Impaq’s Point of Sale Manual 1.0

*Agustin Cabra – February 2015*

Contents

[1. Folder Contents 3](#_Toc412076288)

[2. Installation on eclipse 4](#_Toc412076289)

[3. Final User View 8](#_Toc412076290)

[4. The ‘data’ configuration folder 13](#_Toc412076291)

[a. The devices.txt file 13](#_Toc412076292)

[b. The products.txt file 14](#_Toc412076293)

[c. The config.properties file 15](#_Toc412076294)

[d. Setting a tax value 16](#_Toc412076295)

[5. Technical details 17](#_Toc412076296)

[a. Javadoc 17](#_Toc412076297)

[b. Code coverage 18](#_Toc412076298)

[c. UML Class Diagram 19](#_Toc412076299)

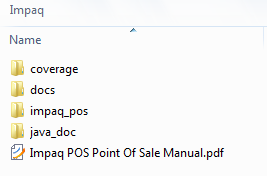
[d. Unit Testing 20](#_Toc412076300)

[6. Conclusions 21](#_Toc412076301)

# Folder Contents

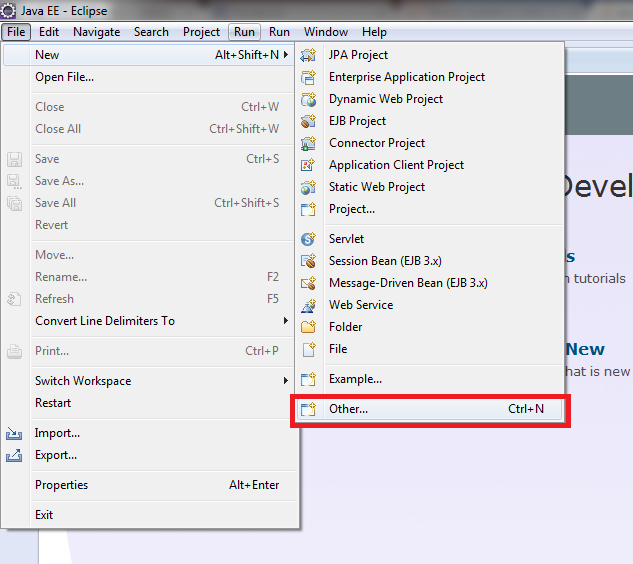
The zip compressed folder contains the following items:

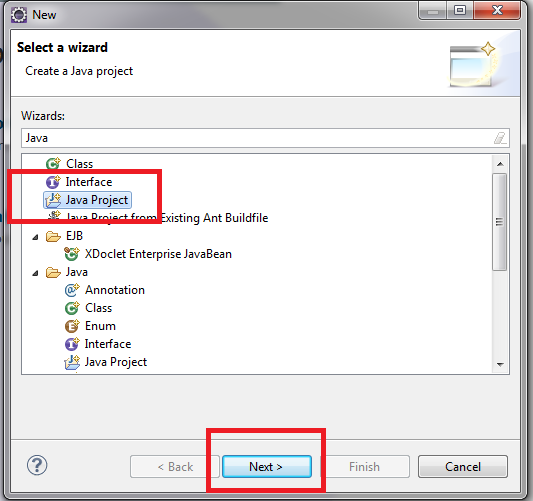
* coverage folder containing the export of the last test run to the project
* docs contains source of Class Diagram and a PDF file exported
* impaq\_pos contains the JavaProject information
* java\_doc references the documentation generated by javadoc after using the annotations through the code
* This manual.



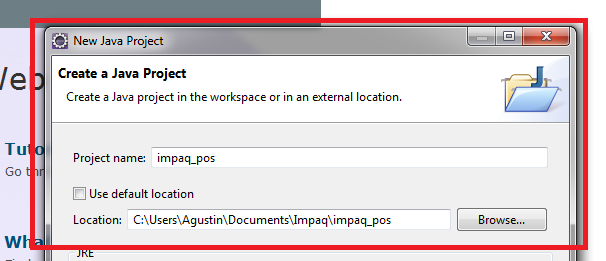
# Installation on eclipse

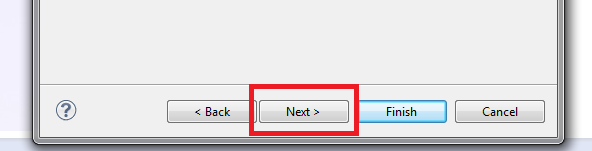
1. Open eclipse and select new Project (Java Project)



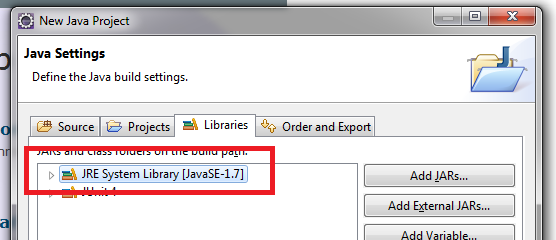


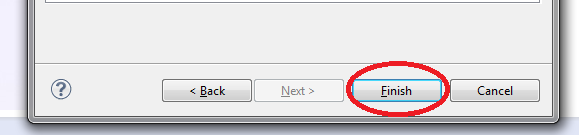
1. Select the folder impaq\_pos as the Location





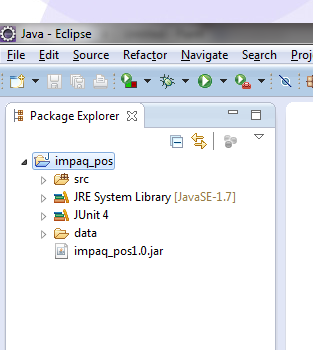
1. Verify the java version 1.7 to run the project





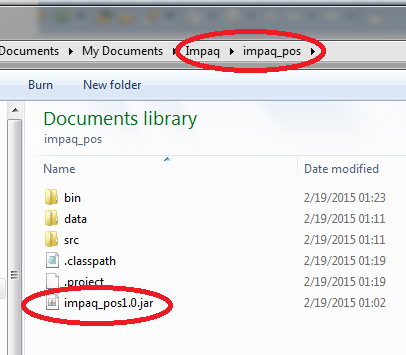
Note: If older version of java is configured some features as Unit Testing or coverage code may not be available. (<http://java.com> for more info)

1. The project should be now available



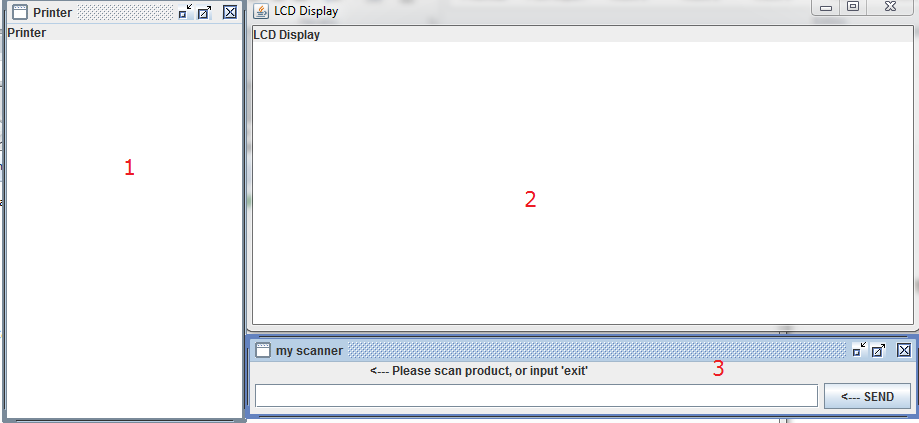
# Final User View

1. Under the folder impaq\_pos there is the runnable version double click it.

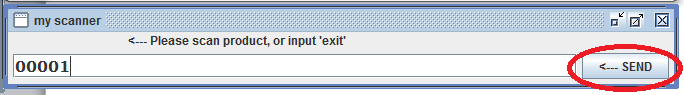


Note: this jar must be executed in the same folder as “data” thus it requires all those configuration files to start.

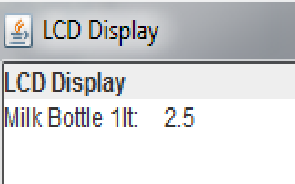
1. The three default views



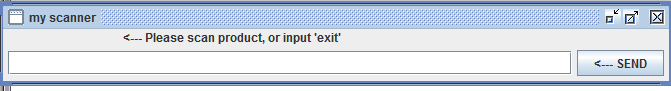
1. Represents the printer (Invoice details will be shown there)
2. Represents LCD Display (Details of scanning a product will appear there)
3. Barcode Input Scanner (Responds to ‘exit’ or to barcodes as defined in the config.properties file. The current configuration has by default 16 products with 5-digit barcode from (00001 to 00016).
4. Scanning a product: type in the barcode and hit the ‘Enter key’ or the ‘send’ button.

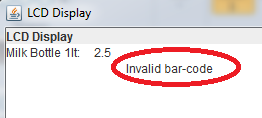


According to specification if the barcode is valid on the display will show its price and name.

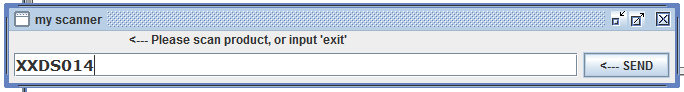


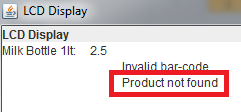
If the input is sent empty ‘invalid code’ message appears on the Display



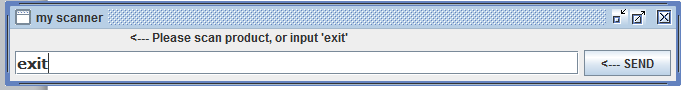


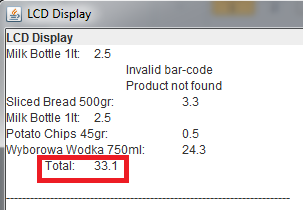
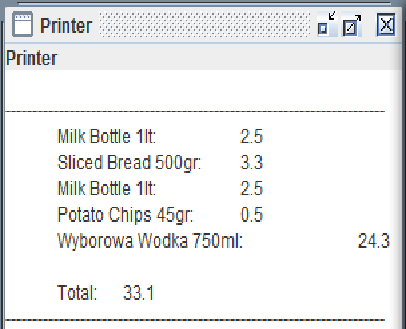
And if the barcode sent is not found as a configured product





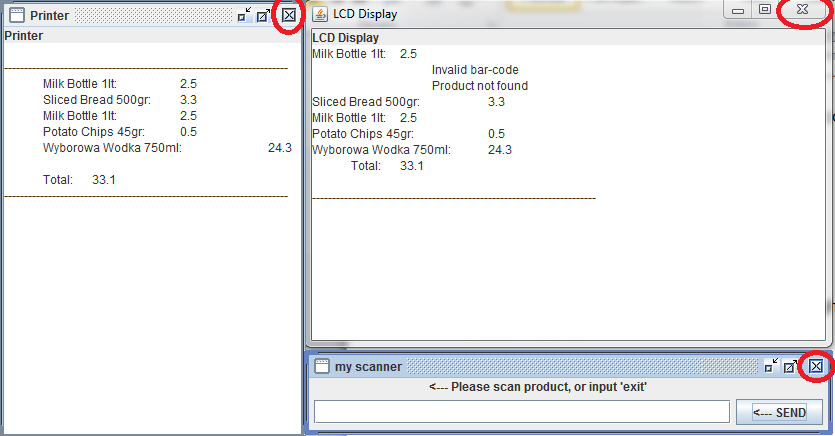
And finally after sending the key-word ‘exit’, the total is printed on Display, and the invoice is printed on the ‘printer’ window.



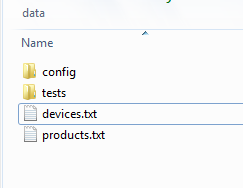


At this point the systems starts a new order list and the process can continue as explained.

1. To Exit the system click on any of the ‘x’ symbols on the top right corner of the windows.

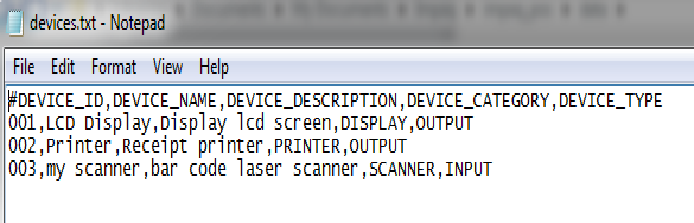


# The ‘data’ configuration folder



## The devices.txt file

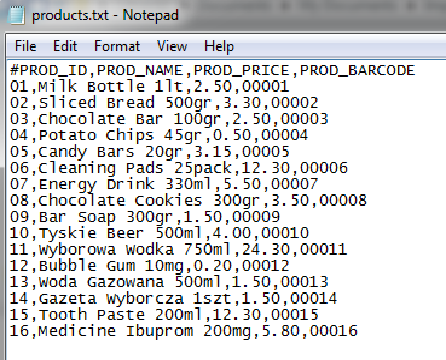
This file contains the mock of the devices that will be loaded into the system, currently are available three categories (DISPLAY, SCANNER and PRINTER) in two types (INPUT and OUTPUT).



The header of this file contains the specification on how each column is taken as property of the device.

## The products.txt file

This file contains the list of products available to use in the system (please note that currently there is no **inventory track** in the system so is assumed that the inventory is always available from the products defined in the file, this functionality may be available in future releases).

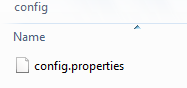


This file also contains a coma separated value format, where every column can be described by its corresponding tag on the header.

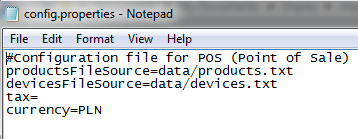
Note: The use of ‘Prod\_id’ and ‘barcode’ is thinking on future releases when multiple catalogs can be loaded so a product may acquire diferent price according to the catalog barcode assigned.

## The config.properties file

This file is contained on the config folder and contains the basic information for the system to run, (before modifying it make sure all the implications are clear)



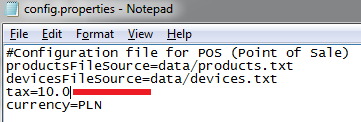
All the configurations are stored here, it resolves the name files for the products and for devices, and also has a two additional parameters, ‘tax’ and ‘currency’, thinking on future releases, where the tax can be calculated and the currency shown depends on it.



If the tax property is stored as a number greater or equal to zero, the system will print additional details calculating subtotal, tax collected and total.

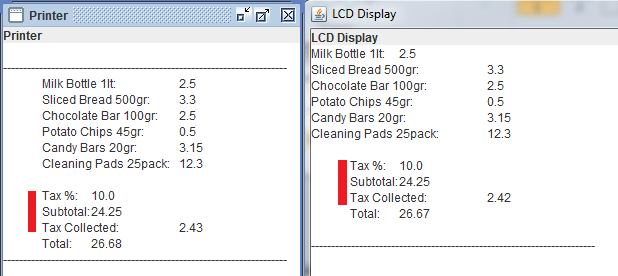
## Setting a tax value

In the following example the system prints the details based on a tax modification of 10.0.



For the configuration changes to take place a restart of the system is required.

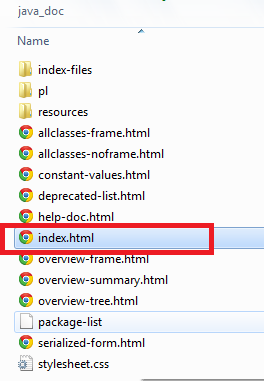
In the following example the new values can be seen.



# Technical details

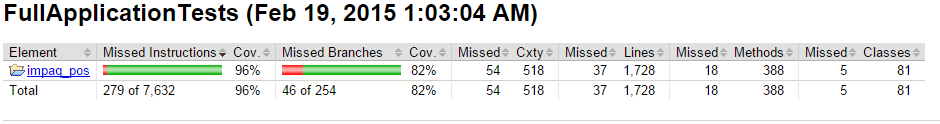
## Javadoc

The source code used javadoc annotations on comments for classes and methods, in order to provide a full documentation of the code. This information may be accessed by opening the index.html file under the folder ‘java\_doc’ folder.

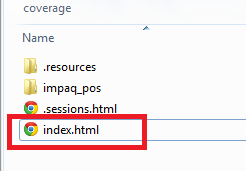


## Code coverage

The coverage test results on last run, provided a 96% coverage, where the missing 4% corresponds to exception handling code where basically no relevant information may be required to return.



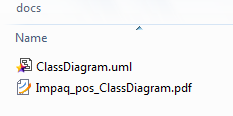
The results can be seen on opening the index.html file under the folder coverage.



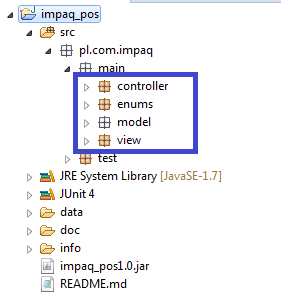
The coverage documentation was executed using the ECLEmma eclipse plugin (<http://www.eclemma.org/> ) available for free on the eclipse market place.

## UML Class Diagram

Under the docs folder the source files for StarUML (<http://staruml.sourceforge.net/v1/download.php>) for class diagram and an exported PDF versions are available, the image was not included in this document to keep an adequate resolution.



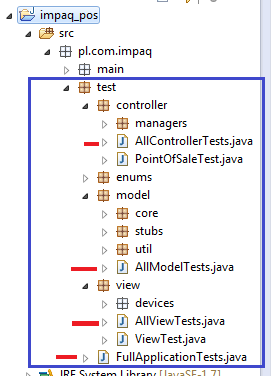
The system was designed based on MVC design pattern; It presents a clear path to add new functionalities to the system. The package presentation reflects this.



The singleton pattern was also used to keep only one instance of PointOfSale, the View and the PropertiesReader class.

## Unit Testing

The unit testing was developed using JUnit4, in the source there are two major packages under ‘pl.com.impaq’ main and test, under the test package all the JUnit test files and Test Suites are available, there are 4 major Suites to test the code as shown in the following figure.



The FullApplicationTest.java file contains all the class specifications and is the one that complies with the 96% percent coverage specified on the previous subjects.

# Conclusions

The system specifications where short, and the huge demand and extension of the possible PointOfSale system, made it a very challenging task to be performed, any questions or comments on this manual are welcome.

[cabra.agustin@gmail.com](mailto:cabra.agustin@gmail.com)  
<https://github.com/acabra85>