “Access rules: No rules defined” assuming you have not added rules yourself previous to this installation process. Double click on this folder; this will bring up another menu. After doing so you will click the button to the right that is labeled as “Add”. Clicking this button will bring up another menu that has a drop down menu labeled “Resolution”, and a text box that is labeled “Rule Pattern”. First click the drop down menu and select “Accessible”, then in the Rule Pattern, type in “**javafx/\*\***”. Then click OK, then on the previous menu click Apply and Close at the bottom right corner of the window. You are now ready to work with JavaFX

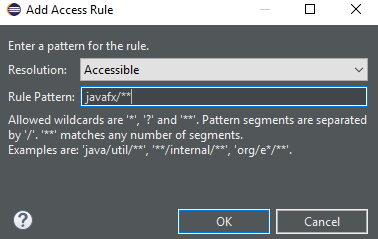








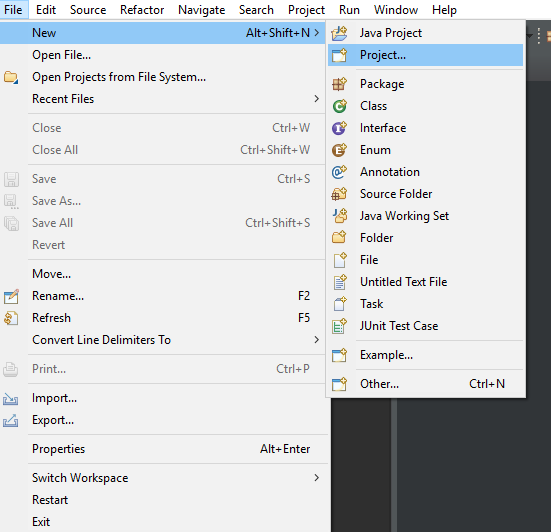


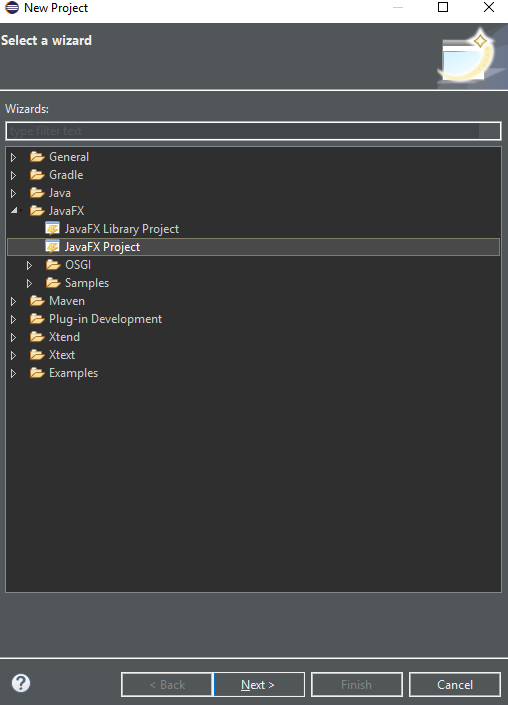


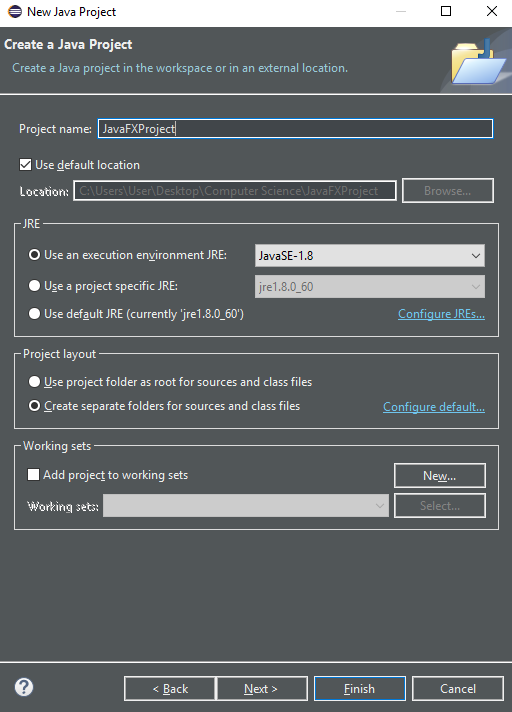
Now that Eclipse Photon is platform has been installed on your machine, the DTCC Toner Inventory System application can be downloaded from the online repository. You can download the project from: <https://github.com/dclenden/Toner-Inventory-System/tree/master/application>.

Once a copy of the project has been exported to your local system, locate the directory you chose and right click and select copy on the exported project. Next, launch Eclipse Photon, after Eclipse has finished launching, navigate to the top right of your screen, where you will see “Workspace”, click this button. Doing this should change the view of your screen. At the top of the program you will see a menu bar, click the first option on the menu bar labeled as “File”. After you click this option you will see a menu, hover over “New”, and select “Project”. Next scroll to the folder labeled “JavaFX”, right click this folder. A series of options will appear, one of which is “JavaFX Project”, click this and select “Next” at the bottom of the window. The next window will indicate a name, create a name and select “Finish” at the bottom of the window.

Open your downloaded project from github open the project “Toner-Inventory-System”, select all of the items within the file and right click, then select “Copy”. Next within your Eclipse IDE, open the project you created, and right click, then select “Paste”, all of the contents within the project should be copied over to your JavaFX project within Eclipse and you are now ready to begin working on the project.







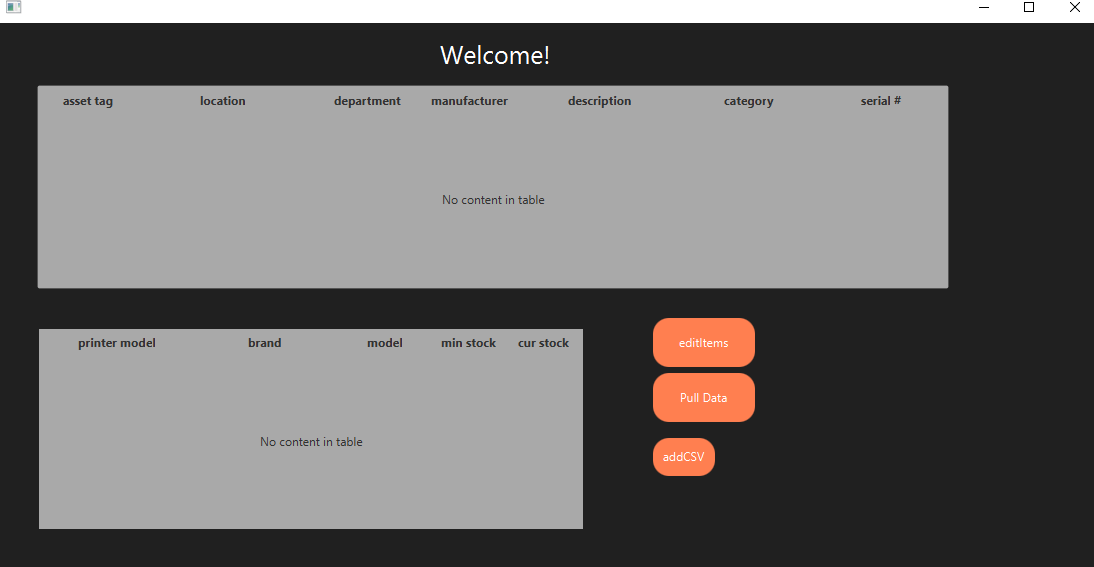
System Maintenance

If any maintenance, changes, or additional functionality needs to be made to the DTCC Toner Inventory System, it is helpful to know some of the ins and outs of the code and to be made aware of some of the specifics of the code. This section of the document will go into a bit of detail on some parts of the project to help the maintenance person understand the project.Because this project is a 3-tier project, this discussion of the code will be grouped into the three different tiers of the architecture.

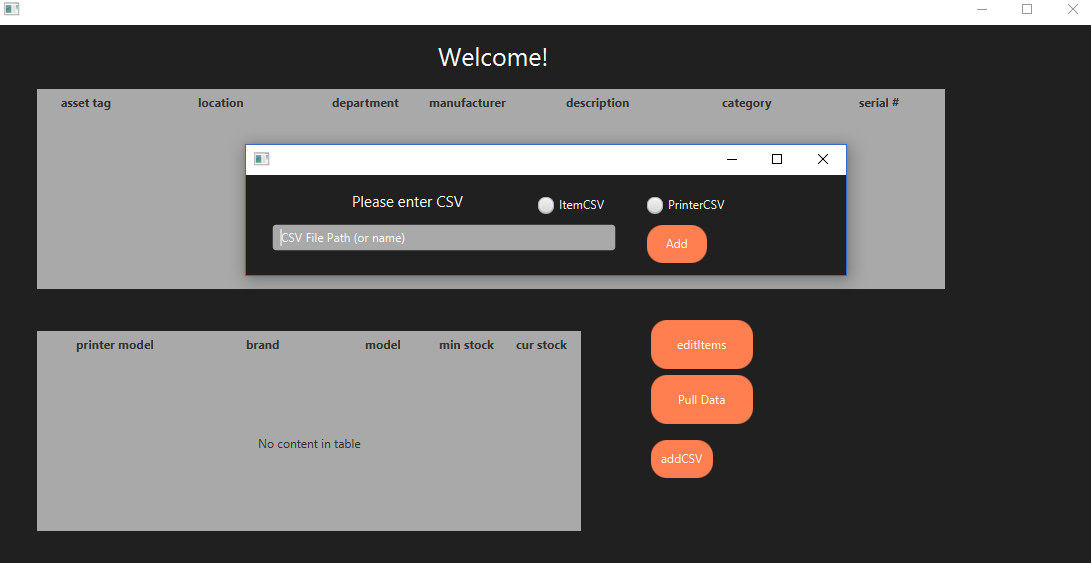
Front End/GUI

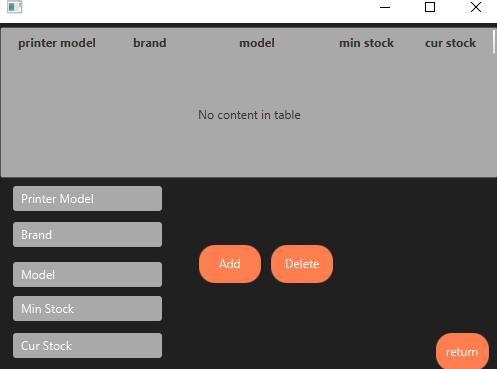
The visual interface of this project consists of 3-6 forms of which the user will navigate through to perform the various operations in the DTCC Toner Inventory System. All GUI functionality is within the “Application” directory (located under the “src” folder). Each form has two files associated with each form, a .fxml file and a Controller.java file. These files are split up to describe different functionality of the overall GUI. The .fxml files hold the parts that make up the forms, such as the local variables and all the pieces that make up what we know as a GUI: text, text boxes, menu items, tables, etc. The Controller.java files will perform operations on the backend based off of the various ActionEvents that will occur according to the users input. Above all of these various files there will be a .css file in which describes the appearance of the project, and how each component of the GUI is styled.

Of the forms in this project there is one that stands out. The most critical form is the “Main” view form. This form is going to be the main view of the program and the starting screen for the end user. This form will be connected to the “MainController.java” class, in which all of the various operations from the user may be handled. The key functionality of this Controller comes from the CSV\_DBIMP object named “dao” (data access object). The dao object, will be performing the connection between the back end and the GUI. (Below is a depiction of the main form)



The various other forms are supporting forms, and will be interacted with through the main form, these forms also have their own Controller.java class, but will interact with the MainController.java class in order to perform operations through the single static dao. (Below is a depiction of the various other forms)





It is essential that the maintenance team understands that the GUI side of the application talks directly with the business logic side of the project, and there is dynamic interaction between them, such as passing Item/Printer CSV’s through the dao. This and the aforementioned information about the GUI side of the application is to help the maintenance person understand the relationships involved so that he/she can be careful about changing the code and knowing where to be careful.

Business Logic/General Classes

The business logic subsystem of the application is between both the GUI side and the data side of the 3-tier architecture, and so it is critical that you are mindful how these systems interact with each other. Although this portion of the system isn’t the most complex, it will still require some dissection to fully understand the system as a whole in order to maintain it.

The general classes of this project are containers for the data extracted from the various CSV files that are to interact with this program, these files are located under the “InventorySystem” directory and each .java file will have an associate interface that is implemented alongside the class. At the highest level of this system stands the CSV\_DBIMP.java class. This class contains the functionality of parsing the CSV files for the Printer and Item objects, while also containing the objects that hold that said data in an array. The only unique thing about the CSV\_DBIMP.java class is that there is design decision that allows for only one instance of the class that will act as the bridge between all components. This will ensure that the data within the object is static amongst all components within a single object, so that all data is the same amongst all components.

The only other classes consist of ItemImp.java, TonerImp.java, and PrinterImp.java, these classes are to provide functionality to it associated objects and act as containers that will be handled within the main class CSV\_DBIMP.java.

These aspects of the business logic classes are the only things to make note of when doing any sort of maintenance to the application. The things to keep in mind is that CSV\_DBIMP is to have one instance and act as a bridge amongst all components, while the other general classes are containers for the various objects of the program and contain minimal functionality as well as the associated getters and setters.

Database and Backend

The physical database is created within the code and is implemented using the various CSV files. In order to retrieve the data from the CSV’s you will have to call the readItemCSV & readPrinterCSV functions on the data access object with a parameter of the CSV file path (or have a file path already determined via the “storeCSV” functions). The database is created in the CSV\_DBIMP.java class and is implemented with an instance of this class; the data that is inserted into the data access object is extracted from the various CSV files and is stored in a Item / Printer container object. Within this file there are the various functions that are used to interact with the data that is stored within the data access object.

The CSV\_DBIMP class is a dual class that will deal with both the back end operations between classes and the data interaction between the various CSV files. These files will be extracted to the itemList array and the printerList array within the data access object. These arrays will contain a list of its respective object type that will have the attributes extracted from the CSV files.