**REPORT FOR SOFTWARE DEVELOPMENT GRADED UNIT 2**

**UNIT: H48W 35**

**Angelo Romel Lopez**

**May 2015**

**HND SOFTWARE DEVELOPMENT 2ND YEAR**

**COMPUTING – BUSINESS SCHOOL**

**PAISLEY CAMPUS, WEST COLLEGE SCOTLAND**

**Contents**

[1. Introduction 3](#_Toc328058742)

[2. Planning 5](#_Toc328058743)

[2.1. Initial Investigation 5](#_Toc328058744)

[2.2. Approach to the project 13](#_Toc328058745)

[2.3. Justification of Approach 14](#_Toc328058746)

[2.4. Project Plan 15](#_Toc328058747)

[2.5. Information Sources 16](#_Toc328058748)

[3. Developing 17](#_Toc328058749)

[3.1.Design/Production of Application 17](#_Toc328058750)

[3.2. Testing/Implementation 308](#_Toc328058751)

[3.3.User Documentation/Guide 320](#_Toc328058752)

[4. Evaluating 321](#_Toc328058753)

[5. Conclusion and Recommendations 322](#_Toc328058754)

[6. Bibliography 323](#_Toc328058755)

[7. Appendix 324](#_Toc328058756)

# 1. Introduction

This report is a requirement for the Software Development Graded Unit 2; H48W 35 and is written and prepared for Computing lecturer Michelle Blaire.

The undertaking of this project would assist me in my effort to develop and improve my software development and technical skills in Java as my primary programming language. It would also help me in gaining more experience and better understanding of using proper object oriented design, techniques and patterns.

This report will cover the planning, developing and evaluating stages of the project.

I will be developing a Point of Sale application in Java that connects to a MySQL database and will run on Raspbian (Debian Linux distribution) operating system. Both the front end and back end will run on a Raspberry Pi B+ device and will use a 3.2/3.5 LCD touch screen and a portable barcode scanner for user input/interaction. The entire unit when fully developed and assembled will become a portable Point of Sale device that will be powered by a portable USB battery. I am naming my project as “POS-Pi”, which stands for **P**oint **O**f **S**ale application on a Raspberry **Pi**.

A Raspberry Pi is a low cost credit card sized computer developed by the Raspberry Pi Foundation in the UK. It can be powered by a 5V micro USB charger or at least 4xAA batteries.

I will carry out the design, development and testing of POS-Pi independently and will aim to produce a code that adheres to a high standard of object oriented design and techniques. I will be referring to the following online reference materials as my guide:

[1] ORACLE online Java Documentation (http://docs.oracle.com/javase/tutorial/index.html)

[2] Tutorials Point Java Programming (http://www.tutorialspoint.com/java\_technology\_tutorials.htm)

[3] MySQL Reference Manual (http://dev.mysql.com/doc/)

[4] Raspberry Pi Documentation (http://www.raspberrypi.org/documentation/)

I hereby declare that I, Angelo Romel Lopez have not and will not plagiarize someone else’s work or code. I will properly quote, acknowledge and give credit for any text or images and illustrations that I feel are needed for my report.**Business School, Paisley**

**Please complete, sign and submit this declaration with your assessment**

I declare that:

* the attached assessment is all my own work
* all information from other sources has been correctly cited and referenced
* I have taken reasonable care to make sure that my work has not been copied by anyone else

Name: Angelo Romel Lopez

Course/Class: HND Computing: Software Development Year 2

Unit Title: H48W 35

Assessment Title

Computing: Software Development Graded Unit 2

Signature Date

**Plagiarism is “taking the work or idea of someone else and pretending it is one’s own” (Oxford English Dictionary) which includes:**

* Submitting a report/assignment written completely or partly by someone else
* Preparing a report/assignment for another student to submit
* Coping a report/assignment or allowing your own assignment to be copied by another student
* Using material sourced from the internet or textbooks without acknowledging the source
* Not labeling diagrams/illustrations etc.

# 2. Planning

## 2.1. Initial Investigation

**Project Initiation Document**

**Project Title**

**POS-Pi**

A **P**oint **O**f **S**ale application written in Java that runs on a portable battery (rechargeable) operated Raspberry **Pi** (touch-screen and barcode scanner enabled).

**Purpose of the Project**

The project aims to increase my knowledge in software development and to further develop my understanding of the Java programming language using object oriented design, techniques and patterns. The project will require me to research and study the following subject areas:

[1] GUI programming with Java Swing.

[2] MVC design pattern.

[3] Database design and administration with MySQL.

[4] Connecting to and manipulating data stored in MySQL with Java.

[5] Deploying and running a Java application in Raspbian (Linux distribution for Raspberry Pi).

[6] Handling barcode scanner input with Java.

**Scope of the Project**

A “Point of Sale “ or “Point of Service” application is a program that runs on a machine or workstation and is sometimes referred to as a “payment terminal” or simply a “checkout counter”. It deals with customer purchases and transaction. A Point of Sale program is usually a part of a solution/system that involves inventory management and cost accounting. However, for this project, I would be focusing on the ‘customer order’ side of the system. The development and testing phase for the project will be carried out on the Raspberry Pi.

The following software is needed to be installed on the Raspberry Pi to develop and run POS-Pi:

[1] Java Development Kit (JDK 7/8).

[2] BlueJ (Java IDE)

[3] MySQL database management system.

[4] PHPMyAdmin.

[5] Apache web server.

A Raspberry Pi is a credit card sized computer that runs on an operating system that is based on Linux.

For the purpose of this project, the Raspberry Pi will be fitted with a 3.2/3.5 LCD touch-screen display attached to a handheld USB powered barcode scanner. A small portable USB battery bank will be secured to either the Raspberry Pi or the barcode scanner to provide power.

**Initiator of the Project**

My concept for the POS-Pi came about when I saw a full sized POS terminal running on Raspberry Pi. It was a proper POS terminal that included a thermal printer; card reader, pole display and a cash drawer. I have wanted to take advantage of the Raspberry Pi’s size by developing a low cost Point of Sale device that runs on a portable USB power bank.

**Names of people available for advice and assistance**

Clare Lowe – Computing Lecturer

Michelle Blaire – Computing Lecturer

**Time Limitations**

Planning phase – February 4, 2015 deadline.

Development phase – April 29, 2015 deadline.

Evaluation phase – May 24, 2015 deadline.

**Other Information**

The cost for a Raspberry Pi is around £25 -£35. With the hand-held USB barcode scanner (£13), 3.2/3.5 LCD touch screen (£18) and a 13000mAh USB power bank (£22), the overall cost for the project would be around £88. The cost of POS-Pi with a mini Bluetooth thermal printer (£45) and a credit card sized (iZettle) credit card reader (£49) attached would be about £182. It is still cheaper compared to commercial hand-held POS devices which cost between £300 to £800. Apart from being low-cost, the main advantage of the POS-Pi is the fact that it is a highly configurable and customisable Linux based minicomputer.

Ideally, the portable POS-Pi device should connect to a remote server/service such as a stock control or inventory system. The device can be fitted with a USB Wi-Fi adapter/dongle to give it Wi-Fi connectivity. I have already tested this functionality at the time of writing the planning phase for this report and I find the speed performance to be fast and reliable. However, for the purpose of this project, both the front end and the back end of the application will reside on the same device.

**Other Limitation**

Due to budget constraints, I will be using my own desk jet printer at my home instead of a thermal printer to test the “print receipt” functionality of the POS-Pi application. Card payment will not be included for this project, but will be a priority for future development and improvement of the project. For this reason, the application will only accept cash payments.

**Requirement Specification**

|  |  |
| --- | --- |
| Program No: 0001 | System: |
| Author: Angelo Romel Lopez | Date: |

***Brief Description***

A Point of Sale application written in Java that records customer sales and payments. The application can accept input from a physical keyboard, virtual keyboard, mouse, touch-screen and barcode scanner. A point of sale application is typically used in retail stores and supermarkets.

|  |  |
| --- | --- |
| ***Inputs*** | ***Outputs*** |
| [1] User Name and password.  [2] Item barcode.  [3] Cash amount.  [4] Add item to stock. | --Application main screen menu.  --Item description and price.  --Reduced stock amount after sale.  --Change amount.  --New items.  --Update stock amount of existing items. |

**Functional Requirements**

***Main Functions***

[1] Retrieve user name and password.

[2] Retrieve item description and unit price.

[3] Manage cash payment.

[4] Reduce stock amount.

[5] Cancel Sale.

[6] Print receipt.

[7] Add item discount.

[8] Update item discount

[9] Delete item discount.

**Description:**

**Name:** Retrieve user name and password.

**Description:**

**Input:** Touchscreen virtual keyboard/physical keyboard

**Output:** Valid user name and password.

**Action:** Query MySQL database and retrieve user name and password where user name and password is equal to the input user name and password.

**Precondition:** Valid user name and password.

**Post condition:** user name and password exists.

**Name:** Retrieve item description and unit price.

**Description:**

**Input:** Barcode scanner/touchscreen virtual keyboard/physical keyboard

**Output:** Item description and unit price

**Action:** Query MySQL database and retrieve item description and unit price where item barcode is equal to the input barcode.

**Precondition:** Valid barcode.

**Post condition:** Barcode exists.

**Name:** Manage cash payment

**Description:**

**Input:** Cash payment received.

**Output:** Change to be given.

**Action:** If cash amount is greater than or equal to the amount to be paid then calculate change and save transaction to database.

**Precondition:** Cash payment received is greater than or equal to the amount to be paid.

**Post condition:** Cash on hand after sale is equal to cash on hand before sale + amount to be paid.

Sale transaction saved to database.

**Name:** Reduce stock amount.

**Description:**

**Input:** Barcode

**Output:** Stock amount after sale transaction.

**Action:** Item stock amount – 1.

**Precondition:** Valid barcode. Item on stock is greater than or equal to number of items in sale transaction.

**Post condition:** Item stock amount after sale = item stock amount before sale –total number of items sold.

**Name:** Cancel sale.

**Description:** User can cancel a sale transaction as long as the sale has not been processed or saved.

**Input:**

**Output:**

**Action:**

**Precondition:** Cash on hand =Cash before sale. Item stock = Item stock before sale.

**Pose condition:** Cash on hand =Cash before sale. Item stock = Item stock before sale.

**Name:** Print receipt.

**Description:** Prints out list of items sold, discount on items, item price, total amount to be paid, amount received, change and date/ time of sale.

**Input:**

**Output:** Items sold, sub-total and total amount to be paid.

**Action:** Print items sold, sub-total and amount to be paid.

**Precondition:** Item stock amount after sale = item stock amount before sale –total number of items sold.

**Post condition:**

**Name:** Insert discount on items.

**Description:** Add a discount amount for an item

**Input:** Barcode

**Output:** Discount amount.

**Action:** Insert product + discount.

**Precondition:** Valid barcode.

**Post condition:**

**Name:** Update discount on items.

**Description:** Update or modify a discount amount for an item

**Input:** Barcode

**Output:** Updated Discount amount.

**Action:** Update discount.

**Precondition:** Valid barcode.

**Post condition:**

**Name:** Delete discount on items.

**Description:** Delete or remove discount amount for an item

**Input:** Barcode

**Output:** Updated Discount amount.

**Action:** Item discount = 0.

**Precondition:** Valid barcode.

**Post condition:** Item discount amount is equal to zero (0).

**Non Functional Requirements**

[1] **Usability** – The user interface and navigation must be straightforward, easy to use and understand. Buttons and texts must be big enough to use on a 3.2/3.5 touch-screen.

[2] **Reliability/Availability** – The system is expected to be available 99% of the time. There should be at least one back-up battery available to use while the other is charging.

[3] **Portability** – Although the project is intended to run on the Raspbian operating system which is based on Debian Linux distribution, it can easily achieve a reasonable level of platform independence since the application is developed in Java and MySQL.

[4] **Security –** A security mechanism must be in place to avoid unauthorized access to the system. All users must login with a valid username and password.

[5] **Maintainability** – Standard naming convention must be adhered to when naming classes, methods and variables. The Model-View-Controller (MVC) design pattern will be utilized to allow for the application’s flexibility and maintainability.

**Hardware Requirements**



deal extreme(20/11/2014) sku\_362889\_2.jpg, available from <http://dxcdn.com/productimages/sku_362889_2.jpg> [Accessed 29/01/2015 ]

[1] 3.2”/3.5” LCD Touch-Screen



MODMYPI (no date) raspberry-pi-b+-2-800x800.jpg, available from https://www.modmypi.com/image/cache/data/raspberry-pi-b+-2-800x800.jpg [Accessed 29/01/2015]

[2] Raspberry Pi Model B+



aliexpress (no date) High-Quality-5000mAh-External-Battery-Charger-Power-Bank-portable-power-2-Dual-USB-2A-for-blackberry.jpg, available from http://i00.i.aliimg.com/wsphoto/v0/645743250/High-Quality-5000mAh-External-Battery-Charger-Power-Bank-portable-power-2-Dual-USB-2A-for-blackberry.jpg [Accessed 29/01/2015]

[3] Portable USB Power Bank

ELEGOAL (no date) pl1463338-handheld\_psc\_portable\_barcode\_scanner\_sc8805\_for\_supermarket\_use.jpg, available from http://www.terminalthinclient.com/photo/pl1463338-handheld\_psc\_portable\_barcode\_scanner\_sc8805\_for\_supermarket\_use.jpg [Accessed 29/01/2015]

[4] USB Handheld Barcode Scanner

**Initial Top Level Use Case Model**

**POS-Pi**

Checkout

Customer

<<includes>>

<<includes>>

Clerk

Payment by Cash

<<includes>>

Calculate Total

Scan Item

## 2.2. Approach to the project

**Systems Development Life Cycle Methodology**

Due to limited development time constraints, I will be utilizing the Waterfall method for the project’s systems development lifecycle method. The method will allow me to closely monitor specific stages and deadlines and loop between each phase to make appropriate modifications.

Below is a diagram of a Waterfall model:

Planning

Analysis

Design

Coding and Testing

Implementation

Maintenance

**Analysis and Design Methods/Models**

I will also be using the following analysis and design methods/models for designing the application:

**[1] Use Case Diagram and Description** – It will show the graphical depiction of the collaboration and interactions among the elements of the POS-Pi application.

**[2] Activity Diagram** – a flowchart representation of the actions, activity and workflow of the application.

**[3] Class Diagram** – will provide a graphical overview of the POS-Pi system by describing the classes, their attributes and methods and the relationships among the classes.

**[4] Sequence Diagram** – It will show a graphical representation of a scenario of a Point of Sale application and the interactions between objects and classes and the exchange of messages between them based on a time sequence.

**[5] Database Schema and Entity Relationship Diagram**. – will show a graphical representation of the model and layout of the database and the structure and relationship of the tables involved in the POS-Pi application.

**[6] Data Flow Diagram** – shows the flow of information and the type of data involved in the system by showing:

- How data enters and exits the system.

- What and how the data changes.

- Where data is stored.

**Programming languages used:**

* Java
* SQL
* Linux terminal commands

## 2.3. Justification of Approach

I have decided that the waterfall method to be the best approach to the project due to resource and time constraints. The project is relatively small with clear and defined requirements. As I will be working alone on the project, having clear set tasks, stages and deadlines would be beneficial in monitoring/gauging the progress of the project. The approach will also allow me to loop between phases to make any corrections or adjustments.

The project will be entirely designed, developed and tested at my home using my own resources. The cost for any materials/hardware needed for the project will be purchased at my expense. The resources needed for the development environment is composed of:

**Hardware:**

[1] Raspberry Pi

[2] Monitor

[3] Mouse/ keyboard

[4] Portable USB barcode scanner

[5] USB Wi-Fi adapter

[6] USB portable power bank

[7] 3.2”/3.5” LCD touch screen

**Software:**

[1] BlueJ IDE

[2] JDK 7/8

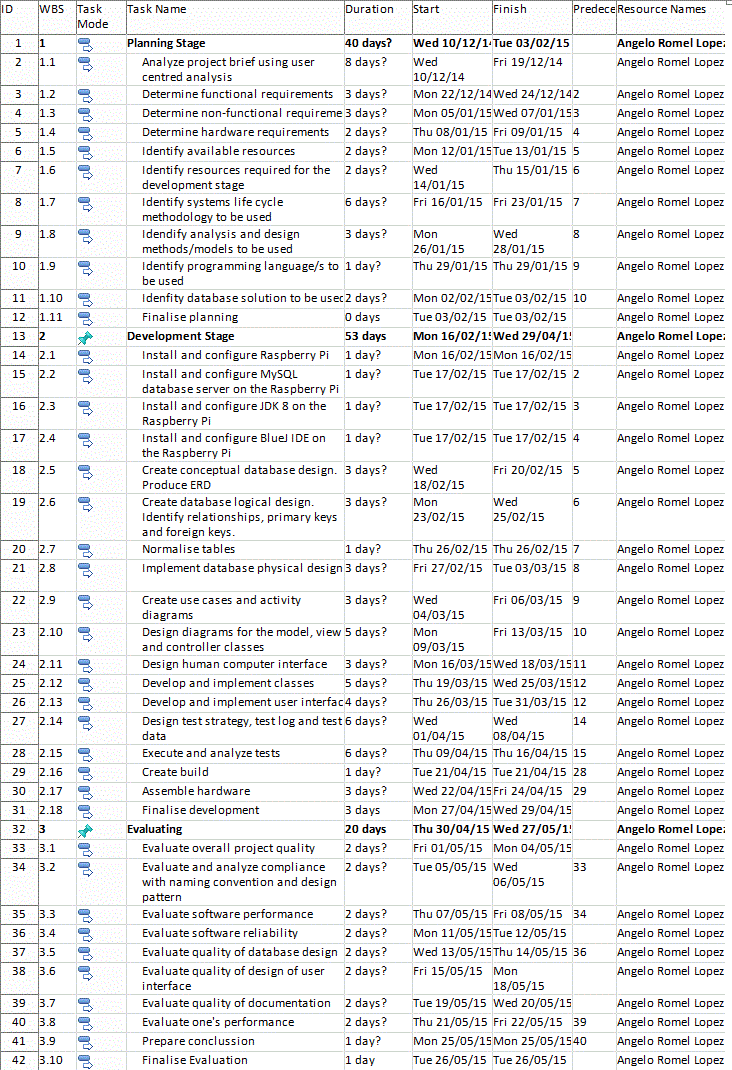
[3] MySQL

[4] PHPMyAdmin

## 2.4. Project Plan

The project plan will help improve my performance and productivity in creating the project by providing clear set tasks and goals. The visual guide that the project plan provides will give me a method to monitor the overall progress of the project.

Below is a screenshots of the actual project plan created with Microsoft Project:



## 2.5. Information Sources

Below is a list of websites and online materials that I have referred to while preparing the initial investigation for the project:

[1] ComputerWeekly [July 2009][website] Point of Sale (POS) Software – Essential Guide, http://www.computerweekly.com/feature/Point-of-Sale-POS-software-software-Essential-Guide [Accessed January2015]

[2] DigitalDining [no date][website] Handheld POS, http://www.digitaldining.com/handheld-pos/ [Accessed January 2015]

[3] Computer Science and Information Engineering [December 2006] [website] A Case Study – Point of Sale, http://www.csie.ntut.edu.tw/sdrc/files/course/20061201/SoftwareRequirementSpecification.pdf [Accessed January2015]

[4] Chip and Pin Machines [no date][website] WorldPayZinc vs iZettle vs Payleven, http://www.chipandpin-machines.com/category/chip-and-pin/ [Accessed January 2015]

[5] Raspberry Pi [no date][website] Raspberry Pi, http://www.raspberrypi.org/ [Accessed December 2014]

# 3. Developing

## 3.1. DEsign/production of application

Top Level Use Case Diagram: POS-Pi Application

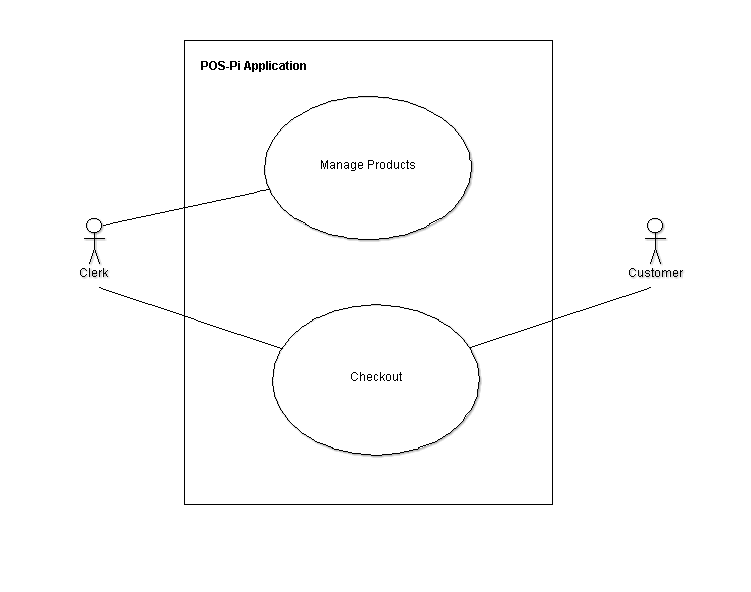
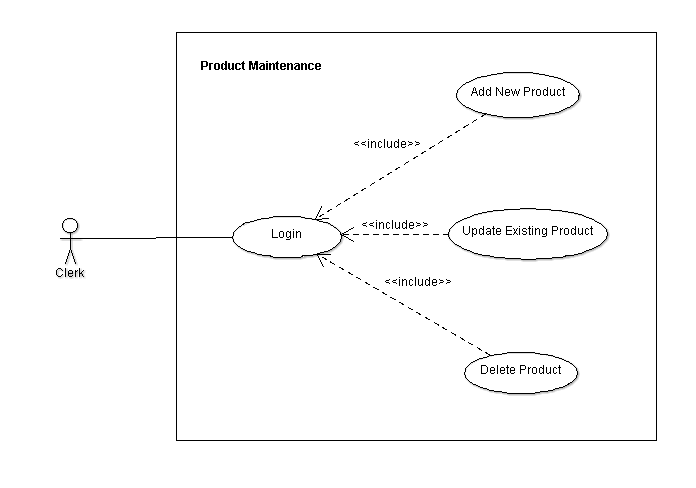


Diagram created with ArgoUML (http://sourceforge.net/projects/argouml/)

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. Available at: <http://sourceforge.net/projects/argouml/> [Accessed April 29, 2015].

Use Case Diagram: Product Maintenance/Manage Products 

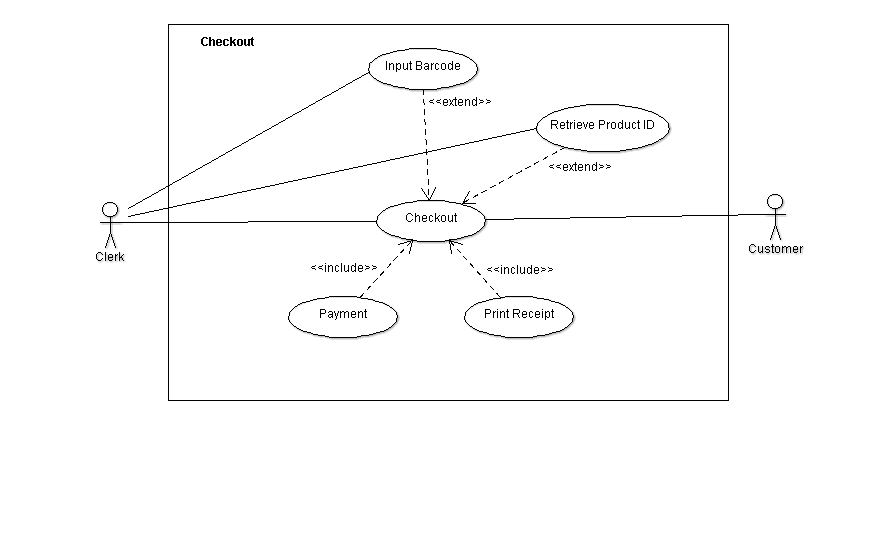
Use Case Diagram: Checkout

Diagram created with ArgoUML (http://sourceforge.net/projects/argouml/)

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. Available at: <http://sourceforge.net/projects/argouml/> [Accessed April 29, 2015].

**Use Case Descriptions – Product Maintenance**

**Use case:**

Login

**Requirements:**

Valid username and password.

**Pre-conditions:**

User has started the Point of Sale application.

**Post-conditions:**

The main menu for the application will be displayed after a successful login.

**Invariants:**

[1] Incorrect username – allow user to cancel or re-enter username.

[2] Incorrect password – allow user to cancel or re-enter password.

**Scenario:**

Ask for username and password

Get username and password

Verify username

If username is valid then

Verify password

If password is valid then

Launch Point of Sale application

Else

Display error message

Allow user to re-enter password

End If

Else

Display error message

Allow user to re-enter username

End If

**Use case:**

Add new product

**Requirements:**

Successful login.

**Pre-conditions:**

User is logged-in and has selected to add a new product in the product maintenance screen.

**Post-condition:**

A new product will be added to the database.

**Invariants:**

Product information does not satisfy the rules of the business – allow user to re-enter product information.

(Rule example: A product can either have a barcode or a ‘uom’-unit of measurement-code, but not both. A barcode is meant to be scanned, whereas a uom code is applied to products where the use of a barcode is not practical or possible).

**Scenario:**

If product information is valid then

Add new product to the database

Else

Display error message

Allow user to re-enter product information

End If

**Use case:**

Update Existing Product

**Requirements:**

[1] Successful login.

[2] Product to be updated exists in the database.

**Pre-conditions:**

User is logged-in and has selected to update an existing product in the product maintenance screen.

**Post-conditions:**

An existing product information is updated.

**Invariants:**

Product information does not satisfy the rules of the business – allow user to re-enter product information.

(Rule example: A product can either have a barcode or a ‘uom’-unit of measurement-code, but not both. A barcode is meant to be scanned, whereas a uom code is applied to products where the use of a barcode is not practical or possible).

**Scenario:**

Allow user to specify the product to be updated

If product is in the database then

Ask user to provide new product information.

If product information is valid then

Update product from the database

Else

Display error message

Allow user to re-enter product information

End If

Else

Display error message

Allow user to select another product from the database

End If

**Use case:**

Delete product

**Requirements:**

[1] Successful login.

[2] Product to be deleted exists in the database.

**Pre-conditions:**

User is logged-in and has selected to delete a product in the product maintenance screen.

**Post-conditions:**

A product is deleted from the database.

**Invariants:**

**Scenario:**

Allow user to select the product to delete

If product is in the database then

Delete product from the database

Else

Display error message

End If

**Use Case Descriptions – Checkout**

**Use case:**

Input barcode

**Requirements:**

Valid barcode.

**Pre-conditions:**

[1] User must be logged in and has accessed the point of sale transaction screen.

[2] User must have access to an input device, such as a barcode scanner/reader; a mouse to access a virtual/on-screen keyboard, or a physical keyboard.

**Post-conditions:**

Product information is retrieved from the database and displayed on-screen and the sale amount updated.

**Invariants:**

Incorrect barcode – allow user to cancel the sale or re-enter the barcode.

**Scenario:**

Enter barcode.

If barcode is in the database then

Retrieve product information

Display product information

Update sale amount

Else

Display error message.

End if

**Use case:**

Retrieve product ID

**Requirements:**

User has selected a product with a valid uom (unit-of-measurement) code.

**Pre-conditions:**

[1] User must be logged in and has accessed the point of sale transaction screen.

[2] User must have accessed the unit-sale screen to choose a product with a uom code.

[3] User must have access to an input device, a virtual/on-screen keyboard; or a physical keyboard.

**Post-condition:**

Product ID is returned and is used to search product information to be displayed on-screen and the sale amount updated.

**Invariants:**

**Scenario:**

Let user choose the product with a unit of measurement

Let user specify the quantity in relation to its unit of measurement

Retrieve product ID

If product ID exists in the database then

Retrieve product information

Display product information

Update sale amount

Else

Display error message

End if

**Use case:**

Payment

**Requirements:**

[1] There must be at least one product entry in the shopping basket.

[2] The cash amount tendered by the customer must be equal to or greater than the total sales amount.

**Pre-conditions:**

User must be logged-in, had accessed the point of sale transaction screen and has pressed the cash button indicating to the system that the user is ready to checkout.

**Post-conditions:**

[1] Payment is accepted.

[2] Sale transaction is saved to the database.

[3] Stock inventory is updated.

[4] Sales receipt is printed.

**Invariants:**

[1] Cash amount tendered by the customer is less than the total sales amount – allow user to re-enter cash tendered.

[2] Unable to print receipt – show error message, but continue the checkout process.

[3] Sale transaction has failed (updating the database has failed) – show error message, rollback changes to the database.

**Scenario:**

If shopping basket is not empty then

Get cash tendered

If cash tendered is equal to or greater than the total sales amount then

Display change if cash tendered is greater than the total sales amount

Update stock inventory

Save transaction to database

Print receipt

Else

Display error message

End If

Else

Display error message

End If

**Use case:**

Print receipt

**Requirements:**

A printer is connected to the point of sale device and is available and ready for printing.

**Pre-conditions:**

[1] Stock inventory is updated.

[2] Cash payment is received.

[3] Sale transaction is saved to the database.

**Post-conditions:**

A receipt is printed.

**Invariants:**

[1] Sale transaction has failed.

[2] No printer is connected or available for printing the receipt.

**Scenario:**

If sales transaction has been successfully committed then

If printer job is ready then

Send print job to printer

Else

Display error message

End If

Else

Display error message

End If

**Class Diagram: POS-Pi Application (with attributes and methods)**



Diagram created with ArgoUML (http://sourceforge.net/projects/argouml/)

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. [April 29, 2015].

**Class Diagram: POS-Pi Application (without attributes and methods)**

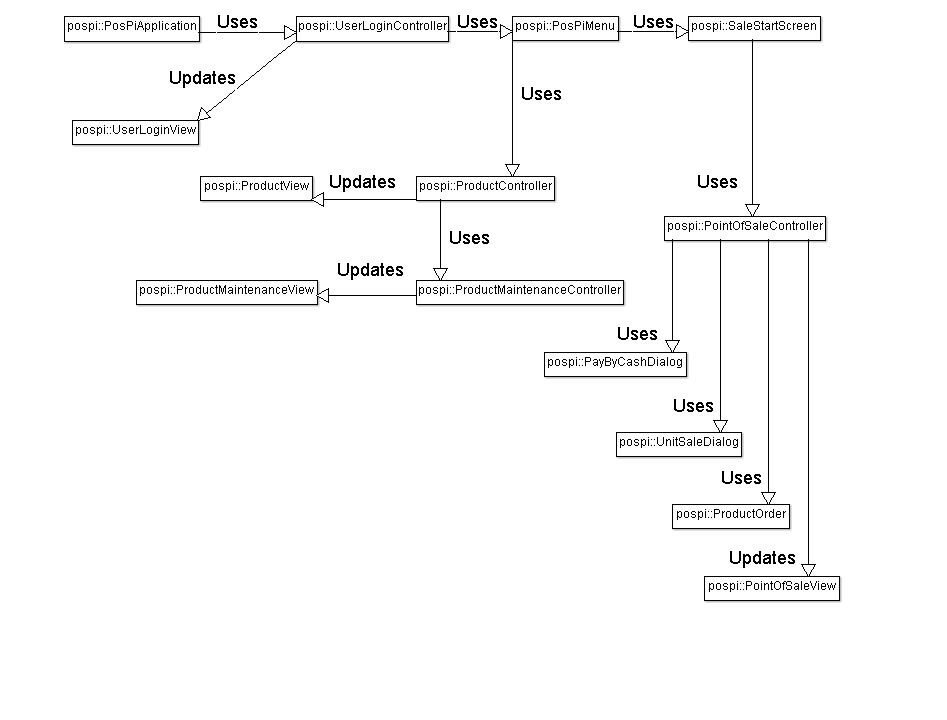
****

Diagram created with ArgoUML (<http://sourceforge.net/projects/argouml/>)

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. [April 29, 2015].

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Author note:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

The **POS-Pi** application is a Point-of-Sale application (fully working) prototype developed in Java. It is being developed using two Java IDEs: 1) eclipse for Windows and, 2) Bluej for the Raspberry Pi. I have to switch between the two ide during development for the following reasons:

1. I find the code-completion in eclipse to be an excellent feature, especially when I have to deal with libraries (Java Swing and JDBC) that were unfamiliar to me when I first started the project.

List of sources that I have used as a reference when developing using Java Swing and JDBC:

-Oracle Java Documentation: <http://docs.oracle.com/javase/tutorial/uiswing/>

-tutorialspoint Swing tutorial: <http://www.tutorialspoint.com/swing/>

-Package javax.swing: <http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

-GUI Swing, Custom Dialog: <http://javalessons.com/cgi-bin/ui/java-swing.cgi?1cd=odl&sid=ao789>

-JDialog: <http://www.java2s.com/Tutorial/Java/0240__Swing/1220__JDialog.htm>

-Lesson Printing: <http://docs.oracle.com/javase/tutorial/2d/printing/index.html>

-Oracle Java Trail JDBC: <http://docs.oracle.com/javase/tutorial/jdbc/>

-MKyong.com JDBC tutorial: <http://www.mkyong.com/tutorials/jdbc-tutorials/>

-MySQL Java Tutorial: <http://zetcode.com/db/mysqljava/>

-tutorialspoint JDBC tutorial: <http://www.tutorialspoint.com/jdbc/>

-Java MySQL Select example: <http://alvinalexander.com/java/java-mysql-select-query-example>

1. I have to use the Bluej IDE on the raspberry pi for testing purposes. Bluej on the raspberry runs fast and smooth compared to eclipse which is very slow and sluggish on the raspberry.

You will find a top level activity diagram on the next page that depicts a bird’s eye view of the entire workflow for the POS-Pi application. Followed by Javadoc documentation, sequence diagrams, flowcharts and source-code listings for the 14 classes that make up the POS-Pi application which I have based on a Model-View-Controller design pattern.

**Top Level Activity Diagram**

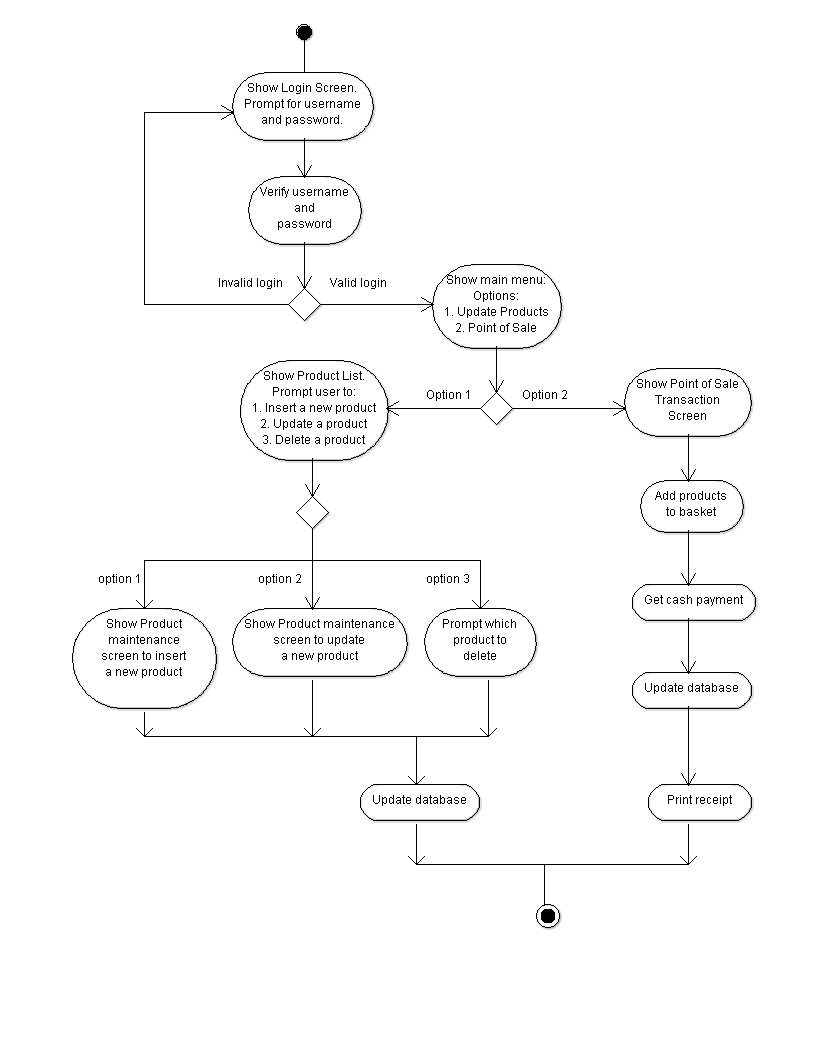
****

Diagram created with ArgoUML (<http://sourceforge.net/projects/argouml/>)

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. [April 29, 2015].

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class PosPiApplication**

java.lang.Object

**pospi.PosPiApplication**

public class **PosPiApplication**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: This class has the static main method which is the main entry point for the POS-Pi application.  
It's main task is to register the MySQL database driver and test for connection to the database.  
MySQL Connector/J is the official JDBC driver for MySQL. You can download it from this site:  
http://dev.mysql.com/downloads/connector/j/

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**PosPiApplication**](about:blank../pospi/PosPiApplication.html#PosPiApplication())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| static java.lang.String | [**getDataConnection**](about:blank../pospi/PosPiApplication.html#getDataConnection())() |
| static java.lang.String | [**getDataDriver**](about:blank../pospi/PosPiApplication.html#getDataDriver())() |
| static java.lang.String | [**getDataPassword**](about:blank../pospi/PosPiApplication.html#getDataPassword())() |
| static java.lang.String | [**getDataUser**](about:blank../pospi/PosPiApplication.html#getDataUser())() |
| static void | [**main**](about:blank../pospi/PosPiApplication.html#main(java.lang.String[]))(java.lang.String[] args)           Main entry point of the application. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### PosPiApplication

public **PosPiApplication**()

|  |
| --- |
| **Method Detail** |

### getDataConnection

public static java.lang.String **getDataConnection**()

### getDataDriver

public static java.lang.String **getDataDriver**()

### getDataPassword

public static java.lang.String **getDataPassword**()

### getDataUser

public static java.lang.String **getDataUser**()

### main

public static void **main**(java.lang.String[] args)

Main entry point of the application.  
[1] Registers the MySQL database connection driver  
[2] Test connection to MySQL database  
[3] Clean-up (closes) connection object

**Parameters:**

args[] - String

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

**Main(String[]) – Sequence Diagram**

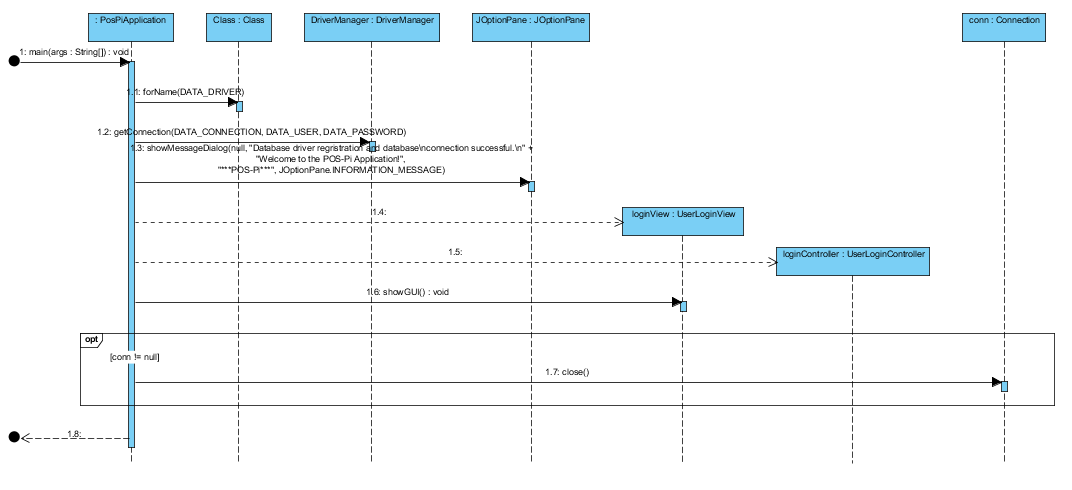
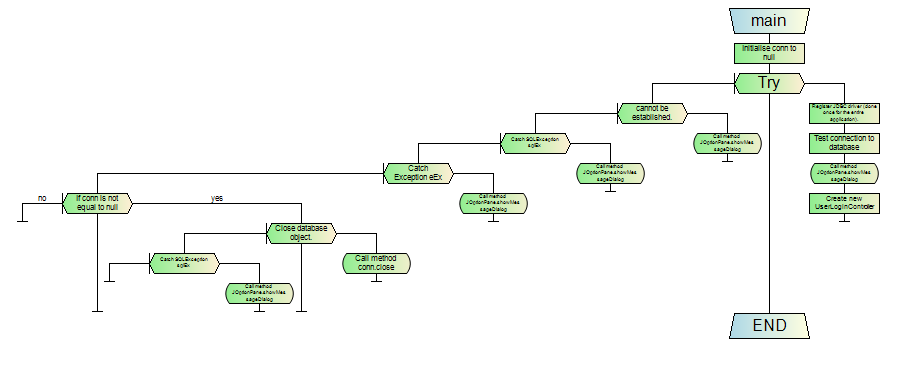


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**Main(String[]) – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**PosPiApplication class code listing:**

package pospi;

import javax.swing.\*;

import java.sql.\*;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: This class has the static main method which is the main entry point for the POS-Pi application.

\* <br>It's main task is to register the MySQL database driver and test for connection to the database.

\* <br>MySQL Connector/J is the official JDBC driver for MySQL. You can download it from this site:

\* <br>http://dev.mysql.com/downloads/connector/j/

\*\*/

public class PosPiApplication {

//MySQL driver

private static final String DATA\_DRIVER = "com.mysql.jdbc.Driver";

//Database URL

private static final String DATA\_CONNECTION = "jdbc:mysql://localhost/pos\_pi";

//User credentials for MySQL, user name and password

private static final String DATA\_USER = "root";

private static final String DATA\_PASSWORD = "raspberry";

/\*\*

\* Main entry point of the application.

\* <br>[1] Registers the MySQL database connection driver

\* <br>[2] Test connection to MySQL database

\* <br>[3] Clean-up (closes) connection object

\* @param args[] String

\*\*/

@SuppressWarnings("unused")

public static void main(String[] args) {

Connection conn = null;

try{

//Register JDBC driver (done once for the entire application).

Class.forName(DATA\_DRIVER);

//Test connection to database

conn = DriverManager.getConnection(DATA\_CONNECTION, DATA\_USER, DATA\_PASSWORD);

JOptionPane.showMessageDialog(null, "Database driver regristration and database\nconnection successful.\n" +

"Welcome to the POS-Pi Application!",

"\*\*\*POS-Pi\*\*\*", JOptionPane.INFORMATION\_MESSAGE);

UserLoginController loginController = new UserLoginController(new UserLoginView(), getDataConnection(),

getDataUser(), getDataPassword());

}//end try

/\*Catch errors if MySQL database driver registration is unsuccessful, or database connection

\* cannot be established.

\* \*/

catch(ClassNotFoundException cnfEx){

JOptionPane.showMessageDialog(null, cnfEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

catch(Exception eEx){

JOptionPane.showMessageDialog(null, eEx.toString(),

"Error in connecting to database",

JOptionPane.WARNING\_MESSAGE);

}//end catch

//perform clean-up. Close database connection

finally{

if(conn != null){

//Close database object.

try{

conn.close();

}//end try

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in closing the database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

}//end if

}//end finally

}//end main

/\*

\* Getters.

\* \*/

public static String getDataDriver() {

return DATA\_DRIVER;

}

public static String getDataConnection() {

return DATA\_CONNECTION;

}

public static String getDataUser() {

return DATA\_USER;

}

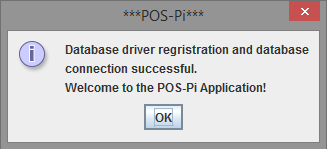
public static String getDataPassword() {

return DATA\_PASSWORD;

}

}//end class

**Screenshot:**



**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class UserLoginView**

java.lang.Object

**pospi.UserLoginView**

public class **UserLoginView**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: Constructs and initializes the components and graphical user interface of the user login screen.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**UserLoginView**](about:blank../pospi/UserLoginView.html#UserLoginView())()           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| javax.swing.JButton | [**getLoginButton**](about:blank../pospi/UserLoginView.html#getLoginButton())() |
| javax.swing.JFrame | [**getMainFrame**](about:blank../pospi/UserLoginView.html#getMainFrame())() |
| javax.swing.JTextField | [**getUserNameTextField**](about:blank../pospi/UserLoginView.html#getUserNameTextField())() |
| javax.swing.JPasswordField | [**getUserPasswordField**](about:blank../pospi/UserLoginView.html#getUserPasswordField())() |
| void | [**setLoginButton**](about:blank../pospi/UserLoginView.html#setLoginButton(javax.swing.JButton))(javax.swing.JButton loginButton) |
| void | [**setMainFrame**](about:blank../pospi/UserLoginView.html#setMainFrame(javax.swing.JFrame))(javax.swing.JFrame mainFrame) |
| void | [**setUserNameTextField**](about:blank../pospi/UserLoginView.html#setUserNameTextField(javax.swing.JTextField))(javax.swing.JTextField userNameTextField) |
| void | [**setUserPasswordField**](about:blank../pospi/UserLoginView.html#setUserPasswordField(javax.swing.JPasswordField))(javax.swing.JPasswordField userPasswordField) |
| void | [**showGUI**](about:blank../pospi/UserLoginView.html#showGUI())()           Makes the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### UserLoginView

public **UserLoginView**()

Constructor.

|  |
| --- |
| **Method Detail** |

### getLoginButton

public javax.swing.JButton **getLoginButton**()

### getMainFrame

public javax.swing.JFrame **getMainFrame**()

### getUserNameTextField

public javax.swing.JTextField **getUserNameTextField**()

### getUserPasswordField

public javax.swing.JPasswordField **getUserPasswordField**()

### setLoginButton

public void **setLoginButton**(javax.swing.JButton loginButton)

### setMainFrame

public void **setMainFrame**(javax.swing.JFrame mainFrame)

### setUserNameTextField

public void **setUserNameTextField**(javax.swing.JTextField userNameTextField)

### setUserPasswordField

public void **setUserPasswordField**(javax.swing.JPasswordField userPasswordField)

### showGUI

public void **showGUI**()

Makes the frame visible.

**UserLoginView class code listing:**

package pospi;

import javax.swing.JFrame;

import java.awt.BorderLayout;

import javax.swing.JPanel;

import javax.swing.JButton;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JLabel;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import java.awt.GridBagLayout;

import java.awt.GridBagConstraints;

import java.awt.Insets;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import javax.swing.SwingConstants;

import java.awt.Font;

import java.awt.Color;

import java.util.Date;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Constructs and initializes the components and graphical user interface

\* of the user login screen.

\*\*/

@SuppressWarnings("unused")

public class UserLoginView {

private JFrame mainFrame;

private JTextField userNameTextField;

private JPasswordField userPasswordField;

private JButton loginButton;

/\*\*

\* Constructor.

\*/

public UserLoginView() {

initialize();

}

/\*\*

\* Initialize the gui components of the frame.

\*/

private void initialize() {

mainFrame = new JFrame();

mainFrame.setBounds(100, 100, 291, 171);

mainFrame.getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

panel.setBackground(Color.LIGHT\_GRAY);

mainFrame.getContentPane().add(panel, BorderLayout.SOUTH);

loginButton = new JButton("Login");

panel.add(loginButton);

JPanel panel\_1 = new JPanel();

mainFrame.getContentPane().add(panel\_1, BorderLayout.CENTER);

JLabel lblNewLabel\_1 = new JLabel("Username:");

JLabel lblNewLabel\_2 = new JLabel("Password:");

userNameTextField = new JTextField();

userNameTextField.setColumns(10);

userPasswordField = new JPasswordField();

JLabel titleLabel = new JLabel("POS-Pi");

titleLabel.setForeground(Color.WHITE);

titleLabel.setBackground(Color.BLACK);

titleLabel.setHorizontalAlignment(SwingConstants.CENTER);

titleLabel.setOpaque(true);

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGap(23)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel\_2)

.addPreferredGap(ComponentPlacement.UNRELATED)

.addComponent(userPasswordField))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel\_1)

.addPreferredGap(ComponentPlacement.UNRELATED)

.addComponent(userNameTextField, GroupLayout.PREFERRED\_SIZE, 142, GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(55, Short.MAX\_VALUE))

.addComponent(titleLabel, GroupLayout.DEFAULT\_SIZE, 282, Short.MAX\_VALUE)

);

gl\_panel\_1.setVerticalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(titleLabel, GroupLayout.PREFERRED\_SIZE, 25, GroupLayout.PREFERRED\_SIZE)

.addGap(12)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(lblNewLabel\_1)

.addComponent(userNameTextField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(lblNewLabel\_2)

.addComponent(userPasswordField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

.addContainerGap(19, Short.MAX\_VALUE))

);

panel\_1.setLayout(gl\_panel\_1);

mainFrame.getRootPane().setDefaultButton(loginButton);

}//end initialize().

//Getters.

public JFrame getMainFrame() {

return mainFrame;

}

public JTextField getUserNameTextField() {

return userNameTextField;

}

public JPasswordField getUserPasswordField() {

return userPasswordField;

}

public JButton getLoginButton() {

return loginButton;

}

//Setters.

public void setMainFrame(JFrame mainFrame) {

this.mainFrame = mainFrame;

}

public void setUserNameTextField(JTextField userNameTextField) {

this.userNameTextField = userNameTextField;

}

public void setUserPasswordField(JPasswordField userPasswordField) {

this.userPasswordField = userPasswordField;

}

public void setLoginButton(JButton loginButton) {

this.loginButton = loginButton;

}

/\*\*

\* Makes the frame visible.

\*/

public void showGUI(){

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.pack();

mainFrame.setLocationRelativeTo(null);

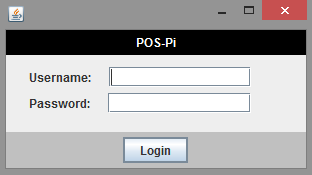
this.mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

mainFrame.setVisible(true);

}//end showGUI

}//end class.

**Screenshot:**



**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class UserLoginController**

java.lang.Object

**pospi.UserLoginController**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

public class **UserLoginController**extends java.lang.Objectimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description: Handles the logic and events of the UserLoginView class.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**UserLoginController**](about:blank../pospi/UserLoginController.html#UserLoginController(pospi.UserLoginView, java.lang.String, java.lang.String, java.lang.String))(pospi.UserLoginView loginView, java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/UserLoginController.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. Handles the events when a button is pressed. |
| java.lang.String | [**getDataConnection**](about:blank../pospi/UserLoginController.html#getDataConnection())() |
| java.lang.String | [**getDataPassword**](about:blank../pospi/UserLoginController.html#getDataPassword())() |
| java.lang.String | [**getDataUser**](about:blank../pospi/UserLoginController.html#getDataUser())() |
| pospi.UserLoginView | [**getLoginView**](about:blank../pospi/UserLoginController.html#getLoginView())() |
| void | [**initialize**](about:blank../pospi/UserLoginController.html#initialize())()           Initialize component's event-listeners. |
| boolean | [**isUserValid**](about:blank../pospi/UserLoginController.html#isUserValid(java.lang.String, char[]))(java.lang.String userName, char[] password)           Checks the database if username and password is valid. |
| void | [**setDataConnection**](about:blank../pospi/UserLoginController.html#setDataConnection(java.lang.String))(java.lang.String dataConnection) |
| void | [**setDataPassword**](about:blank../pospi/UserLoginController.html#setDataPassword(java.lang.String))(java.lang.String dataPassword) |
| void | [**setDataUser**](about:blank../pospi/UserLoginController.html#setDataUser(java.lang.String))(java.lang.String dataUser) |
| void | [**setLoginView**](about:blank../pospi/UserLoginController.html#setLoginView(pospi.UserLoginView))(pospi.UserLoginView loginView) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### UserLoginController

public **UserLoginController**(pospi.UserLoginView loginView,

java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor.

**Parameters:**

loginView - UserLoginView

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### getDataConnection

public java.lang.String **getDataConnection**()

### getDataPassword

public java.lang.String **getDataPassword**()

### getDataUser

public java.lang.String **getDataUser**()

### getLoginView

public pospi.UserLoginView **getLoginView**()

### initialize

public void **initialize**()

Initialize component's event-listeners.

### isUserValid

public boolean **isUserValid**(java.lang.String userName,

char[] password)

Checks the database if username and password is valid.

**Parameters:**

userName - String

password - char[]

**Returns:**

boolean

### setDataConnection

public void **setDataConnection**(java.lang.String dataConnection)

### setDataPassword

public void **setDataPassword**(java.lang.String dataPassword)

### setDataUser

public void **setDataUser**(java.lang.String dataUser)

### setLoginView

public void **setLoginView**(pospi.UserLoginView loginView)

**isUserValid(String usernName, char[] password) – Sequence Diagram**

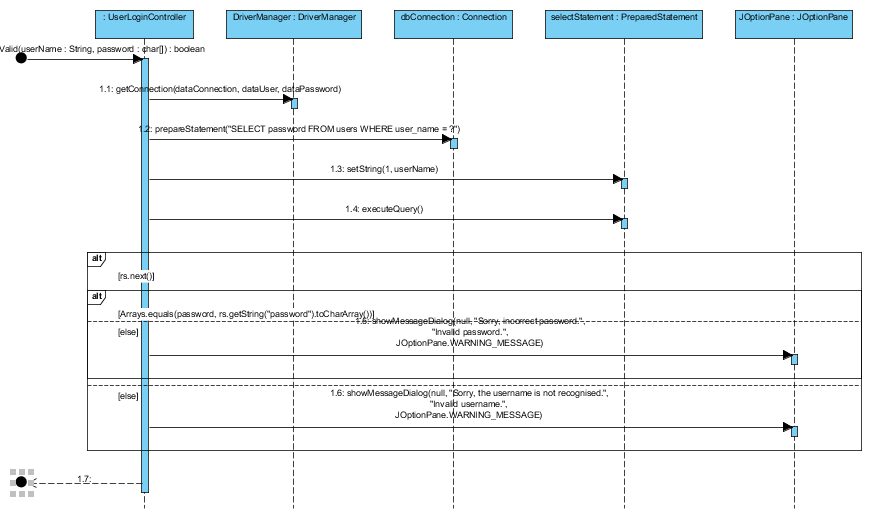
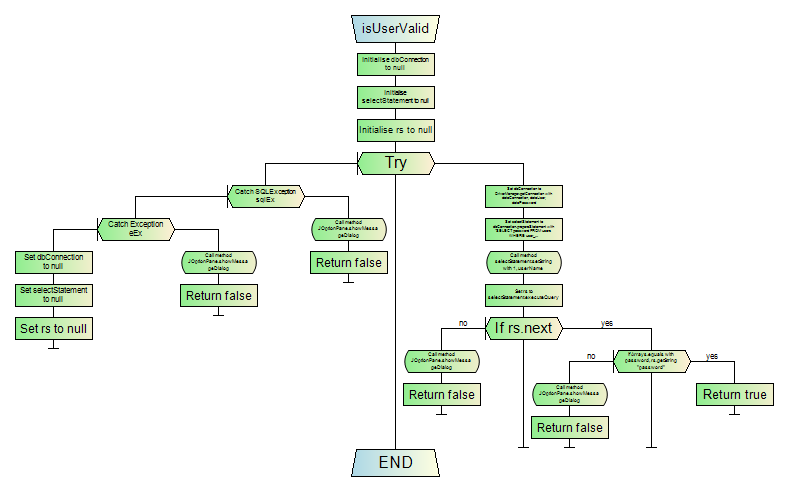


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**isUserValid(String usernName, char[] password) – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**UserLoginController class code listing:**

package pospi;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.sql.\*;

import java.util.Arrays;

import javax.swing.JOptionPane;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events of the UserLoginView class.

\* \*/

public class UserLoginController implements ActionListener{

private UserLoginView loginView;

private String dataConnection;

private String dataUser;

private String dataPassword;

/\*\*

\* Constructor.

\* @param loginView UserLoginView

\* @param connection String

\* @param user String

\* @param password String

\* \*/

public UserLoginController(UserLoginView loginView, String connection, String user, String password){

this.loginView = loginView;

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

initialize();

this.loginView.showGUI();

}

//Getters.

public UserLoginView getLoginView() {

return loginView;

}

public String getDataConnection() {

return dataConnection;

}

public String getDataUser() {

return dataUser;

}

public String getDataPassword() {

return dataPassword;

}

//Setters.

public void setLoginView(UserLoginView loginView) {

this.loginView = loginView;

}

public void setDataConnection(String dataConnection) {

this.dataConnection = dataConnection;

}

public void setDataUser(String dataUser) {

this.dataUser = dataUser;

}

public void setDataPassword(String dataPassword) {

this.dataPassword = dataPassword;

}

/\*\*

\* Initialize component's event-listeners.

\*/

public void initialize(){

this.loginView.getLoginButton().addActionListener(this);

this.loginView.getLoginButton().setActionCommand("login");

}//end initialize().

/\*\*

\* Implemented ActionListener abstract methods.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

if(command == "login"){

if(isUserValid(loginView.getUserNameTextField().getText(),

loginView.getUserPasswordField().getPassword())){

PosPiMenu posMenu = new PosPiMenu(this.dataConnection, this.dataUser, this.dataPassword);

posMenu.showGUI();

this.loginView.getMainFrame().dispose();

}

}//end if

}//end actionPerformed().

/\*\*

\* Checks the database if username and password is valid.

\* @param userName String

\* @param password char[]

\* @return boolean

\*/

public boolean isUserValid(String userName, char[] password){

Connection dbConnection = null;

PreparedStatement selectStatement = null;

ResultSet rs = null;

try{

dbConnection = DriverManager.getConnection(dataConnection, dataUser, dataPassword);

selectStatement = dbConnection.prepareStatement("SELECT password FROM users WHERE user\_name = ?");

selectStatement.setString(1, userName);

rs = selectStatement.executeQuery();

if(rs.next()){

if(Arrays.equals(password, rs.getString("password").toCharArray())){

return true;

}

else{

JOptionPane.showMessageDialog(null, "Sorry, incorrect password.",

"Invalid password.",

JOptionPane.WARNING\_MESSAGE);

return false;

}//end if

}

else{

JOptionPane.showMessageDialog(null, "Sorry, the username is not recognised.",

"Invalid username.",

JOptionPane.WARNING\_MESSAGE);

return false;

}//end if

}//end try

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

return false;

}//end catch

catch(Exception eEx){

JOptionPane.showMessageDialog(null, eEx.toString(),

"Error in connecting to database",

JOptionPane.WARNING\_MESSAGE);

return false;

}//end catch

finally{

dbConnection = null;

selectStatement = null;

rs = null;

}//end finally

}//end isUserValid()

}//end class

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class PosPiMenu**

java.lang.Object

**pospi.PosPiMenu**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

public class **PosPiMenu**extends java.lang.Objectimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description: Handles the logic and events, constructs and initializes the components and graphical user interface of the PosPiMenu class.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**PosPiMenu**](about:blank../pospi/PosPiMenu.html#PosPiMenu(java.lang.String, java.lang.String, java.lang.String))(java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/PosPiMenu.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. Handles the events when a button is pressed. |
| void | [**showGUI**](about:blank../pospi/PosPiMenu.html#showGUI())()           Makes the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### PosPiMenu

public **PosPiMenu**(java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor.

**Parameters:**

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### showGUI

public void **showGUI**()

Makes the frame visible.

**actionPerformed(ActionEvent) – Sequence Diagram**

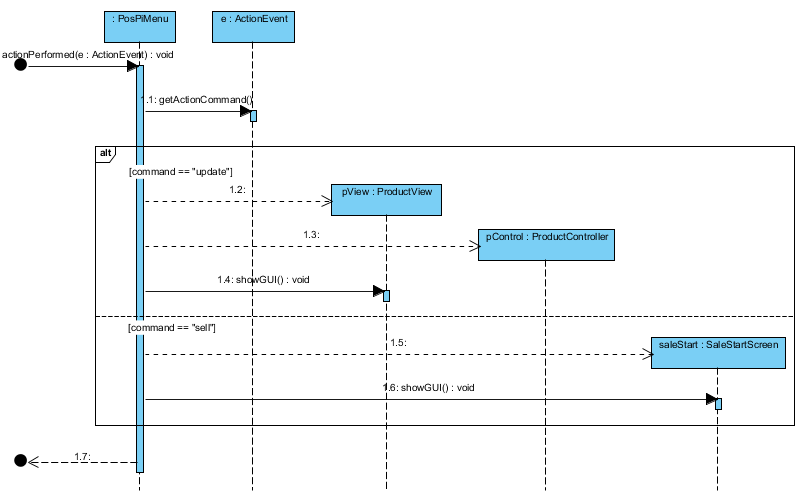
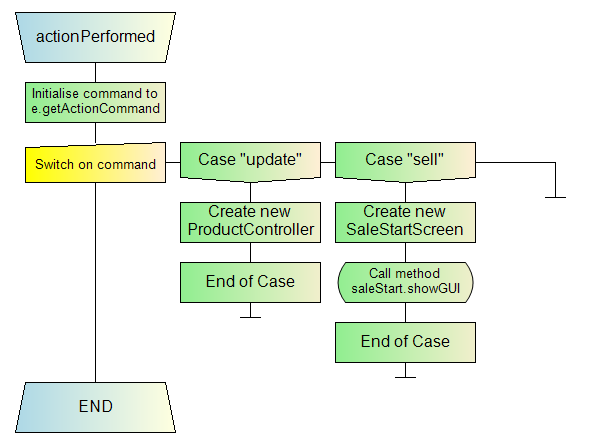


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

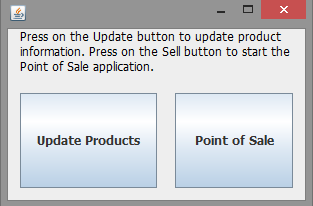
**actionPerformed(ActionEvent) – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**Screenshot:**



**PosPiMenu class code listing:**

package pospi;

import javax.swing.JFrame;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JPanel;

import javax.swing.JButton;

import javax.swing.JTextArea;

import java.awt.BorderLayout;

import java.awt.Font;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.UIManager;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events, constructs and initializes the components and graphical user interface

\* of the PosPiMenu class.

\* \*/

public class PosPiMenu implements ActionListener{

private String dataConnection;

private String dataUser;

private String dataPassword;

private JFrame mainFrame;

private JButton sellButton;

private JButton updateButton;

/\*\*

\* Constructor.

\* @param connection String

\* @param user String

\* @param password String

\*/

public PosPiMenu(String connection, String user, String password) {

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

initialize();

}

/\*\*

\* Initialize the components and gui of the frame.

\*/

private void initialize() {

mainFrame = new JFrame();

mainFrame.getContentPane().setBackground(UIManager.getColor("Button.background"));

mainFrame.setBounds(100, 100, 282, 204);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

JPanel panel\_1 = new JPanel();

JPanel panel\_2 = new JPanel();

GroupLayout groupLayout = new GroupLayout(mainFrame.getContentPane());

groupLayout.setHorizontalGroup(

groupLayout.createParallelGroup(Alignment.TRAILING)

.addGroup(groupLayout.createSequentialGroup()

.addContainerGap()

.addGroup(groupLayout.createParallelGroup(Alignment.TRAILING)

.addComponent(panel\_2, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 246, Short.MAX\_VALUE)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(panel, GroupLayout.DEFAULT\_SIZE, 110, Short.MAX\_VALUE)

.addGap(18)

.addComponent(panel\_1, GroupLayout.DEFAULT\_SIZE, 118, Short.MAX\_VALUE)))

.addContainerGap())

);

groupLayout.setVerticalGroup(

groupLayout.createParallelGroup(Alignment.TRAILING)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(panel\_2, GroupLayout.PREFERRED\_SIZE, 58, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(groupLayout.createParallelGroup(Alignment.TRAILING)

.addComponent(panel\_1, GroupLayout.DEFAULT\_SIZE, 95, Short.MAX\_VALUE)

.addComponent(panel, GroupLayout.DEFAULT\_SIZE, 90, Short.MAX\_VALUE))

.addContainerGap())

);

panel\_2.setLayout(new BorderLayout(0, 0));

JTextArea textArea = new JTextArea();

textArea.setForeground(Color.BLACK);

textArea.setWrapStyleWord(true);

textArea.setText("Press on the Update button to update product information. Press on the Sell button to start the Point of Sale application.");

textArea.setLineWrap(true);

textArea.setFont(new Font("Tahoma", Font.PLAIN, 12));

textArea.setEditable(false);

textArea.setBackground(UIManager.getColor("Button.background"));

panel\_2.add(textArea, BorderLayout.CENTER);

panel\_1.setLayout(new BorderLayout(0, 0));

//sellButton = new JButton("Sell");

sellButton = new JButton("Point of Sale");

sellButton.setFont(new Font("Tahoma", Font.BOLD, 12));

sellButton.addActionListener(this);

sellButton.setActionCommand("sell");

panel\_1.add(sellButton, BorderLayout.CENTER);

panel.setLayout(new BorderLayout(0, 0));

//updateButton = new JButton("Update");

updateButton = new JButton("Update Products");

updateButton.setFont(new Font("Tahoma", Font.BOLD, 12));

updateButton.addActionListener(this);

updateButton.setActionCommand("update");

panel.add(updateButton, BorderLayout.CENTER);

mainFrame.getContentPane().setLayout(groupLayout);

}//end initialize.

/\*\*

\* Makes the frame visible.

\*/

public void showGUI(){

//mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

mainFrame.pack();

mainFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

mainFrame.setLocationRelativeTo(null);

mainFrame.setVisible(true);

//mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

}//end showGUI.

/\*\*

\* Implemented ActionListener abstract methods.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

switch(command){

case "update":

@SuppressWarnings("unused")

ProductController pControl = new ProductController(new ProductView(), this.dataConnection, this.dataUser, this.dataPassword);

break;

case "sell":

SaleStartScreen saleStart = new SaleStartScreen(this.dataConnection, this.dataUser, this.dataPassword);

saleStart.showGUI();

break;

}

}//end actionPerformed().

}//end class.

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. computer program. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class SaleStartScreen**

java.lang.Object

**pospi.SaleStartScreen**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

public class **SaleStartScreen**extends java.lang.Objectimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description: Handles the logic and events, constructs and initializes the components and graphical user interface of the SaleStartScreen class.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**SaleStartScreen**](about:blank../pospi/SaleStartScreen.html#SaleStartScreen(java.lang.String, java.lang.String, java.lang.String))(java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/SaleStartScreen.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. Handles the events when a button is pressed. |
| java.lang.String | [**getDataConnection**](about:blank../pospi/SaleStartScreen.html#getDataConnection())() |
| java.lang.String | [**getDataPassword**](about:blank../pospi/SaleStartScreen.html#getDataPassword())() |
| java.lang.String | [**getDataUser**](about:blank../pospi/SaleStartScreen.html#getDataUser())() |
| javax.swing.JFrame | [**getMainFrame**](about:blank../pospi/SaleStartScreen.html#getMainFrame())() |
| javax.swing.JButton | [**getStartSaleButton**](about:blank../pospi/SaleStartScreen.html#getStartSaleButton())() |
| void | [**setDataConnection**](about:blank../pospi/SaleStartScreen.html#setDataConnection(java.lang.String))(java.lang.String dataConnection) |
| void | [**setDataPassword**](about:blank../pospi/SaleStartScreen.html#setDataPassword(java.lang.String))(java.lang.String dataPassword) |
| void | [**setDataUser**](about:blank../pospi/SaleStartScreen.html#setDataUser(java.lang.String))(java.lang.String dataUser) |
| void | [**setMainFrame**](about:blank../pospi/SaleStartScreen.html#setMainFrame(javax.swing.JFrame))(javax.swing.JFrame frmPospi) |
| void | [**setStartSaleButton**](about:blank../pospi/SaleStartScreen.html#setStartSaleButton(javax.swing.JButton))(javax.swing.JButton startSaleButton) |
| void | [**showGUI**](about:blank../pospi/SaleStartScreen.html#showGUI())()           Makes the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### SaleStartScreen

public **SaleStartScreen**(java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor.

**Parameters:**

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### getDataConnection

public java.lang.String **getDataConnection**()

### getDataPassword

public java.lang.String **getDataPassword**()

### getDataUser

public java.lang.String **getDataUser**()

### getMainFrame

public javax.swing.JFrame **getMainFrame**()

### getStartSaleButton

public javax.swing.JButton **getStartSaleButton**()

### setDataConnection

public void **setDataConnection**(java.lang.String dataConnection)

### setDataPassword

public void **setDataPassword**(java.lang.String dataPassword)

### setDataUser

public void **setDataUser**(java.lang.String dataUser)

### setMainFrame

public void **setMainFrame**(javax.swing.JFrame frmPospi)

### setStartSaleButton

public void **setStartSaleButton**(javax.swing.JButton startSaleButton)

### showGUI

public void **showGUI**()

Makes the frame visible.

**actionPerformed(ActionEvent) – Sequence Diagram**

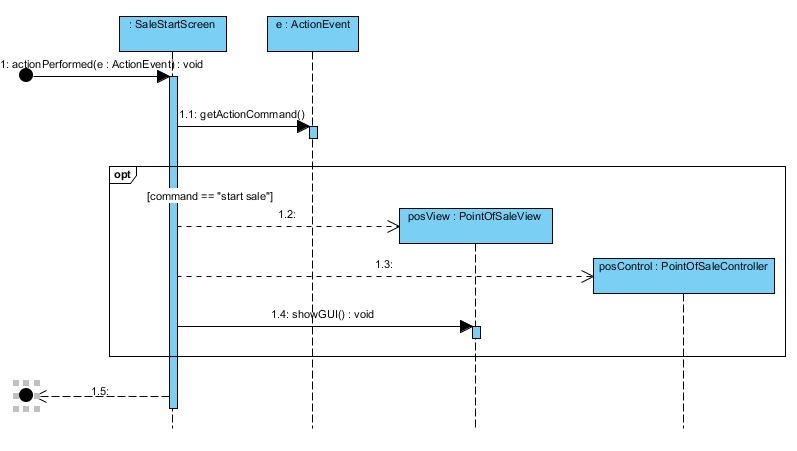
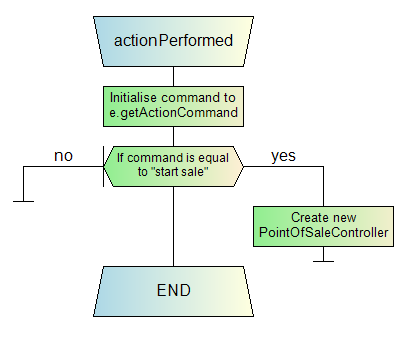


Diagram created with Visual Paradigm (http://www.visual-paradigm.com/)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

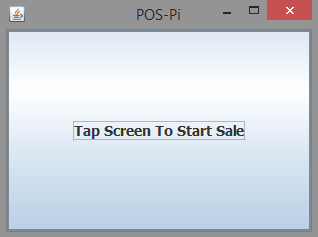
**actionPerformed(ActionEvent) – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**Screenshot:**



**SaleStartScreen class code listing:**

package pospi;

import javax.swing.JFrame;

import java.awt.BorderLayout;

import javax.swing.JButton;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events, constructs and initializes the components and graphical user interface

\* of the SaleStartScreen class.

\* \*/

public class SaleStartScreen implements ActionListener{

private String dataConnection;

private String dataUser;

private String dataPassword;

private JFrame mainFrame;

private JButton startSaleButton;

/\*\*

\* Constructor.

\* @param connection String

\* @param user String

\* @param password String

\*/

public SaleStartScreen(String connection, String user, String password){

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

initialize();

}//end SaleStartScreen.

/\*\*

\* Initialize the gui components of the frame.

\*/

private void initialize() {

mainFrame = new JFrame();

mainFrame.setTitle("POS-Pi");

mainFrame.setBounds(100, 100, 320, 240);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.getContentPane().setLayout(new BorderLayout(0, 0));

startSaleButton = new JButton("Tap Screen To Start Sale");

startSaleButton.setFont(new Font("Tahoma", Font.BOLD, 14));

startSaleButton.addActionListener(this);

startSaleButton.setActionCommand("start sale");

mainFrame.getContentPane().add(startSaleButton, BorderLayout.CENTER);

mainFrame.getRootPane().setDefaultButton(startSaleButton);

}//end initialize().

/\*\*

\* Implemented ActionListener abstract methods.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

if(command == "start sale"){

@SuppressWarnings("unused")

PointOfSaleController posControl = new PointOfSaleController(new PointOfSaleView(), this.dataConnection,

this.dataUser, this.dataPassword);

}//end if

}//end actionPerformed();

/\*\*

\* Makes the frame visible.

\*/

public void showGUI(){

mainFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

mainFrame.setLocationRelativeTo(null);;

mainFrame.setVisible(true);

mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

}//end showGUI

//Getters.

public String getDataConnection() {

return dataConnection;

}

public String getDataUser() {

return dataUser;

}

public String getDataPassword() {

return dataPassword;

}

public JFrame getMainFrame() {

return mainFrame;

}

public JButton getStartSaleButton() {

return startSaleButton;

}

//Setters.

public void setDataConnection(String dataConnection) {

this.dataConnection = dataConnection;

}

public void setDataUser(String dataUser) {

this.dataUser = dataUser;

}

public void setDataPassword(String dataPassword) {

this.dataPassword = dataPassword;

}

public void setMainFrame(JFrame frmPospi) {

this.mainFrame = frmPospi;

}

public void setStartSaleButton(JButton startSaleButton) {

this.startSaleButton = startSaleButton;

}

}//end class

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class PointOfSaleView**

java.lang.Object

**pospi.PointOfSaleView**

public class **PointOfSaleView**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: Constructs and initializes the components and graphical user interface of the pos-pi sales screen

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**PointOfSaleView**](about:blank../pospi/PointOfSaleView.html#PointOfSaleView())()           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| javax.swing.JTextField | [**getBarcodeTextField**](about:blank../pospi/PointOfSaleView.html#getBarcodeTextField())() |
| javax.swing.JButton | [**getCancelItemButton**](about:blank../pospi/PointOfSaleView.html#getCancelItemButton())() |
| javax.swing.JButton | [**getCancelSaleButton**](about:blank../pospi/PointOfSaleView.html#getCancelSaleButton())() |
| javax.swing.JButton | [**getCardButton**](about:blank../pospi/PointOfSaleView.html#getCardButton())() |
| javax.swing.JButton | [**getCashButton**](about:blank../pospi/PointOfSaleView.html#getCashButton())() |
| javax.swing.JButton | [**getEightButton**](about:blank../pospi/PointOfSaleView.html#getEightButton())() |
| javax.swing.JButton | [**getEnterBarcodeButton**](about:blank../pospi/PointOfSaleView.html#getEnterBarcodeButton())() |
| javax.swing.JButton | [**getFiveButton**](about:blank../pospi/PointOfSaleView.html#getFiveButton())() |
| java.lang.String | [**getFormattedDate**](about:blank../pospi/PointOfSaleView.html#getFormattedDate(java.util.Date))(java.util.Date date)           Returns the formatted date as a String using the MM-DD-YYYY format. |
| javax.swing.JButton | [**getFourButton**](about:blank../pospi/PointOfSaleView.html#getFourButton())() |
| javax.swing.JTextArea | [**getItemDisplayTextArea**](about:blank../pospi/PointOfSaleView.html#getItemDisplayTextArea())() |
| javax.swing.JFrame | [**getMainFrame**](about:blank../pospi/PointOfSaleView.html#getMainFrame())() |
| javax.swing.JButton | [**getNineButton**](about:blank../pospi/PointOfSaleView.html#getNineButton())() |
| javax.swing.JButton | [**getOneButton**](about:blank../pospi/PointOfSaleView.html#getOneButton())() |
| javax.swing.JButton | [**getOtherItemButton**](about:blank../pospi/PointOfSaleView.html#getOtherItemButton())() |
| javax.swing.JScrollPane | [**getScrollPane**](about:blank../pospi/PointOfSaleView.html#getScrollPane())() |
| javax.swing.JButton | [**getSevenButton**](about:blank../pospi/PointOfSaleView.html#getSevenButton())() |
| javax.swing.JButton | [**getSixButton**](about:blank../pospi/PointOfSaleView.html#getSixButton())() |
| javax.swing.JButton | [**getThreeButton**](about:blank../pospi/PointOfSaleView.html#getThreeButton())() |
| javax.swing.JTextField | [**getTotalTextField**](about:blank../pospi/PointOfSaleView.html#getTotalTextField())() |
| javax.swing.JButton | [**getTwoButton**](about:blank../pospi/PointOfSaleView.html#getTwoButton())() |
| javax.swing.JButton | [**getxButton**](about:blank../pospi/PointOfSaleView.html#getxButton())() |
| javax.swing.JButton | [**getZeroButton**](about:blank../pospi/PointOfSaleView.html#getZeroButton())() |
| void | [**setBarcodeTextField**](about:blank../pospi/PointOfSaleView.html#setBarcodeTextField(javax.swing.JTextField))(javax.swing.JTextField barcodeTextField) |
| void | [**setCancelItemButton**](about:blank../pospi/PointOfSaleView.html#setCancelItemButton(javax.swing.JButton))(javax.swing.JButton cancelItemButton) |
| void | [**setCancelSaleButton**](about:blank../pospi/PointOfSaleView.html#setCancelSaleButton(javax.swing.JButton))(javax.swing.JButton cancelSaleButton) |
| void | [**setCardButton**](about:blank../pospi/PointOfSaleView.html#setCardButton(javax.swing.JButton))(javax.swing.JButton cardButton) |
| void | [**setCashButton**](about:blank../pospi/PointOfSaleView.html#setCashButton(javax.swing.JButton))(javax.swing.JButton cashButton) |
| void | [**setEightButton**](about:blank../pospi/PointOfSaleView.html#setEightButton(javax.swing.JButton))(javax.swing.JButton eightButton) |
| void | [**setEnterBarcodeButton**](about:blank../pospi/PointOfSaleView.html#setEnterBarcodeButton(javax.swing.JButton))(javax.swing.JButton backSpaceButton) |
| void | [**setFiveButton**](about:blank../pospi/PointOfSaleView.html#setFiveButton(javax.swing.JButton))(javax.swing.JButton fiveButton) |
| void | [**setFourButton**](about:blank../pospi/PointOfSaleView.html#setFourButton(javax.swing.JButton))(javax.swing.JButton fourButton) |
| void | [**setItemDisplayTextArea**](about:blank../pospi/PointOfSaleView.html#setItemDisplayTextArea(javax.swing.JTextArea))(javax.swing.JTextArea itemDisplayTextArea) |
| void | [**setMainFrame**](about:blank../pospi/PointOfSaleView.html#setMainFrame(javax.swing.JFrame))(javax.swing.JFrame mainFrame) |
| void | [**setNineButton**](about:blank../pospi/PointOfSaleView.html#setNineButton(javax.swing.JButton))(javax.swing.JButton nineButton) |
| void | [**setOneButton**](about:blank../pospi/PointOfSaleView.html#setOneButton(javax.swing.JButton))(javax.swing.JButton oneButton) |
| void | [**setOtherItemButton**](about:blank../pospi/PointOfSaleView.html#setOtherItemButton(javax.swing.JButton))(javax.swing.JButton otherItemButton) |
| void | [**setScrollPane**](about:blank../pospi/PointOfSaleView.html#setScrollPane(javax.swing.JScrollPane))(javax.swing.JScrollPane scrollPane) |
| void | [**setSevenButton**](about:blank../pospi/PointOfSaleView.html#setSevenButton(javax.swing.JButton))(javax.swing.JButton sevenButton) |
| void | [**setSixButton**](about:blank../pospi/PointOfSaleView.html#setSixButton(javax.swing.JButton))(javax.swing.JButton sixButton) |
| void | [**setThreeButton**](about:blank../pospi/PointOfSaleView.html#setThreeButton(javax.swing.JButton))(javax.swing.JButton threeButton) |
| void | [**setTotalTextField**](about:blank../pospi/PointOfSaleView.html#setTotalTextField(javax.swing.JTextField))(javax.swing.JTextField totalTextField) |
| void | [**setTwoButton**](about:blank../pospi/PointOfSaleView.html#setTwoButton(javax.swing.JButton))(javax.swing.JButton twoButton) |
| void | [**setxButton**](about:blank../pospi/PointOfSaleView.html#setxButton(javax.swing.JButton))(javax.swing.JButton xButton) |
| void | [**setZeroButton**](about:blank../pospi/PointOfSaleView.html#setZeroButton(javax.swing.JButton))(javax.swing.JButton zeroButton) |
| void | [**showGUI**](about:blank../pospi/PointOfSaleView.html#showGUI())()           Makes the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### PointOfSaleView

public **PointOfSaleView**()

Constructor.

|  |
| --- |
| **Method Detail** |

### getBarcodeTextField

public javax.swing.JTextField **getBarcodeTextField**()

### getCancelItemButton

public javax.swing.JButton **getCancelItemButton**()

### getCancelSaleButton

public javax.swing.JButton **getCancelSaleButton**()

### getCardButton

public javax.swing.JButton **getCardButton**()

### getCashButton

public javax.swing.JButton **getCashButton**()

### getEightButton

public javax.swing.JButton **getEightButton**()

### getEnterBarcodeButton

public javax.swing.JButton **getEnterBarcodeButton**()

### getFiveButton

public javax.swing.JButton **getFiveButton**()

### getFormattedDate

public java.lang.String **getFormattedDate**(java.util.Date date)

Returns the formatted date as a String using the MM-DD-YYYY format.

**Parameters:**

date - Date

**Returns:**

String

### getFourButton

public javax.swing.JButton **getFourButton**()

### getItemDisplayTextArea

public javax.swing.JTextArea **getItemDisplayTextArea**()

### getMainFrame

public javax.swing.JFrame **getMainFrame**()

### getNineButton

public javax.swing.JButton **getNineButton**()

### getOneButton

public javax.swing.JButton **getOneButton**()

### getOtherItemButton

public javax.swing.JButton **getOtherItemButton**()

### getScrollPane

public javax.swing.JScrollPane **getScrollPane**()

### getSevenButton

public javax.swing.JButton **getSevenButton**()

### getSixButton

public javax.swing.JButton **getSixButton**()

### getThreeButton

public javax.swing.JButton **getThreeButton**()

### getTotalTextField

public javax.swing.JTextField **getTotalTextField**()

### getTwoButton

public javax.swing.JButton **getTwoButton**()

### getxButton

public javax.swing.JButton **getxButton**()

### getZeroButton

public javax.swing.JButton **getZeroButton**()

### setBarcodeTextField

public void **setBarcodeTextField**(javax.swing.JTextField barcodeTextField)

### setCancelItemButton

public void **setCancelItemButton**(javax.swing.JButton cancelItemButton)

### setCancelSaleButton

public void **setCancelSaleButton**(javax.swing.JButton cancelSaleButton)

### setCardButton

public void **setCardButton**(javax.swing.JButton cardButton)

### setCashButton

public void **setCashButton**(javax.swing.JButton cashButton)

### setEightButton

public void **setEightButton**(javax.swing.JButton eightButton)

### setEnterBarcodeButton

public void **setEnterBarcodeButton**(javax.swing.JButton backSpaceButton)

### setFiveButton

public void **setFiveButton**(javax.swing.JButton fiveButton)

### setFourButton

public void **setFourButton**(javax.swing.JButton fourButton)

### setItemDisplayTextArea

public void **setItemDisplayTextArea**(javax.swing.JTextArea itemDisplayTextArea)

### setMainFrame

public void **setMainFrame**(javax.swing.JFrame mainFrame)

### setNineButton

public void **setNineButton**(javax.swing.JButton nineButton)

### setOneButton

public void **setOneButton**(javax.swing.JButton oneButton)

### setOtherItemButton

public void **setOtherItemButton**(javax.swing.JButton otherItemButton)

### setScrollPane

public void **setScrollPane**(javax.swing.JScrollPane scrollPane)

### setSevenButton

public void **setSevenButton**(javax.swing.JButton sevenButton)

### setSixButton

public void **setSixButton**(javax.swing.JButton sixButton)

### setThreeButton

public void **setThreeButton**(javax.swing.JButton threeButton)

### setTotalTextField

public void **setTotalTextField**(javax.swing.JTextField totalTextField)

### setTwoButton

public void **setTwoButton**(javax.swing.JButton twoButton)

### setxButton

public void **setxButton**(javax.swing.JButton xButton)

### setZeroButton

public void **setZeroButton**(javax.swing.JButton zeroButton)

### showGUI

public void **showGUI**()

Makes the frame visible.

**PointOfSaleView class code listing:**

package pospi;

import java.util.\*;

import java.text.\*;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JTextArea;

import javax.swing.JButton;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JTextField;

import java.awt.Font;

import java.awt.Color;

import javax.swing.SwingConstants;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Constructs and initializes the components and graphical user interface

\* of the pos-pi sales screen

\* \*/

public class PointOfSaleView {

//gui components declaration.

private JFrame mainFrame;

private JTextField barcodeTextField;

private JTextField totalTextField;

private JTextArea itemDisplayTextArea;

private JScrollPane scrollPane;

private JButton sevenButton;

private JButton eightButton;

private JButton nineButton;

private JButton fourButton;

private JButton fiveButton;

private JButton sixButton;

private JButton oneButton;

private JButton twoButton;

private JButton threeButton;

private JButton xButton;

private JButton zeroButton;

private JButton enterBarcodeButton;

private JButton cashButton;

private JButton cardButton;

private JButton cancelSaleButton;

private JButton cancelItemButton;

private JButton otherItemButton;

/\*\*

\* Constructor.

\*/

public PointOfSaleView() {

initialize();

}//end constructor

/\*

\* Getters.

\* \*/

public JFrame getMainFrame() {

return mainFrame;

}

public JTextField getBarcodeTextField() {

return barcodeTextField;

}

public JTextField getTotalTextField() {

return totalTextField;

}

public JTextArea getItemDisplayTextArea() {

return itemDisplayTextArea;

}

public JButton getSevenButton() {

return sevenButton;

}

public JButton getEightButton() {

return eightButton;

}

public JButton getNineButton() {

return nineButton;

}

public JButton getFourButton() {

return fourButton;

}

public JButton getFiveButton() {

return fiveButton;

}

public JButton getSixButton() {

return sixButton;

}

public JButton getOneButton() {

return oneButton;

}

public JButton getTwoButton() {

return twoButton;

}

public JButton getThreeButton() {

return threeButton;

}

public JButton getxButton() {

return xButton;

}

public JButton getZeroButton() {

return zeroButton;

}

public JButton getEnterBarcodeButton() {

return enterBarcodeButton;

}

public JButton getCashButton() {

return cashButton;

}

public JButton getCardButton() {

return cardButton;

}

public JButton getCancelSaleButton() {

return cancelSaleButton;

}

public JButton getCancelItemButton() {

return cancelItemButton;

}

public JButton getOtherItemButton() {

return otherItemButton;

}

public JScrollPane getScrollPane(){

return scrollPane;

}

/\*

\* Setters.

\* \*/

public void setMainFrame(JFrame mainFrame) {

this.mainFrame = mainFrame;

}

public void setBarcodeTextField(JTextField barcodeTextField) {

this.barcodeTextField = barcodeTextField;

}

public void setTotalTextField(JTextField totalTextField) {

this.totalTextField = totalTextField;

}

public void setItemDisplayTextArea(JTextArea itemDisplayTextArea) {

this.itemDisplayTextArea = itemDisplayTextArea;

}

public void setSevenButton(JButton sevenButton) {

this.sevenButton = sevenButton;

}

public void setEightButton(JButton eightButton) {

this.eightButton = eightButton;

}

public void setNineButton(JButton nineButton) {

this.nineButton = nineButton;

}

public void setFourButton(JButton fourButton) {

this.fourButton = fourButton;

}

public void setFiveButton(JButton fiveButton) {

this.fiveButton = fiveButton;

}

public void setSixButton(JButton sixButton) {

this.sixButton = sixButton;

}

public void setOneButton(JButton oneButton) {

this.oneButton = oneButton;

}

public void setTwoButton(JButton twoButton) {

this.twoButton = twoButton;

}

public void setThreeButton(JButton threeButton) {

this.threeButton = threeButton;

}

public void setxButton(JButton xButton) {

this.xButton = xButton;

}

public void setZeroButton(JButton zeroButton) {

this.zeroButton = zeroButton;

}

public void setEnterBarcodeButton(JButton backSpaceButton) {

this.enterBarcodeButton = backSpaceButton;

}

public void setCashButton(JButton cashButton) {

this.cashButton = cashButton;

}

public void setCardButton(JButton cardButton) {

this.cardButton = cardButton;

}

public void setCancelSaleButton(JButton cancelSaleButton) {

this.cancelSaleButton = cancelSaleButton;

}

public void setCancelItemButton(JButton cancelItemButton) {

this.cancelItemButton = cancelItemButton;

}

public void setOtherItemButton(JButton otherItemButton) {

this.otherItemButton = otherItemButton;

}

public void setScrollPane(JScrollPane scrollPane){

this.scrollPane = scrollPane;

}

/\*\*

\* Initialize the gui components of the frame.

\*/

private void initialize() {

mainFrame = new JFrame();

mainFrame.setForeground(Color.BLACK);

mainFrame.setBounds(50, 50, 320, 240);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

totalTextField = new JTextField();

totalTextField.setForeground(Color.WHITE);

totalTextField.setBackground(Color.BLACK);

totalTextField.setHorizontalAlignment(SwingConstants.CENTER);

totalTextField.setFont(new Font("Tahoma", Font.BOLD, 12));

totalTextField.setColumns(10);

totalTextField.setText("0.00");

totalTextField.setFocusable(false);

JLabel lblTotal = new JLabel("TOTAL");

lblTotal.setFont(new Font("Tahoma", Font.BOLD, 10));

scrollPane = new JScrollPane();

sevenButton = new JButton("7");

sevenButton.setForeground(Color.BLACK);

sevenButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sevenButton.setFocusable(false);

eightButton = new JButton("8");

eightButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

eightButton.setFocusable(false);

nineButton = new JButton("9");

nineButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

nineButton.setFocusable(false);

fourButton = new JButton("4");

fourButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

fourButton.setFocusable(false);

barcodeTextField = new JTextField();

barcodeTextField.setColumns(10);

fiveButton = new JButton("5");

fiveButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

fiveButton.setFocusable(false);

sixButton = new JButton("6");

sixButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sixButton.setFocusable(false);

oneButton = new JButton("1");

oneButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

oneButton.setFocusable(false);

twoButton = new JButton("2");

twoButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

twoButton.setFocusable(false);

threeButton = new JButton("3");

threeButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

threeButton.setFocusable(false);

xButton = new JButton("x");

xButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

xButton.setFocusable(false);

zeroButton = new JButton("0");

zeroButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

zeroButton.setFocusable(false);

enterBarcodeButton = new JButton("<");

enterBarcodeButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

enterBarcodeButton.setFocusable(false);

cashButton = new JButton("cash");

cashButton.setForeground(Color.BLACK);

cashButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

cashButton.setFocusable(false);

cardButton = new JButton("card");

cardButton.setForeground(Color.BLACK);

cardButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

cardButton.setFocusable(false);

cancelSaleButton = new JButton("cancel sale");

cancelSaleButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

cancelSaleButton.setFocusable(false);

/\*

\* Layout the components using the GroupLayout manager.

\* \*/

GroupLayout gl\_panel = new GroupLayout(panel);

gl\_panel.setHorizontalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addComponent(barcodeTextField, GroupLayout.DEFAULT\_SIZE, 131, Short.MAX\_VALUE)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.TRAILING, false)

.addComponent(fourButton, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(sevenButton, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addPreferredGap(ComponentPlacement.RELATED, 7, Short.MAX\_VALUE)

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(eightButton)

.addGap(5)

.addComponent(nineButton))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(fiveButton)

.addPreferredGap(ComponentPlacement.RELATED, 7, Short.MAX\_VALUE)

.addComponent(sixButton))))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(oneButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(twoButton)

.addPreferredGap(ComponentPlacement.RELATED, 8, Short.MAX\_VALUE)

.addComponent(threeButton))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(xButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(zeroButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(enterBarcodeButton))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(cashButton, GroupLayout.PREFERRED\_SIZE, 59, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED)

.addComponent(cardButton, GroupLayout.DEFAULT\_SIZE, 62, Short.MAX\_VALUE))

.addComponent(cancelSaleButton, GroupLayout.DEFAULT\_SIZE, 131, Short.MAX\_VALUE))

.addContainerGap())

);

gl\_panel.setVerticalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(barcodeTextField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(sevenButton)

.addComponent(nineButton)

.addComponent(eightButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(fourButton)

.addComponent(fiveButton)

.addComponent(sixButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(oneButton)

.addComponent(twoButton)

.addComponent(threeButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(xButton)

.addComponent(zeroButton)

.addComponent(enterBarcodeButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(cashButton)

.addComponent(cardButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(cancelSaleButton)

.addContainerGap(GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

panel.setLayout(gl\_panel);

cancelItemButton = new JButton("cancel (i)");

cancelItemButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

cancelItemButton.setFocusable(false);

otherItemButton = new JButton("other (i)");

otherItemButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

otherItemButton.setFocusable(false);

GroupLayout groupLayout = new GroupLayout(mainFrame.getContentPane());

groupLayout.setHorizontalGroup(

groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup()

.addContainerGap()

.addGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addComponent(scrollPane, GroupLayout.DEFAULT\_SIZE, 142, Short.MAX\_VALUE)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(lblTotal)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(totalTextField, GroupLayout.DEFAULT\_SIZE, 102, Short.MAX\_VALUE))

.addGroup(groupLayout.createSequentialGroup()

.addComponent(cancelItemButton)

.addPreferredGap(ComponentPlacement.RELATED, 14, Short.MAX\_VALUE)

.addComponent(otherItemButton)))

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(panel, GroupLayout.PREFERRED\_SIZE, 141, GroupLayout.PREFERRED\_SIZE)

.addGap(5))

);

groupLayout.setVerticalGroup(

groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup()

.addGap(11)

.addGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addComponent(panel, Alignment.TRAILING, GroupLayout.PREFERRED\_SIZE, 190, Short.MAX\_VALUE)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(scrollPane, GroupLayout.DEFAULT\_SIZE, 125, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addComponent(totalTextField, GroupLayout.PREFERRED\_SIZE, 23, GroupLayout.PREFERRED\_SIZE)

.addComponent(lblTotal))

.addGap(2)

.addGroup(groupLayout.createParallelGroup(Alignment.BASELINE)

.addComponent(cancelItemButton)

.addComponent(otherItemButton))

.addContainerGap())))

);

itemDisplayTextArea = new JTextArea();

itemDisplayTextArea.setWrapStyleWord(true);

itemDisplayTextArea.setLineWrap(true);

itemDisplayTextArea.setEditable(false);

itemDisplayTextArea.setFocusable(false);

scrollPane.setViewportView(itemDisplayTextArea);

mainFrame.getContentPane().setLayout(groupLayout);

}//end initialize()

/\*\*

\* Makes the frame visible.

\*\*/

public void showGUI(){

Date currentDate = new Date();

mainFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

mainFrame.setLocationRelativeTo(null);

mainFrame.setVisible(true);

mainFrame.setTitle("Sale Date: " + getFormattedDate(currentDate));

mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

}//end showGUI

/\*\*

\* Returns the formatted date as a String using the MM-DD-YYYY format.

\* @param date Date

\* @return String

\*\*/

public String getFormattedDate(Date date){

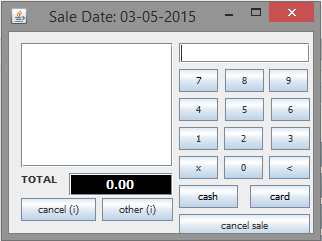
SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");

return format.format(date);

}//

}//end class

**Screenshot:**



**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class PointOfSaleController**

java.lang.Object

**pospi.PointOfSaleController**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.awt.event.KeyListener, java.awt.print.Printable, java.util.EventListener

public class **PointOfSaleController**extends java.lang.Objectimplements java.awt.event.KeyListener, java.awt.event.ActionListener, java.awt.print.Printable

Author: Angelo Romel Lopez  
Description: Handles the logic and events of the PointOfSaleView class.

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from interface java.awt.print.Printable** |
| NO\_SUCH\_PAGE, PAGE\_EXISTS |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**PointOfSaleController**](about:blank../pospi/PointOfSaleController.html#PointOfSaleController())()           Constructor. |  |
| [**PointOfSaleController**](about:blank../pospi/PointOfSaleController.html#PointOfSaleController(pospi.PointOfSaleView, java.lang.String, java.lang.String, java.lang.String))(pospi.PointOfSaleView view, java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor, overloaded. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/PointOfSaleController.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. |
| void | [**clearAllDisplay**](about:blank../pospi/PointOfSaleController.html#clearAllDisplay())()           Clears the ArrayList, TextArea and TextFields. |
| boolean | [**commitSaleTransaction**](about:blank../pospi/PointOfSaleController.html#commitSaleTransaction())()           Save changes to database: [1] insert new order record. |
| void | [**deleteItemOrdered**](about:blank../pospi/PointOfSaleController.html#deleteItemOrdered(int))(int itemPos)           Deletes the item from the TextArea of items ordered. |
| void | [**displayItemsOrdered**](about:blank../pospi/PointOfSaleController.html#displayItemsOrdered())()           Displays the contents of the Product ArrayList on the TextArea. |
| int | [**getDeleteItemPosition**](about:blank../pospi/PointOfSaleController.html#getDeleteItemPosition())()           Returns the line number of an item from the TextArea by making the user choose from a dropdown list. |
| void | [**getItemFromBarcode**](about:blank../pospi/PointOfSaleController.html#getItemFromBarcode())()           Retrieves the item from the database using the barcode attribute and displays it in the TextArea. |
| void | [**getItemFromUOM**](about:blank../pospi/PointOfSaleController.html#getItemFromUOM(int, double))(int productID, double quantity)           Retrieves an item from the database that have a unit-of-measurement code and displays it in the TextArea. |
| java.util.ArrayList<pospi.ProductOrder> | [**getProductOrderList**](about:blank../pospi/PointOfSaleController.html#getProductOrderList())() |
| double | [**getTotalPrice**](about:blank../pospi/PointOfSaleController.html#getTotalPrice())() |
| void | [**initialize**](about:blank../pospi/PointOfSaleController.html#initialize())()           Initialize the event listener of the components. |
| void | [**keyPressed**](about:blank../pospi/PointOfSaleController.html#keyPressed(java.awt.event.KeyEvent))(java.awt.event.KeyEvent e) |
| void | [**keyReleased**](about:blank../pospi/PointOfSaleController.html#keyReleased(java.awt.event.KeyEvent))(java.awt.event.KeyEvent e) |
| void | [**keyTyped**](about:blank../pospi/PointOfSaleController.html#keyTyped(java.awt.event.KeyEvent))(java.awt.event.KeyEvent e)           Implemented KeyListener abstract methods. |
| int | [**print**](about:blank../pospi/PointOfSaleController.html#print(java.awt.Graphics, java.awt.print.PageFormat, int))(java.awt.Graphics graphics, java.awt.print.PageFormat pageFormat, int pageIndex)           Implemented abstract Printable.print() method. |
| void | [**printReceipt**](about:blank../pospi/PointOfSaleController.html#printReceipt())()           Format the page/s to be printed to setPrintable. |
| java.sql.ResultSet | [**runSelect**](about:blank../pospi/PointOfSaleController.html#runSelect(java.lang.String))(java.lang.String selectStatement)           Executes a select query on the database. |
| void | [**setProductOrderList**](about:blank../pospi/PointOfSaleController.html#setProductOrderList(java.util.ArrayList))(java.util.ArrayList<pospi.ProductOrder> productOrderList) |
| void | [**setTotalPrice**](about:blank../pospi/PointOfSaleController.html#setTotalPrice(double))(double totalPrice) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |
| **Constructor Detail** |

### PointOfSaleController

public **PointOfSaleController**()

Constructor.

### PointOfSaleController

public **PointOfSaleController**(pospi.PointOfSaleView view,

java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor, overloaded.

**Parameters:**

view - PointOfSaleView

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods. Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### clearAllDisplay

public void **clearAllDisplay**()

Clears the ArrayList, TextArea and TextFields.

### commitSaleTransaction

public boolean **commitSaleTransaction**()

Save changes to database: [1] insert new order record. [2] update product inventory table. Use manual transaction to commit changes. Return true if changes to database was successful, return false otherwise.

**Returns:**

boolean

### deleteItemOrdered

public void **deleteItemOrdered**(int itemPos)

Deletes the item from the TextArea of items ordered.

**Parameters:**

itemPos - int

### displayItemsOrdered

public void **displayItemsOrdered**()

Displays the contents of the Product ArrayList on the TextArea.

### getDeleteItemPosition

public int **getDeleteItemPosition**()

Returns the line number of an item from the TextArea by making the user choose from a dropdown list.

**Returns:**

int

### getItemFromBarcode

public void **getItemFromBarcode**()

Retrieves the item from the database using the barcode attribute and displays it in the TextArea.

### getItemFromUOM

public void **getItemFromUOM**(int productID,

double quantity)

Retrieves an item from the database that have a unit-of-measurement code and displays it in the TextArea.

**Parameters:**

productID - int

quantity - double

### getProductOrderList

public java.util.ArrayList<pospi.ProductOrder> **getProductOrderList**()

### getTotalPrice

public double **getTotalPrice**()

### initialize

public void **initialize**()

Initialize the event listener of the components.

### keyPressed

public void **keyPressed**(java.awt.event.KeyEvent e)

**Specified by:**

keyPressed in interface java.awt.event.KeyListener

### keyReleased

public void **keyReleased**(java.awt.event.KeyEvent e)

**Specified by:**

keyReleased in interface java.awt.event.KeyListener

### keyTyped

public void **keyTyped**(java.awt.event.KeyEvent e)

Implemented KeyListener abstract methods.

**Specified by:**

keyTyped in interface java.awt.event.KeyListener

**Parameters:**

e - KeyEvent

### print

public int **print**(java.awt.Graphics graphics,

java.awt.print.PageFormat pageFormat,

int pageIndex)

throws java.awt.print.PrinterException

Implemented abstract Printable.print() method. Print the actual contents of the TextArea.

**Specified by:**

print in interface java.awt.print.Printable

**Parameters:**

graphics - Graphics

pageFormat - PageFormat

pageIndex - int

**Returns:**

int

**Throws:**

java.awt.print.PrinterException

### printReceipt

public void **printReceipt**()

Format the page/s to be printed to setPrintable.

### runSelect

public java.sql.ResultSet **runSelect**(java.lang.String selectStatement)

Executes a select query on the database.

**Parameters:**

selectStatement -

**Returns:**

ResultSet

### setProductOrderList

public void **setProductOrderList**(java.util.ArrayList<pospi.ProductOrder> productOrderList)

### setTotalPrice

public void **setTotalPrice**(double totalPrice)

**runSelect() – Sequence diagram**

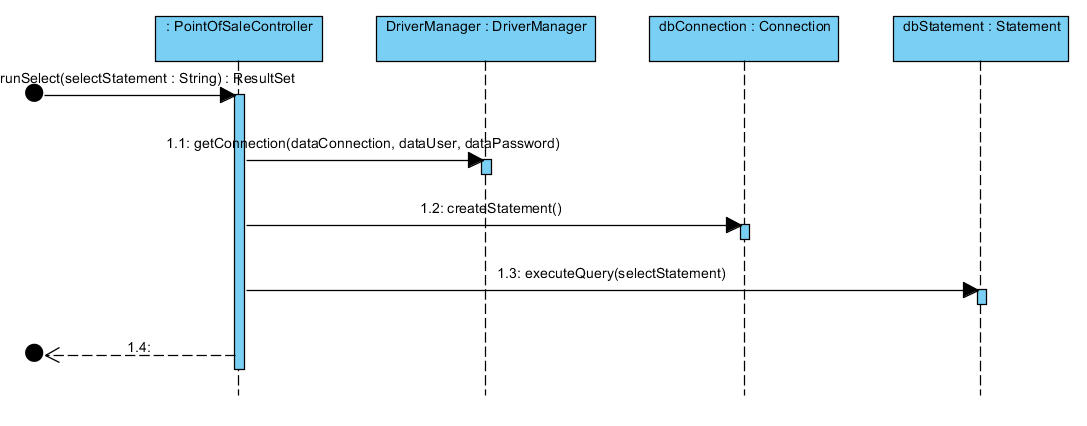
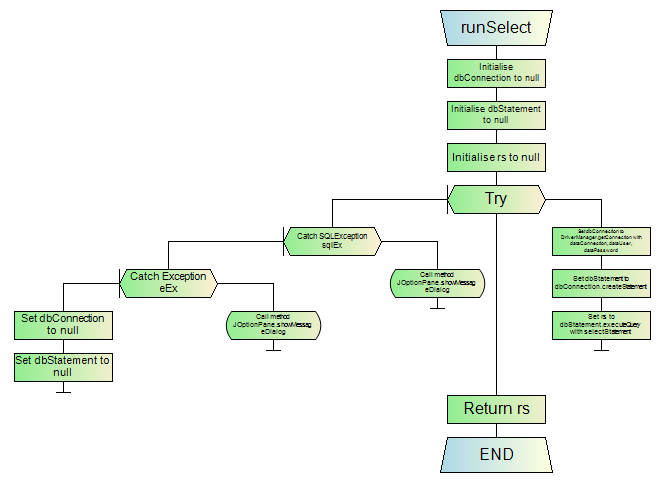


Diagram created with Visual Paradigm (http://www.visual-paradigm.com/)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**runSelect() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**printReceipt() – Sequence diagram**

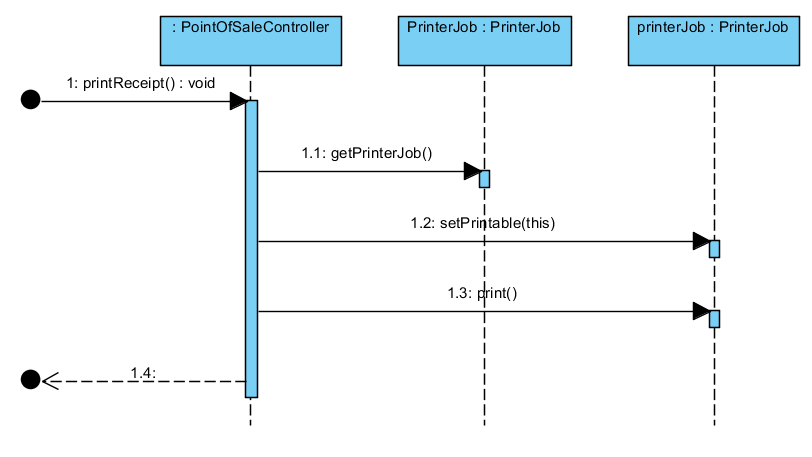
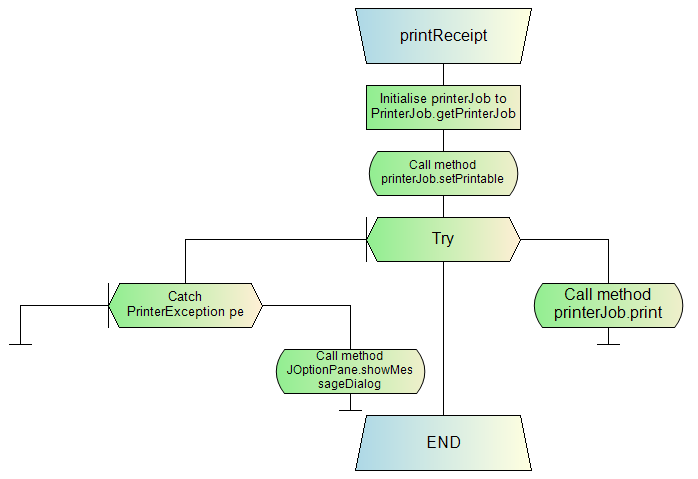


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**printReceipt() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**displayItemsOrdered() – Sequence diagram**

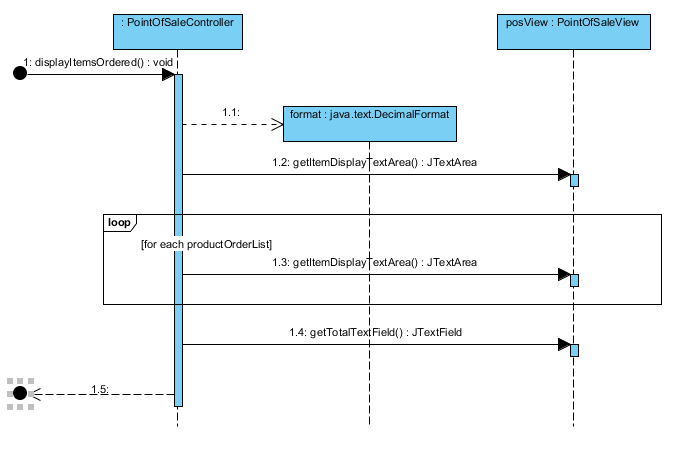
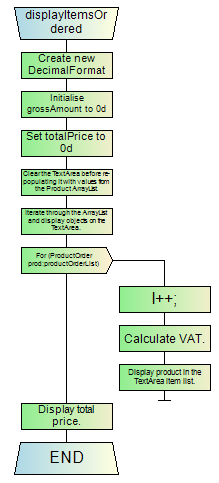


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**displayItemsOrdered() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**PointOfSaleController class code listing:**

package pospi;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.print.\*;

import javax.swing.JOptionPane;

import javax.swing.JScrollBar;

import java.sql.\*;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ArrayList;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events of the PointOfSaleView class.

\* \*/

public class PointOfSaleController implements KeyListener, ActionListener, Printable{

private PointOfSaleView posView;//Declare the view class to attach the controller class to.

private String dataConnection;

private String dataUser;

private String dataPassword;

private ArrayList <ProductOrder> productOrderList;

private double totalPrice;

/\*\*

\* Constructor.

\*/

public PointOfSaleController(){

posView = new PointOfSaleView();

dataConnection = "";

dataUser = "";

dataPassword = "";

productOrderList = new ArrayList <ProductOrder>();

totalPrice = 0d;

initialize();

posView = new PointOfSaleView();

posView.showGUI();

}//end constructor.

/\*\*

\* Constructor, overloaded.

\* @param view PointOfSaleView

\* @param connection String

\* @param user String

\* @param password String

\*/

public PointOfSaleController(PointOfSaleView view, String connection, String user, String password){

posView = view;

dataConnection = connection;

dataUser = user;

dataPassword = password;

productOrderList = new ArrayList <ProductOrder>();

totalPrice = 0d;

initialize();

posView.showGUI();

}//end constructor.

/\*\*

\* Initialize the event listener of the components.

\*/

public void initialize(){

posView.getBarcodeTextField().addKeyListener(this);

posView.getSevenButton().addActionListener(this);

posView.getSevenButton().setActionCommand("seven");

posView.getEightButton().addActionListener(this);

posView.getEightButton().setActionCommand("eight");

posView.getNineButton().addActionListener(this);

posView.getNineButton().setActionCommand("nine");

posView.getFourButton().addActionListener(this);

posView.getFourButton().setActionCommand("four");

posView.getFiveButton().addActionListener(this);

posView.getFiveButton().setActionCommand("five");

posView.getSixButton().addActionListener(this);

posView.getSixButton().setActionCommand("six");

posView.getOneButton().addActionListener(this);

posView.getOneButton().setActionCommand("one");

posView.getTwoButton().addActionListener(this);

posView.getTwoButton().setActionCommand("two");

posView.getThreeButton().addActionListener(this);

posView.getThreeButton().setActionCommand("three");

posView.getZeroButton().addActionListener(this);

posView.getZeroButton().setActionCommand("zero");

posView.getxButton().addActionListener(this);

posView.getxButton().setActionCommand("x");

posView.getEnterBarcodeButton().addActionListener(this);

posView.getEnterBarcodeButton().setActionCommand("enter");

posView.getCashButton().addActionListener(this);

posView.getCashButton().setActionCommand("cash");

posView.getCardButton().addActionListener(this);

posView.getCardButton().setActionCommand("card");

posView.getCancelSaleButton().addActionListener(this);

posView.getCancelSaleButton().setActionCommand("cancel sale");

posView.getCancelItemButton().addActionListener(this);

posView.getCancelItemButton().setActionCommand("cancel item");

posView.getOtherItemButton().addActionListener(this);

posView.getOtherItemButton().setActionCommand("other item");

}//end initialize

//Getters.

public ArrayList<ProductOrder> getProductOrderList() {

return productOrderList;

}

public double getTotalPrice() {

return totalPrice;

}

//Setters.

public void setProductOrderList(ArrayList<ProductOrder> productOrderList) {

this.productOrderList = productOrderList;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

/\*\*

\* Implemented KeyListener abstract methods.

\* @param e KeyEvent

\* \*/

public void keyTyped(KeyEvent e){};

public void keyReleased(KeyEvent e){};

public void keyPressed(KeyEvent e){

int key = e.getKeyCode();

if(key == KeyEvent.VK\_ENTER){//barcode suffix encountered. Retrieve barcode from database.

getItemFromBarcode();

posView.getBarcodeTextField().setText("");

}

}//end keyPressed()

/\*\*

\* Implemented ActionListener abstract methods.

\* Handles the events when a button is pressed.

\* @param e ActionEvent

\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

try{

Robot robot = new Robot();

switch(command){

case "seven":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_7);

/\*

\*keyRelease is needed for linux based systems for the keyPress

\*to work properly. This is not needed on a Microsoft Windows system.

\*/

robot.keyRelease(KeyEvent.VK\_7);

break;

case "eight":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_8);

robot.keyRelease(KeyEvent.VK\_8);

break;

case "nine":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_9);

robot.keyRelease(KeyEvent.VK\_9);

break;

case "four":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_4);

robot.keyRelease(KeyEvent.VK\_4);

break;

case "five":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_5);

robot.keyRelease(KeyEvent.VK\_5);

break;

case "six":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_6);

robot.keyRelease(KeyEvent.VK\_6);

break;

case "one":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_1);

robot.keyRelease(KeyEvent.VK\_1);

break;

case "two":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_2);

robot.keyRelease(KeyEvent.VK\_2);

break;

case "three":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_3);

robot.keyRelease(KeyEvent.VK\_3);

break;

case "zero":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_0);

robot.keyRelease(KeyEvent.VK\_0);

break;

case "x":

posView.getBarcodeTextField().setText("");

break;

case "enter":

//Simulate a key press

robot.keyPress(KeyEvent.VK\_ENTER);

robot.keyRelease(KeyEvent.VK\_ENTER);

break;

case "cash":

//Process cash sale.

DecimalFormat format = new DecimalFormat("#####.00");

SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

if(posView.getItemDisplayTextArea().getText().equals("")){

JOptionPane.showMessageDialog(null,"Your basket is empty.\nThere are no items to checkout.",

"POS-Pi", JOptionPane.WARNING\_MESSAGE);

}

else{

int replyCash = JOptionPane.showConfirmDialog(null, "Proceed to cash checkout?", "POS-Pi",

JOptionPane.YES\_NO\_OPTION);

if(replyCash == JOptionPane.YES\_OPTION){

PayByCashDialog cash = new PayByCashDialog(posView.getMainFrame(), true, getTotalPrice());

cash.setLocationRelativeTo(null);

cash.setVisible(true);

if(cash.isDialogOk()){

if(commitSaleTransaction()){

JOptionPane.showMessageDialog(null,"Checkout was successful!!!",

"POS-Pi", JOptionPane.INFORMATION\_MESSAGE);

//Add title to the TextArea.

posView.getItemDisplayTextArea().insert("POS-Pi\nTransaction Date: " + dateFormat.format(new Date()) + "\n", 0);

//Append total price to the item display TextArea.

posView.getItemDisplayTextArea().append("Total Price: " + format.format(this.totalPrice));

//Print receipt.

printReceipt();

clearAllDisplay();

posView.getMainFrame().dispose();

}

else{

JOptionPane.showMessageDialog(null,"An error was encountered\n" +

"during the checkout process.\n" +

"Sale transaction was cancelled.",

"POS-Pi", JOptionPane.WARNING\_MESSAGE);

clearAllDisplay();

}//end if

}//end if

cash.dispose();

}//end if

}//end if

break;

case "card":

JOptionPane.showMessageDialog(null,"Payment by card is not \n"+

"currently available.\n" +

"Only cash payment is allowed.",

"POS-Pi", JOptionPane.WARNING\_MESSAGE);

break;

case "cancel sale":

//Cancel the sale and close the POS window.

int replyCancelSale = JOptionPane.showConfirmDialog(null, "Do you wish to\ncancel/close the sale?", "POS-Pi",

JOptionPane.YES\_NO\_OPTION);

if(replyCancelSale == JOptionPane.YES\_OPTION){

clearAllDisplay();

posView.getMainFrame().dispose();

}//end if

break;

case "cancel item":

//Removes an item from the sale.

if(posView.getItemDisplayTextArea().getText().equals("")){

JOptionPane.showMessageDialog(null,"The product list is empty.\nThere are no items to delete.",

"POS-Pi", JOptionPane.WARNING\_MESSAGE);

}

else{

//Get the item position to be deleted.

int itemPosition = getDeleteItemPosition();

if(itemPosition != 0){

//Delete the item from the product list in the TextArea.

deleteItemOrdered(itemPosition - 1);

//Update the items ordered TextArea.

displayItemsOrdered();

}//end if

}//end if

break;

case "other item":

//Create an instance of the custom dialog and set it to modal.

UnitSaleDialog usd = new UnitSaleDialog(posView.getMainFrame(), true,

this.dataConnection,

this.dataUser,

this.dataPassword);

usd.setLocationRelativeTo(null);

usd.setVisible(true);

if(usd.isDialogOk()){

getItemFromUOM(usd.getProductID(), usd.getQuantity());

}//end if

usd.dispose();//Dispose the custom dialog.

break;

}//end switch

robot = null;

}//end try

catch(Exception ex){

ex.printStackTrace();

}//end catch

}//end actionPerformed

/\*\*

\* Retrieves the item from the database using the barcode attribute and displays it in the TextArea.

\*/

public void getItemFromBarcode(){

ResultSet rs = null;

String selectStatement = "SELECT products.product\_id, " +

"products.barcode, " +

"products.uom\_id, " +

"products.description, " +

"products.price, " +

"products.vat\_code, " +

"vat.rate " + "FROM products LEFT JOIN vat " +

"ON products.vat\_code = vat.vat\_code " +

"WHERE products.barcode = '" + posView.getBarcodeTextField().getText() + "'";

try{

rs = runSelect(selectStatement);

//if ResultSet is not before the first result, then ResultSet is empty.

if(! rs.isBeforeFirst()){

JOptionPane.showMessageDialog(null, "Sorry, the barcode is\nnot recognized.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

//ResultSet is before the first result, record is found.

else{

rs.next();

//Create a new object to hold the product information and store it

//in an ArrayList.

ProductOrder product = new ProductOrder();

product.setProductID(rs.getInt("product\_id"));

product.setBarcode(rs.getString("barcode"));

product.setUomID(rs.getInt("uom\_id"));

product.setUomCode("");

product.setDescription(rs.getString("description"));

product.setPrice(rs.getDouble("price"));

product.setVatRate(rs.getByte("rate"));

product.setQuantity(1);

//Add product to the ArrayList.

productOrderList.add(product);

//Display the contents of the ArrayList.

displayItemsOrdered();

//Show the most recent item on the text area by scrolling all the way down to the bottom.

JScrollBar scrollBar = posView.getScrollPane().getVerticalScrollBar();

scrollBar.setValue(scrollBar.getMaximum());

}//end if

}//end try

catch(Exception ex){

JOptionPane.showMessageDialog(null, ex.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

}//end getItem()

/\*\*

\* Retrieves an item from the database that have a unit-of-measurement code and displays it in the TextArea.

\* @param productID int

\* @param quantity double

\*/

public void getItemFromUOM(int productID, double quantity){

ResultSet rs = null;

String selectStatement = "SELECT products.product\_id, products.barcode, products.uom\_id, " +

"products.description, products.price, products.vat\_code, vat.rate, " +

"unit\_of\_measurement.uom\_code " +

"FROM products LEFT JOIN vat " +

"ON products.vat\_code = vat.vat\_code " +

"LEFT JOIN unit\_of\_measurement " +

"ON products.uom\_id = unit\_of\_measurement.uom\_id " +

"WHERE products.product\_id = " + productID + ";";

try{

rs = runSelect(selectStatement);

//if ResultSet is not before the first result, then ResultSet is empty.

if(! rs.isBeforeFirst()){

JOptionPane.showMessageDialog(null, "Sorry, the product\nis not recognized.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

//ResultSet is before the first result, record is found.

else{

rs.next();

//Create a new object to hold the product information and store it

//in an ArrayList.

ProductOrder product = new ProductOrder();

product.setProductID(rs.getInt("product\_id"));

product.setBarcode(rs.getString("barcode"));

product.setUomID(rs.getInt("uom\_id"));

product.setUomCode(rs.getString("uom\_code"));

product.setDescription(rs.getString("description"));

product.setPrice(rs.getDouble("price"));

product.setVatRate(rs.getByte("rate"));

product.setQuantity(quantity);

//Add product to the ArrayList.

productOrderList.add(product);

//Display the contents of the ArrayList.

displayItemsOrdered();

//Show the most recent item on the text area by scrolling all the way down to the bottom.

JScrollBar scrollBar = posView.getScrollPane().getVerticalScrollBar();

scrollBar.setValue(scrollBar.getMaximum());

}//end if

}//end try

catch(Exception ex){

JOptionPane.showMessageDialog(null, ex.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

}//end getItemFromUOM

/\*\*

\* Executes a select query on the database.

\* @param selectStatement

\* @return ResultSet

\*/

public ResultSet runSelect(String selectStatement){

Connection dbConnection = null;

Statement dbStatement = null;

ResultSet rs = null;

try{

dbConnection = DriverManager.getConnection(dataConnection, dataUser, dataPassword);

dbStatement = dbConnection.createStatement();

rs = dbStatement.executeQuery(selectStatement);

}//end try

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

catch(Exception eEx){

JOptionPane.showMessageDialog(null, eEx.toString(),

"Error in connecting to database",

JOptionPane.WARNING\_MESSAGE);

}//end catch

finally{

dbConnection = null;

dbStatement = null;

}//end finally

return rs;

}//end runSelect

/\*\*

\* Displays the contents of the Product ArrayList on the TextArea.

\*/

public void displayItemsOrdered(){

DecimalFormat format = new DecimalFormat("#####.00");

double grossAmount = 0d;

totalPrice = 0d;

//Clear the TextArea before re-populating it with values from the Product ArrayList.

posView.getItemDisplayTextArea().setText("");

//Iterate through the ArrayList and display objects on the TextArea.

int i = 0;//For displaying line number.

for(ProductOrder prod:productOrderList){

i++;

//Calculate VAT.

grossAmount = (prod.getQuantity() \* (prod.getPrice() + (prod.getVatRate() / 100.0) \* prod.getPrice()));

//Display product in the TextArea item list.

posView.getItemDisplayTextArea().append(i + " > " + prod.getDescription() +

" Qty: " + prod.getQuantity() +

" " + prod.getUomCode() + " (" +

format.format(prod.getPrice()) +

" + " + prod.getVatRate() + "% VAT) = " +

format.format(grossAmount) +

"\n"); //+

//"- - - - - - - - - - - - - - - - - - - -" + "\n");

totalPrice += grossAmount;

}//end for

//Display total price.

posView.getTotalTextField().setText(format.format(totalPrice));

}//displayProductOrder

/\*\*

\* Clears the ArrayList, TextArea and TextFields.

\*/

public void clearAllDisplay(){

productOrderList.clear();

posView.getBarcodeTextField().setText("");

posView.getItemDisplayTextArea().setText("");

posView.getTotalTextField().setText("ï¿½0.00");

}//end clearAll

/\*\*

\* Returns the line number of an item from the TextArea by making the user choose from a dropdown list.

\* @return int

\*/

public int getDeleteItemPosition(){

String [] itemPosition = new String [productOrderList.size()];

//Populate the array with the position of the products.

for(int x = 0; x < itemPosition.length; x ++){

itemPosition[x] = String.valueOf(x + 1);

}//end for

//Prompt user to choose the position of the item to delete.

String input = (String)JOptionPane.showInputDialog(null, "Please select the position of\nthe item you wish to delete.",

"POS-Pi", JOptionPane.QUESTION\_MESSAGE, null,

itemPosition, itemPosition[0]);

if(input != null){

return Integer.valueOf(input);//Return the position of the item.

}

else{

return 0;//No item was selected. The user chose cancel.

}//end if

}//end getItemPosition

/\*\*

\* Deletes the item from the TextArea of items ordered.

\* @param itemPos int

\*/

public void deleteItemOrdered(int itemPos){

productOrderList.remove(itemPos);

}//end deleteItemOrdered

/\*\*

\* Save changes to database: [1] insert new order record. [2] update product inventory table.

\* Use manual transaction to commit changes. Return true if changes to database was successful,

\* return false otherwise.

\* @return boolean

\*/

public boolean commitSaleTransaction(){

Connection dbConnection = null;

Statement insertStatement = null;

PreparedStatement updateStatement = null;

int orderPK;//Store auto-generated primary key value.

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

ResultSet rs = null;//For holding auto-generated keys.

try{

dbConnection = DriverManager.getConnection(dataConnection, dataUser, dataPassword);

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//----------------Start of transaction.----------------

insertStatement = dbConnection.createStatement();

insertStatement.executeUpdate("INSERT INTO orders " +

" (order\_date) VALUES ('" + dateFormat.format(new Date()) + "')", Statement.RETURN\_GENERATED\_KEYS);

//Get the auto-increment primary key value.

rs = insertStatement.getGeneratedKeys();

if(rs.next()){

orderPK = rs.getInt(1);//Store PK value.

//Navigate thru entire list of products ordered (Product ArrayList) and update database.

for(ProductOrder prod:productOrderList){

//Insert a new record in the order\_products table.

insertStatement = dbConnection.createStatement();

insertStatement.executeUpdate("INSERT INTO order\_products " +

"(order\_id, product\_id) VALUES (" + orderPK + ", " + prod.getProductID() +")");

//Update inventory table.

updateStatement = dbConnection.prepareStatement("UPDATE product\_inventory " +

"SET quantity\_in\_stock = (quantity\_in\_stock - ?)" +

" WHERE product\_id = ?");

updateStatement.setDouble(1, prod.getQuantity());

updateStatement.setInt(2, prod.getProductID());

updateStatement.executeUpdate();

}//end for

//Commit transaction.

dbConnection.commit();

return true;

}

else{

JOptionPane.showMessageDialog(null, "Unable to retrieve orders\nPK value.",

"Error PK retrieval.",

JOptionPane.WARNING\_MESSAGE);

throw new SQLException();//Throw an exception and rollback changes.

}//end if

}//end try

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

try {

dbConnection.rollback();//Rollback changes if an exception was thrown.

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}//end catch

catch(Exception eEx){

JOptionPane.showMessageDialog(null, eEx.toString(),

"Error in connecting to database",

JOptionPane.WARNING\_MESSAGE);

try {

dbConnection.rollback();//Rollback changes if an exception was thrown.

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}//end catch

finally{

//Do some clean-up.

dbConnection = null;

insertStatement = null;

updateStatement = null;

}//end finally

}//end commitSaleTransaction

/\*\*

\* Format the page/s to be printed to setPrintable.

\*/

public void printReceipt(){

PrinterJob printerJob = PrinterJob.getPrinterJob();

printerJob.setPrintable(this);

try{

printerJob.print();

}//end try

catch(PrinterException pe){

JOptionPane.showMessageDialog(null, "No printer found.",

"Error in printing receipt.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

}//end printReceipt().

/\*\*

\* Implemented abstract Printable.print() method. Print the actual contents of the TextArea.

\* @param graphics Graphics

\* @param pageFormat PageFormat

\* @param pageIndex int

\* @return int

\* @throws PrinterException

\*/

public int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException{

try{

Font oldFont = graphics.getFont();//Store current font, to be restored after printing the receipt.

Font font = new Font("", Font.PLAIN, 6);//Specify the font to be used for printing the receipt.

graphics.setFont(font);//Set the font.

String receipt;

int lineStart,

lineEnd,

lineNumber,

lineCount;

if(pageIndex >= 1) return Printable.NO\_SUCH\_PAGE;

graphics.translate((int)(pageFormat.getImageableX()), (int)(pageFormat.getImageableY()));

graphics.setColor(Color.BLACK);

lineNumber = 0;

lineCount = posView.getItemDisplayTextArea().getLineCount();

receipt = posView.getItemDisplayTextArea().getText();

while(lineNumber < lineCount){

try{

lineStart = posView.getItemDisplayTextArea().getLineStartOffset(lineNumber);

lineEnd = posView.getItemDisplayTextArea().getLineEndOffset(lineNumber);

receipt = posView.getItemDisplayTextArea().getText(lineStart, lineEnd-lineStart);

}//end try

catch(Exception ex){

ex.printStackTrace();

}//end catch.

graphics.drawString(receipt, 0, (lineNumber + 1) \* 20);

lineNumber = lineNumber + 1;

}//end while.

graphics.setFont(oldFont);

return Printable.PAGE\_EXISTS;

}//end try

catch(Exception ex){

JOptionPane.showMessageDialog(null, "No printer found.",

"Error in printing receipt.",

JOptionPane.WARNING\_MESSAGE);

return Printable.NO\_SUCH\_PAGE;

}//end catch

}//end print().

}//end class

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class PayByCashDialog**

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Window

java.awt.Dialog

javax.swing.JDialog

**pospi.PayByCashDialog**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class **PayByCashDialog**extends javax.swing.JDialogimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description:  
[1] Constructs and initializes the components and graphical user interface of the cash payment custom dialog screen.  
[2] Handle the logic and events of the cash payment custom dialog screen.

**See Also:**

[Serialized Form](about:blank../serialized-form.html#pospi.PayByCashDialog)

|  |
| --- |
| **Nested Class Summary** |

|  |
| --- |
| **Nested classes/interfaces inherited from class javax.swing.JDialog** |
| javax.swing.JDialog.AccessibleJDialog |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Dialog** |
| java.awt.Dialog.AccessibleAWTDialog, java.awt.Dialog.ModalExclusionType, java.awt.Dialog.ModalityType |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Window** |
| java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Container** |
| java.awt.Container.AccessibleAWTContainer |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy |

|  |  |
| --- | --- |
| **Field Summary** | |
| (package private)  java.text.DecimalFormat | [**format**](about:blank../pospi/PayByCashDialog.html#format) |

|  |
| --- |
| **Fields inherited from class javax.swing.JDialog** |
| accessibleContext, rootPane, rootPaneCheckingEnabled |

|  |
| --- |
| **Fields inherited from class java.awt.Dialog** |
| DEFAULT\_MODALITY\_TYPE |

|  |
| --- |
| **Fields inherited from class java.awt.Component** |
| BOTTOM\_ALIGNMENT, CENTER\_ALIGNMENT, LEFT\_ALIGNMENT, RIGHT\_ALIGNMENT, TOP\_ALIGNMENT |

|  |
| --- |
| **Fields inherited from interface javax.swing.WindowConstants** |
| DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, EXIT\_ON\_CLOSE, HIDE\_ON\_CLOSE |

|  |
| --- |
| **Fields inherited from interface java.awt.image.ImageObserver** |
| ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**PayByCashDialog**](about:blank../pospi/PayByCashDialog.html#PayByCashDialog(javax.swing.JFrame, boolean, double))(javax.swing.JFrame frame, boolean modal, double saleAmount)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/PayByCashDialog.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. Handles the events when a button is pressed. |
| javax.swing.JButton | [**getCancelButton**](about:blank../pospi/PayByCashDialog.html#getCancelButton())() |
| javax.swing.JButton | [**getClearButton**](about:blank../pospi/PayByCashDialog.html#getClearButton())() |
| javax.swing.JButton | [**getDotButton**](about:blank../pospi/PayByCashDialog.html#getDotButton())() |
| javax.swing.JButton | [**getEightButton**](about:blank../pospi/PayByCashDialog.html#getEightButton())() |
| javax.swing.JButton | [**getFiveButton**](about:blank../pospi/PayByCashDialog.html#getFiveButton())() |
| javax.swing.JButton | [**getFourButton**](about:blank../pospi/PayByCashDialog.html#getFourButton())() |
| javax.swing.JButton | [**getNineButton**](about:blank../pospi/PayByCashDialog.html#getNineButton())() |
| javax.swing.JButton | [**getOkButton**](about:blank../pospi/PayByCashDialog.html#getOkButton())() |
| javax.swing.JButton | [**getOneButton**](about:blank../pospi/PayByCashDialog.html#getOneButton())() |
| double | [**getPaymentAmount**](about:blank../pospi/PayByCashDialog.html#getPaymentAmount())() |
| javax.swing.JTextField | [**getPaymentTextField**](about:blank../pospi/PayByCashDialog.html#getPaymentTextField())() |
| double | [**getSaleAmount**](about:blank../pospi/PayByCashDialog.html#getSaleAmount())() |
| javax.swing.JTextField | [**getSaleAmountTextField**](about:blank../pospi/PayByCashDialog.html#getSaleAmountTextField())() |
| javax.swing.JButton | [**getSevenButton**](about:blank../pospi/PayByCashDialog.html#getSevenButton())() |
| javax.swing.JButton | [**getSixButton**](about:blank../pospi/PayByCashDialog.html#getSixButton())() |
| javax.swing.JButton | [**getThreeButton**](about:blank../pospi/PayByCashDialog.html#getThreeButton())() |
| javax.swing.JButton | [**getTwoButton**](about:blank../pospi/PayByCashDialog.html#getTwoButton())() |
| javax.swing.JButton | [**getZeroButton**](about:blank../pospi/PayByCashDialog.html#getZeroButton())() |
| void | [**initialize**](about:blank../pospi/PayByCashDialog.html#initialize())()           Initialize the event listener of the components. |
| boolean | [**isDialogOk**](about:blank../pospi/PayByCashDialog.html#isDialogOk())()           User had pressed the ok button. |
| boolean | [**isPaymentEnough**](about:blank../pospi/PayByCashDialog.html#isPaymentEnough(double))(double paymentAmount)           Returns true if payment is equal to or greater than total amount to be paid. |
| boolean | [**isValidNumber**](about:blank../pospi/PayByCashDialog.html#isValidNumber(java.lang.String))(java.lang.String number)           Test if a String value is a valid double. |
| void | [**setCancelButton**](about:blank../pospi/PayByCashDialog.html#setCancelButton(javax.swing.JButton))(javax.swing.JButton cancelButton) |
| void | [**setClearButton**](about:blank../pospi/PayByCashDialog.html#setClearButton(javax.swing.JButton))(javax.swing.JButton clearButton) |
| void | [**setDialogOk**](about:blank../pospi/PayByCashDialog.html#setDialogOk(boolean))(boolean dialogOk) |
| void | [**setDotButton**](about:blank../pospi/PayByCashDialog.html#setDotButton(javax.swing.JButton))(javax.swing.JButton dotButton) |
| void | [**setEightButton**](about:blank../pospi/PayByCashDialog.html#setEightButton(javax.swing.JButton))(javax.swing.JButton eightButton) |
| void | [**setFiveButton**](about:blank../pospi/PayByCashDialog.html#setFiveButton(javax.swing.JButton))(javax.swing.JButton fiveButton) |
| void | [**setFourButton**](about:blank../pospi/PayByCashDialog.html#setFourButton(javax.swing.JButton))(javax.swing.JButton fourButton) |
| void | [**setNineButton**](about:blank../pospi/PayByCashDialog.html#setNineButton(javax.swing.JButton))(javax.swing.JButton nineButton) |
| void | [**setOkButton**](about:blank../pospi/PayByCashDialog.html#setOkButton(javax.swing.JButton))(javax.swing.JButton okButton) |
| void | [**setOneButton**](about:blank../pospi/PayByCashDialog.html#setOneButton(javax.swing.JButton))(javax.swing.JButton oneButton) |
| void | [**setPaymentAmount**](about:blank../pospi/PayByCashDialog.html#setPaymentAmount(double))(double paymentAmount) |
| void | [**setPaymentTextField**](about:blank../pospi/PayByCashDialog.html#setPaymentTextField(javax.swing.JTextField))(javax.swing.JTextField paymentTextField) |
| void | [**setSaleAmount**](about:blank../pospi/PayByCashDialog.html#setSaleAmount(double))(double saleAmount) |
| void | [**setSaleAmountTextField**](about:blank../pospi/PayByCashDialog.html#setSaleAmountTextField(javax.swing.JTextField))(javax.swing.JTextField saleAmountTextField) |
| void | [**setSevenButton**](about:blank../pospi/PayByCashDialog.html#setSevenButton(javax.swing.JButton))(javax.swing.JButton sevenButton) |
| void | [**setSixButton**](about:blank../pospi/PayByCashDialog.html#setSixButton(javax.swing.JButton))(javax.swing.JButton sixButton) |
| void | [**setThreeButton**](about:blank../pospi/PayByCashDialog.html#setThreeButton(javax.swing.JButton))(javax.swing.JButton threeButton) |
| void | [**setTwoButton**](about:blank../pospi/PayByCashDialog.html#setTwoButton(javax.swing.JButton))(javax.swing.JButton twoButton) |
| void | [**setZeroButton**](about:blank../pospi/PayByCashDialog.html#setZeroButton(javax.swing.JButton))(javax.swing.JButton zeroButton) |

|  |
| --- |
| **Methods inherited from class javax.swing.JDialog** |
| addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update |

|  |
| --- |
| **Methods inherited from class java.awt.Dialog** |
| addNotify, getModalityType, getTitle, hide, isModal, isResizable, isUndecorated, setBackground, setModal, setModalityType, setOpacity, setResizable, setShape, setTitle, setUndecorated, setVisible, show, toBack |

|  |
| --- |
| **Methods inherited from class java.awt.Window** |
| addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBackground, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOpacity, getOwnedWindows, getOwner, getOwnerlessWindows, getShape, getToolkit, getType, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isOpaque, isShowing, isValidateRoot, pack, paint, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeNotify, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImage, setIconImages, setLocation, setLocation, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setType, toFront |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusDownCycle, validate, validateTree |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resize, resize, revalidate, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusBackward, transferFocusUpCycle |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### format

java.text.DecimalFormat **format**

|  |
| --- |
| **Constructor Detail** |

### PayByCashDialog

public **PayByCashDialog**(javax.swing.JFrame frame,

boolean modal,

double saleAmount)

Constructor.

**Parameters:**

frame - JFrame

modal - boolean

saleAmount - double

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### getCancelButton

public javax.swing.JButton **getCancelButton**()

### getClearButton

public javax.swing.JButton **getClearButton**()

### getDotButton

public javax.swing.JButton **getDotButton**()

### getEightButton

public javax.swing.JButton **getEightButton**()

### getFiveButton

public javax.swing.JButton **getFiveButton**()

### getFourButton

public javax.swing.JButton **getFourButton**()

### getNineButton

public javax.swing.JButton **getNineButton**()

### getOkButton

public javax.swing.JButton **getOkButton**()

### getOneButton

public javax.swing.JButton **getOneButton**()

### getPaymentAmount

public double **getPaymentAmount**()

### getPaymentTextField

public javax.swing.JTextField **getPaymentTextField**()

### getSaleAmount

public double **getSaleAmount**()

### getSaleAmountTextField

public javax.swing.JTextField **getSaleAmountTextField**()

### getSevenButton

public javax.swing.JButton **getSevenButton**()

### getSixButton

public javax.swing.JButton **getSixButton**()

### getThreeButton

public javax.swing.JButton **getThreeButton**()

### getTwoButton

public javax.swing.JButton **getTwoButton**()

### getZeroButton

public javax.swing.JButton **getZeroButton**()

### initialize

public void **initialize**()

Initialize the event listener of the components.

### isDialogOk

public boolean **isDialogOk**()

User had pressed the ok button.

**Returns:**

boolean

### isPaymentEnough

public boolean **isPaymentEnough**(double paymentAmount)

Returns true if payment is equal to or greater than total amount to be paid. Return false otherwise.

**Parameters:**

paymentAmount - double

**Returns:**

boolean

### isValidNumber

public boolean **isValidNumber**(java.lang.String number)

Test if a String value is a valid double. Return true if it is, false otherwise.

**Parameters:**

number - String

**Returns:**

boolean

### setCancelButton

public void **setCancelButton**(javax.swing.JButton cancelButton)

### setClearButton

public void **setClearButton**(javax.swing.JButton clearButton)

### setDialogOk

public void **setDialogOk**(boolean dialogOk)

### setDotButton

public void **setDotButton**(javax.swing.JButton dotButton)

### setEightButton

public void **setEightButton**(javax.swing.JButton eightButton)

### setFiveButton

public void **setFiveButton**(javax.swing.JButton fiveButton)

### setFourButton

public void **setFourButton**(javax.swing.JButton fourButton)

### setNineButton

public void **setNineButton**(javax.swing.JButton nineButton)

### setOkButton

public void **setOkButton**(javax.swing.JButton okButton)

### setOneButton

public void **setOneButton**(javax.swing.JButton oneButton)

### setPaymentAmount

public void **setPaymentAmount**(double paymentAmount)

### setPaymentTextField

public void **setPaymentTextField**(javax.swing.JTextField paymentTextField)

### setSaleAmount

public void **setSaleAmount**(double saleAmount)

### setSaleAmountTextField

public void **setSaleAmountTextField**(javax.swing.JTextField saleAmountTextField)

### setSevenButton

public void **setSevenButton**(javax.swing.JButton sevenButton)

### setSixButton

public void **setSixButton**(javax.swing.JButton sixButton)

### setThreeButton

public void **setThreeButton**(javax.swing.JButton threeButton)

### setTwoButton

public void **setTwoButton**(javax.swing.JButton twoButton)

### setZeroButton

public void **setZeroButton**(javax.swing.JButton zeroButton)

**isPaymentEnough(double) – Sequence diagram**

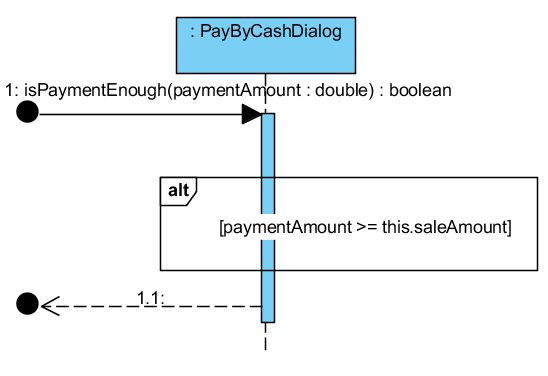
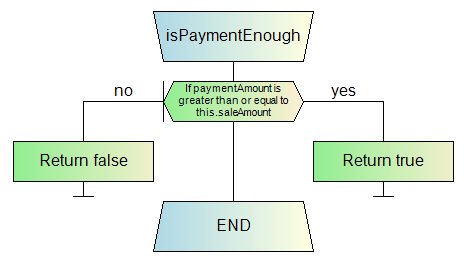


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**isPaymentEnough(double) – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**isValidNumber() – Sequence diagram**

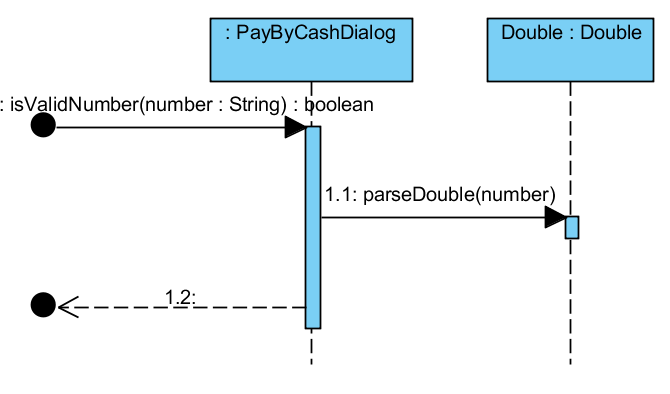
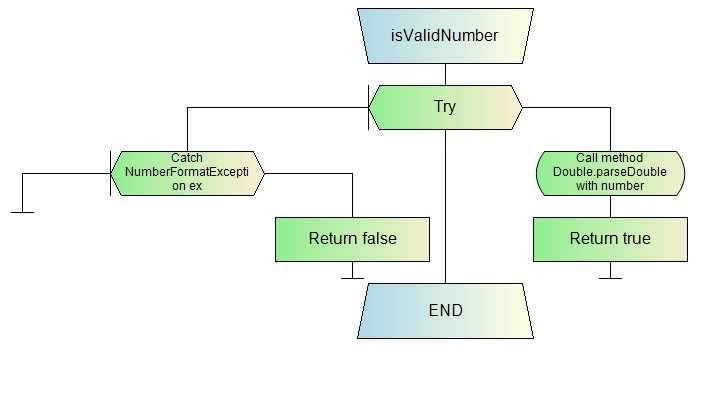


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**isValidNumber() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (http://www.rapidqualitysystems.com/)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**PayByCashDialog class code listing:**

package pospi;

import java.awt.BorderLayout;

import java.awt.FlowLayout;

import java.awt.Robot;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import java.awt.Font;

import java.awt.Color;

import javax.swing.JFrame;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.SwingConstants;

import javax.swing.border.EtchedBorder;

import java.text.DecimalFormat;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: <br>[1] Constructs and initializes the components and graphical user interface

\* of the cash payment custom dialog screen.<br>[2] Handle the logic and events of the cash payment custom dialog screen.

\* \*/

@SuppressWarnings("serial")

public class PayByCashDialog extends JDialog implements ActionListener{

DecimalFormat format;

private double saleAmount;

private double paymentAmount;

private boolean dialogOk;

private final JPanel contentPanel = new JPanel();

private JTextField saleAmountTextField;

private JTextField paymentTextField;

private JButton fourButton;

private JButton sevenButton;

private JButton fiveButton;

private JButton eightButton;

private JButton nineButton;

private JButton sixButton;

private JButton oneButton;

private JButton twoButton;

private JButton threeButton;

private JButton dotButton;

private JButton zeroButton;

private JButton clearButton;

private JButton okButton;

private JButton cancelButton;

/\*\*

\* Constructor.

\* @param frame JFrame

\* @param modal boolean

\* @param saleAmount double

\*\*/

public PayByCashDialog(JFrame frame, boolean modal, double saleAmount) {

super(frame, "Cash Payment", modal);

this.setSaleAmount(saleAmount);

this.format = new DecimalFormat("###,###.00");

setBounds(100, 100, 307, 201);

getContentPane().setLayout(new BorderLayout());

contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

getContentPane().add(contentPanel, BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.setBorder(null);

fourButton = new JButton("4");

fourButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sevenButton = new JButton("7");

sevenButton.setForeground(Color.BLACK);

sevenButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

fiveButton = new JButton("5");

fiveButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

eightButton = new JButton("8");

eightButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

nineButton = new JButton("9");

nineButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sixButton = new JButton("6");

sixButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

oneButton = new JButton("1");

oneButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

twoButton = new JButton("2");

twoButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

threeButton = new JButton("3");

threeButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

dotButton = new JButton(".");

dotButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

zeroButton = new JButton("0");

zeroButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

clearButton = new JButton("x");

clearButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

GroupLayout gl\_panel = new GroupLayout(panel);

gl\_panel.setHorizontalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGap(0, 138, Short.MAX\_VALUE)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_panel.createSequentialGroup()

.addContainerGap()

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.TRAILING, false)

.addComponent(fourButton, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(sevenButton, Alignment.LEADING))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addComponent(fiveButton)

.addComponent(eightButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addComponent(nineButton)

.addComponent(sixButton)))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(oneButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(twoButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(threeButton))

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(dotButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(zeroButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(clearButton)))))

.addContainerGap())

);

gl\_panel.setVerticalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGap(0, 119, Short.MAX\_VALUE)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(sevenButton)

.addComponent(eightButton)

.addComponent(nineButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(fourButton)

.addComponent(fiveButton)

.addComponent(sixButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(oneButton)

.addComponent(twoButton)

.addComponent(threeButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel.createParallelGroup(Alignment.BASELINE)

.addComponent(dotButton)

.addComponent(zeroButton)

.addComponent(clearButton))

.addGap(59)

//.addComponent(button)

.addContainerGap(GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

panel.setLayout(gl\_panel);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

GroupLayout gl\_contentPanel = new GroupLayout(contentPanel);

gl\_contentPanel.setHorizontalGroup(

gl\_contentPanel.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_contentPanel.createSequentialGroup()

.addComponent(panel\_1, GroupLayout.DEFAULT\_SIZE, 147, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(panel, GroupLayout.PREFERRED\_SIZE, 138, GroupLayout.PREFERRED\_SIZE))

);

gl\_contentPanel.setVerticalGroup(

gl\_contentPanel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_contentPanel.createSequentialGroup()

.addGroup(gl\_contentPanel.createParallelGroup(Alignment.LEADING)

.addComponent(panel, GroupLayout.PREFERRED\_SIZE, 119, GroupLayout.PREFERRED\_SIZE)

.addComponent(panel\_1, GroupLayout.DEFAULT\_SIZE, 108, Short.MAX\_VALUE))

.addContainerGap())

);

JLabel lblNewLabel = new JLabel("Sale Amount:");

saleAmountTextField = new JTextField();

saleAmountTextField.setEditable(false);

saleAmountTextField.setFont(new Font("Tahoma", Font.BOLD, 11));

saleAmountTextField.setForeground(Color.WHITE);

saleAmountTextField.setBackground(Color.BLACK);

saleAmountTextField.setHorizontalAlignment(SwingConstants.CENTER);

saleAmountTextField.setText("\u00A30.00");

saleAmountTextField.setColumns(10);

saleAmountTextField.setText(format.format(this.saleAmount));

JLabel lblNewLabel\_1 = new JLabel("Payment:");

paymentTextField = new JTextField();

paymentTextField.setFont(new Font("Tahoma", Font.BOLD, 11));

paymentTextField.setColumns(10);

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel)

.addContainerGap(81, Short.MAX\_VALUE))

.addComponent(saleAmountTextField, GroupLayout.DEFAULT\_SIZE, 145, Short.MAX\_VALUE)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel\_1)

.addContainerGap())

.addComponent(paymentTextField, GroupLayout.DEFAULT\_SIZE, 143, Short.MAX\_VALUE)

);

gl\_panel\_1.setVerticalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(saleAmountTextField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(lblNewLabel\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(paymentTextField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(45, Short.MAX\_VALUE))

);

panel\_1.setLayout(gl\_panel\_1);

contentPanel.setLayout(gl\_contentPanel);

{

JPanel buttonPane = new JPanel();

buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));

getContentPane().add(buttonPane, BorderLayout.SOUTH);

{

okButton = new JButton("OK");

buttonPane.add(okButton);

getRootPane().setDefaultButton(okButton);

}

{

cancelButton = new JButton("Cancel");

buttonPane.add(cancelButton);

}

}

initialize();

}//end construtor.

//Getters.

public double getSaleAmount() {

return saleAmount;

}

public double getPaymentAmount() {

return paymentAmount;

}

public JTextField getSaleAmountTextField() {

return saleAmountTextField;

}

public JTextField getPaymentTextField() {

return paymentTextField;

}

public JButton getFourButton() {

return fourButton;

}

public JButton getSevenButton() {

return sevenButton;

}

public JButton getFiveButton() {

return fiveButton;

}

public JButton getEightButton() {

return eightButton;

}

public JButton getNineButton() {

return nineButton;

}

public JButton getSixButton() {

return sixButton;

}

public JButton getOneButton() {

return oneButton;

}

public JButton getTwoButton() {

return twoButton;

}

public JButton getThreeButton() {

return threeButton;

}

public JButton getDotButton() {

return dotButton;

}

public JButton getZeroButton() {

return zeroButton;

}

public JButton getClearButton() {

return clearButton;

}

public JButton getOkButton() {

return okButton;

}

public JButton getCancelButton() {

return cancelButton;

}

//Setters.

public void setSaleAmount(double saleAmount) {

this.saleAmount = saleAmount;

}

public void setPaymentAmount(double paymentAmount) {

this.paymentAmount = paymentAmount;

}

public void setSaleAmountTextField(JTextField saleAmountTextField) {

this.saleAmountTextField = saleAmountTextField;

}

public void setPaymentTextField(JTextField paymentTextField) {

this.paymentTextField = paymentTextField;

}

public void setFourButton(JButton fourButton) {

this.fourButton = fourButton;

}

public void setSevenButton(JButton sevenButton) {

this.sevenButton = sevenButton;

}

public void setFiveButton(JButton fiveButton) {

this.fiveButton = fiveButton;

}

public void setEightButton(JButton eightButton) {

this.eightButton = eightButton;

}

public void setNineButton(JButton nineButton) {

this.nineButton = nineButton;

}

public void setSixButton(JButton sixButton) {

this.sixButton = sixButton;

}

public void setOneButton(JButton oneButton) {

this.oneButton = oneButton;

}

public void setTwoButton(JButton twoButton) {

this.twoButton = twoButton;

}

public void setThreeButton(JButton threeButton) {

this.threeButton = threeButton;

}

public void setDotButton(JButton dotButton) {

this.dotButton = dotButton;

}

public void setZeroButton(JButton zeroButton) {

this.zeroButton = zeroButton;

}

public void setClearButton(JButton clearButton) {

this.clearButton = clearButton;

}

public void setOkButton(JButton okButton) {

this.okButton = okButton;

}

public void setCancelButton(JButton cancelButton) {

this.cancelButton = cancelButton;

}

public void setDialogOk(boolean dialogOk){

this.dialogOk = dialogOk;

}

/\*\*

\* Initialize the event listener of the components.

\*\*/

public void initialize(){

getSevenButton().addActionListener(this);

getSevenButton().setActionCommand("seven");

getSevenButton().setFocusable(false);

getEightButton().addActionListener(this);

getEightButton().setActionCommand("eight");

getEightButton().setFocusable(false);

getNineButton().addActionListener(this);

getNineButton().setActionCommand("nine");

getNineButton().setFocusable(false);

getFourButton().addActionListener(this);

getFourButton().setActionCommand("four");

getFourButton().setFocusable(false);

getFiveButton().addActionListener(this);

getFiveButton().setActionCommand("five");

getFiveButton().setFocusable(false);

getSixButton().addActionListener(this);

getSixButton().setActionCommand("six");

getSixButton().setFocusable(false);

getOneButton().addActionListener(this);

getOneButton().setActionCommand("one");

getOneButton().setFocusable(false);

getTwoButton().addActionListener(this);

getTwoButton().setActionCommand("two");

getTwoButton().setFocusable(false);

getThreeButton().addActionListener(this);

getThreeButton().setActionCommand("three");

getThreeButton().setFocusable(false);

getZeroButton().addActionListener(this);

getZeroButton().setActionCommand("zero");

getZeroButton().setFocusable(false);

getClearButton().addActionListener(this);

getClearButton().setActionCommand("clear");

getClearButton().setFocusable(false);

getDotButton().addActionListener(this);

getDotButton().setActionCommand("dot");

getDotButton().setFocusable(false);

getOkButton().addActionListener(this);

getOkButton().setActionCommand("ok");

//getOkButton().setFocusable(false);

getCancelButton().addActionListener(this);

getCancelButton().setActionCommand("cancel");

//getCancelButton().setFocusable(false);

}//end initialize

/\*\*

\* Implemented ActionListener abstract methods.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

try{

Robot robot = new Robot();

switch(command){

case "seven":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_7);

robot.keyRelease(KeyEvent.VK\_7);

break;

case "eight":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_8);

robot.keyRelease(KeyEvent.VK\_8);

break;

case "nine":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_9);

robot.keyRelease(KeyEvent.VK\_9);

break;

case "four":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_4);

robot.keyRelease(KeyEvent.VK\_4);

break;

case "five":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_5);

robot.keyRelease(KeyEvent.VK\_5);

break;

case "six":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_6);

robot.keyRelease(KeyEvent.VK\_6);

break;

case "one":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_1);

robot.keyRelease(KeyEvent.VK\_1);

break;

case "two":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_2);

robot.keyRelease(KeyEvent.VK\_2);

break;

case "three":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_3);

robot.keyRelease(KeyEvent.VK\_3);

break;

case "zero":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_0);

robot.keyRelease(KeyEvent.VK\_0);

break;

case "clear":

this.getPaymentTextField().requestFocusInWindow();

this.getPaymentTextField().setText("");

break;

case "dot":

this.getPaymentTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_PERIOD);

robot.keyRelease(KeyEvent.VK\_PERIOD);

break;

case "ok":

/\*

\* Set a boolean attribute to true if the payment is valid and indicate to the calling class

\* to proceed to checkout.

\* \*/

if(isValidNumber(this.getPaymentTextField().getText())){

this.paymentAmount = (Double.valueOf(this.getPaymentTextField().getText()));

if(isPaymentEnough(this.paymentAmount) == true){

double changeAmount = (this.paymentAmount - this.saleAmount);

dialogOk = true;

JOptionPane.showMessageDialog(null, "Cash Tendered:\n" + this.format.format(this.paymentAmount) +

"\n-----------------------\nChange:\n" +

this.format.format(changeAmount),

"POS-Pi",

JOptionPane.INFORMATION\_MESSAGE);

this.setVisible(false);

}

else{//Show message if payment is less than sale amount.

JOptionPane.showMessageDialog(null, "Sorry. Insufficient payment amount.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

this.paymentTextField.setText("");

}//end if

}

else{//Show message if payment is not in a valid number format.

JOptionPane.showMessageDialog(null, "Sorry. Invalid payment amount.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

this.paymentTextField.setText("");

}//end if

break;

case "cancel":

dialogOk = false;

JOptionPane.showMessageDialog(null, "Cash payment cancelled.",

"POS-Pi",

JOptionPane.INFORMATION\_MESSAGE);

this.setVisible(false);

break;

}//end switch

robot = null;

}//end try

catch(Exception ex){

ex.printStackTrace();

}//end catch

}//end actionPerformed

/\*\*

\* Test if a String value is a valid double. Return true if it is, false otherwise.

\* @param number String

\* @return boolean

\*\*/

public boolean isValidNumber(String number){

try{

Double.parseDouble(number);

return true;

}

catch(NumberFormatException ex){

return false;

}//end try/catch

}//end isValidNumber

/\*\*

\* Returns true if payment is equal to or greater than total amount to be paid.

\* Return false otherwise.

\* @param paymentAmount double

\* @return boolean

\*\*/

public boolean isPaymentEnough(double paymentAmount){

if(paymentAmount >= this.saleAmount){

return true;

}

else{

return false;

}//end if

}//end isPaymentEnough

/\*\*

\* User had pressed the ok button.

\* @return boolean

\*\*/

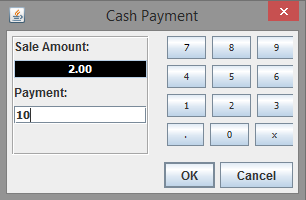
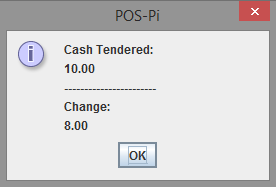
public boolean isDialogOk(){

return this.dialogOk;

}

}//end class.

**Screenshots:**

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class UnitSaleDialog**

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Window

java.awt.Dialog

javax.swing.JDialog

**pospi.UnitSaleDialog**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.awt.event.ItemListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class **UnitSaleDialog**extends javax.swing.JDialogimplements java.awt.event.ActionListener, java.awt.event.ItemListener

Author: Angelo Romel Lopez  
Description:  
[1] Constructs and initializes the components and graphical user interface of the unit-sale custom dialog screen.  
[2] Handles the logic and events of the unit-sale custom dialog screen.

**See Also:**

[Serialized Form](about:blank../serialized-form.html#pospi.UnitSaleDialog)

|  |
| --- |
| **Nested Class Summary** |

|  |
| --- |
| **Nested classes/interfaces inherited from class javax.swing.JDialog** |
| javax.swing.JDialog.AccessibleJDialog |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Dialog** |
| java.awt.Dialog.AccessibleAWTDialog, java.awt.Dialog.ModalExclusionType, java.awt.Dialog.ModalityType |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Window** |
| java.awt.Window.AccessibleAWTWindow, java.awt.Window.Type |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Container** |
| java.awt.Container.AccessibleAWTContainer |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy |

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class javax.swing.JDialog** |
| accessibleContext, rootPane, rootPaneCheckingEnabled |

|  |
| --- |
| **Fields inherited from class java.awt.Dialog** |
| DEFAULT\_MODALITY\_TYPE |

|  |
| --- |
| **Fields inherited from class java.awt.Component** |
| BOTTOM\_ALIGNMENT, CENTER\_ALIGNMENT, LEFT\_ALIGNMENT, RIGHT\_ALIGNMENT, TOP\_ALIGNMENT |

|  |
| --- |
| **Fields inherited from interface javax.swing.WindowConstants** |
| DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, EXIT\_ON\_CLOSE, HIDE\_ON\_CLOSE |

|  |
| --- |
| **Fields inherited from interface java.awt.image.ImageObserver** |
| ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**UnitSaleDialog**](about:blank../pospi/UnitSaleDialog.html#UnitSaleDialog(javax.swing.JFrame, boolean, java.lang.String, java.lang.String, java.lang.String))(javax.swing.JFrame frame, boolean modal, java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/UnitSaleDialog.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract methods. Handles the events when a button is pressed. |
| javax.swing.JButton | [**getCancelButton**](about:blank../pospi/UnitSaleDialog.html#getCancelButton())() |
| javax.swing.JButton | [**getClearButton**](about:blank../pospi/UnitSaleDialog.html#getClearButton())() |
| javax.swing.JPanel | [**getContentPanel**](about:blank../pospi/UnitSaleDialog.html#getContentPanel())() |
| javax.swing.JButton | [**getDotButton**](about:blank../pospi/UnitSaleDialog.html#getDotButton())() |
| javax.swing.JButton | [**getEightButton**](about:blank../pospi/UnitSaleDialog.html#getEightButton())() |
| javax.swing.JButton | [**getFiveButton**](about:blank../pospi/UnitSaleDialog.html#getFiveButton())() |
| javax.swing.JButton | [**getFourButton**](about:blank../pospi/UnitSaleDialog.html#getFourButton())() |
| void | [**getItemWithUOM**](about:blank../pospi/UnitSaleDialog.html#getItemWithUOM())()           Retrieves items from the products table that has a unit of measurement. |
| javax.swing.JButton | [**getNineButton**](about:blank../pospi/UnitSaleDialog.html#getNineButton())() |
| javax.swing.JButton | [**getOkButton**](about:blank../pospi/UnitSaleDialog.html#getOkButton())() |
| javax.swing.JButton | [**getOneButton**](about:blank../pospi/UnitSaleDialog.html#getOneButton())() |
| javax.swing.JComboBox | [**getProductCombo**](about:blank../pospi/UnitSaleDialog.html#getProductCombo())() |
| int | [**getProductID**](about:blank../pospi/UnitSaleDialog.html#getProductID())() |
| double | [**getQuantity**](about:blank../pospi/UnitSaleDialog.html#getQuantity())() |
| javax.swing.JLabel | [**getQuantityLabel**](about:blank../pospi/UnitSaleDialog.html#getQuantityLabel())() |
| javax.swing.JTextField | [**getQuantityTextField**](about:blank../pospi/UnitSaleDialog.html#getQuantityTextField())() |
| javax.swing.JButton | [**getSevenButton**](about:blank../pospi/UnitSaleDialog.html#getSevenButton())() |
| javax.swing.JButton | [**getSixButton**](about:blank../pospi/UnitSaleDialog.html#getSixButton())() |
| javax.swing.JButton | [**getThreeButton**](about:blank../pospi/UnitSaleDialog.html#getThreeButton())() |
| javax.swing.JButton | [**getTwoButton**](about:blank../pospi/UnitSaleDialog.html#getTwoButton())() |
| javax.swing.JButton | [**getZeroButton**](about:blank../pospi/UnitSaleDialog.html#getZeroButton())() |
| void | [**initialize**](about:blank../pospi/UnitSaleDialog.html#initialize())()           Initialize the event listener of the components. |
| boolean | [**isDialogOk**](about:blank../pospi/UnitSaleDialog.html#isDialogOk())() |
| boolean | [**isValidNumber**](about:blank../pospi/UnitSaleDialog.html#isValidNumber(java.lang.String))(java.lang.String number)           Test if a String value is a valid double. |
| void | [**itemStateChanged**](about:blank../pospi/UnitSaleDialog.html#itemStateChanged(java.awt.event.ItemEvent))(java.awt.event.ItemEvent e)           Implemented ItemListener abstract methods. Handles the event when an item is selected from the combobox. |
| java.sql.ResultSet | [**runSelect**](about:blank../pospi/UnitSaleDialog.html#runSelect(java.lang.String))(java.lang.String selectStatement)           Executes a select query on the database to fetch records that has a uom\_id instead of a barcode. |
| void | [**setCancelButton**](about:blank../pospi/UnitSaleDialog.html#setCancelButton(javax.swing.JButton))(javax.swing.JButton cancelButton) |
| void | [**setClearButton**](about:blank../pospi/UnitSaleDialog.html#setClearButton(javax.swing.JButton))(javax.swing.JButton backButton) |
| void | [**setDialogOk**](about:blank../pospi/UnitSaleDialog.html#setDialogOk(boolean))(boolean dialogOk) |
| void | [**setDotButton**](about:blank../pospi/UnitSaleDialog.html#setDotButton(javax.swing.JButton))(javax.swing.JButton dotButton) |
| void | [**setEightButton**](about:blank../pospi/UnitSaleDialog.html#setEightButton(javax.swing.JButton))(javax.swing.JButton eightButton) |
| void | [**setFiveButton**](about:blank../pospi/UnitSaleDialog.html#setFiveButton(javax.swing.JButton))(javax.swing.JButton fiveButton) |
| void | [**setFourButton**](about:blank../pospi/UnitSaleDialog.html#setFourButton(javax.swing.JButton))(javax.swing.JButton fourButton) |
| void | [**setNineButton**](about:blank../pospi/UnitSaleDialog.html#setNineButton(javax.swing.JButton))(javax.swing.JButton nineButton) |
| void | [**setOkButton**](about:blank../pospi/UnitSaleDialog.html#setOkButton(javax.swing.JButton))(javax.swing.JButton okButton) |
| void | [**setOneButton**](about:blank../pospi/UnitSaleDialog.html#setOneButton(javax.swing.JButton))(javax.swing.JButton oneButton) |
| void | [**setProductCombo**](about:blank../pospi/UnitSaleDialog.html#setProductCombo(javax.swing.JComboBox))(javax.swing.JComboBox productCombo) |
| void | [**setQuantity**](about:blank../pospi/UnitSaleDialog.html#setQuantity(double))(double quantity) |
| void | [**setQuantityLabel**](about:blank../pospi/UnitSaleDialog.html#setQuantityLabel(javax.swing.JLabel))(javax.swing.JLabel quantityLabel) |
| void | [**setQuantityTextField**](about:blank../pospi/UnitSaleDialog.html#setQuantityTextField(javax.swing.JTextField))(javax.swing.JTextField quantityTextField) |
| void | [**setSevenButton**](about:blank../pospi/UnitSaleDialog.html#setSevenButton(javax.swing.JButton))(javax.swing.JButton sevenButton) |
| void | [**setSixButton**](about:blank../pospi/UnitSaleDialog.html#setSixButton(javax.swing.JButton))(javax.swing.JButton sixButton) |
| void | [**setThreeButton**](about:blank../pospi/UnitSaleDialog.html#setThreeButton(javax.swing.JButton))(javax.swing.JButton threeButton) |
| void | [**setTwoButton**](about:blank../pospi/UnitSaleDialog.html#setTwoButton(javax.swing.JButton))(javax.swing.JButton twoButton) |
| void | [**setZeroButton**](about:blank../pospi/UnitSaleDialog.html#setZeroButton(javax.swing.JButton))(javax.swing.JButton zeroButton) |

|  |
| --- |
| **Methods inherited from class javax.swing.JDialog** |
| addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update |

|  |
| --- |
| **Methods inherited from class java.awt.Dialog** |
| addNotify, getModalityType, getTitle, hide, isModal, isResizable, isUndecorated, setBackground, setModal, setModalityType, setOpacity, setResizable, setShape, setTitle, setUndecorated, setVisible, show, toBack |

|  |
| --- |
| **Methods inherited from class java.awt.Window** |
| addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBackground, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOpacity, getOwnedWindows, getOwner, getOwnerlessWindows, getShape, getToolkit, getType, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isAutoRequestFocus, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isOpaque, isShowing, isValidateRoot, pack, paint, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeNotify, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setAutoRequestFocus, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImage, setIconImages, setLocation, setLocation, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setType, toFront |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusDownCycle, validate, validateTree |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resize, resize, revalidate, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusBackward, transferFocusUpCycle |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### UnitSaleDialog

public **UnitSaleDialog**(javax.swing.JFrame frame,

boolean modal,

java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor.

**Parameters:**

frame - JFrame

modal - boolean

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract methods.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### getCancelButton

public javax.swing.JButton **getCancelButton**()

### getClearButton

public javax.swing.JButton **getClearButton**()

### getContentPanel

public javax.swing.JPanel **getContentPanel**()

### getDotButton

public javax.swing.JButton **getDotButton**()

### getEightButton

public javax.swing.JButton **getEightButton**()

### getFiveButton

public javax.swing.JButton **getFiveButton**()

### getFourButton

public javax.swing.JButton **getFourButton**()

### getItemWithUOM

public void **getItemWithUOM**()

Retrieves items from the products table that has a unit of measurement.

### getNineButton

public javax.swing.JButton **getNineButton**()

### getOkButton

public javax.swing.JButton **getOkButton**()

### getOneButton

public javax.swing.JButton **getOneButton**()

### getProductCombo

public javax.swing.JComboBox **getProductCombo**()

### getProductID

public int **getProductID**()

### getQuantity

public double **getQuantity**()

### getQuantityLabel

public javax.swing.JLabel **getQuantityLabel**()

### getQuantityTextField

public javax.swing.JTextField **getQuantityTextField**()

### getSevenButton

public javax.swing.JButton **getSevenButton**()

### getSixButton

public javax.swing.JButton **getSixButton**()

### getThreeButton

public javax.swing.JButton **getThreeButton**()

### getTwoButton

public javax.swing.JButton **getTwoButton**()

### getZeroButton

public javax.swing.JButton **getZeroButton**()

### initialize

public void **initialize**()

Initialize the event listener of the components.

### isDialogOk

public boolean **isDialogOk**()

### isValidNumber

public boolean **isValidNumber**(java.lang.String number)

Test if a String value is a valid double. Return true if it is, false otherwise.

**Parameters:**

number - String

**Returns:**

boolean

### itemStateChanged

public void **itemStateChanged**(java.awt.event.ItemEvent e)

Implemented ItemListener abstract methods.  
Handles the event when an item is selected from the combobox.

**Specified by:**

itemStateChanged in interface java.awt.event.ItemListener

**Parameters:**

e - ItemEvent

### runSelect

public java.sql.ResultSet **runSelect**(java.lang.String selectStatement)

Executes a select query on the database to fetch records that has a uom\_id instead of a barcode.

**Parameters:**

selectStatement - String

**Returns:**

ResultSet

### setCancelButton

public void **setCancelButton**(javax.swing.JButton cancelButton)

### setClearButton

public void **setClearButton**(javax.swing.JButton backButton)

### setDialogOk

public void **setDialogOk**(boolean dialogOk)

### setDotButton

public void **setDotButton**(javax.swing.JButton dotButton)

### setEightButton

public void **setEightButton**(javax.swing.JButton eightButton)

### setFiveButton

public void **setFiveButton**(javax.swing.JButton fiveButton)

### setFourButton

public void **setFourButton**(javax.swing.JButton fourButton)

### setNineButton

public void **setNineButton**(javax.swing.JButton nineButton)

### setOkButton

public void **setOkButton**(javax.swing.JButton okButton)

### setOneButton

public void **setOneButton**(javax.swing.JButton oneButton)

### setProductCombo

public void **setProductCombo**(javax.swing.JComboBox productCombo)

### setQuantity

public void **setQuantity**(double quantity)

### setQuantityLabel

public void **setQuantityLabel**(javax.swing.JLabel quantityLabel)

### setQuantityTextField

public void **setQuantityTextField**(javax.swing.JTextField quantityTextField)

### setSevenButton

public void **setSevenButton**(javax.swing.JButton sevenButton)

### setSixButton

public void **setSixButton**(javax.swing.JButton sixButton)

### setThreeButton

public void **setThreeButton**(javax.swing.JButton threeButton)

### setTwoButton

public void **setTwoButton**(javax.swing.JButton twoButton)

### setZeroButton

public void **setZeroButton**(javax.swing.JButton zeroButton)

**getItemWithUOM() – Sequence Diagram**

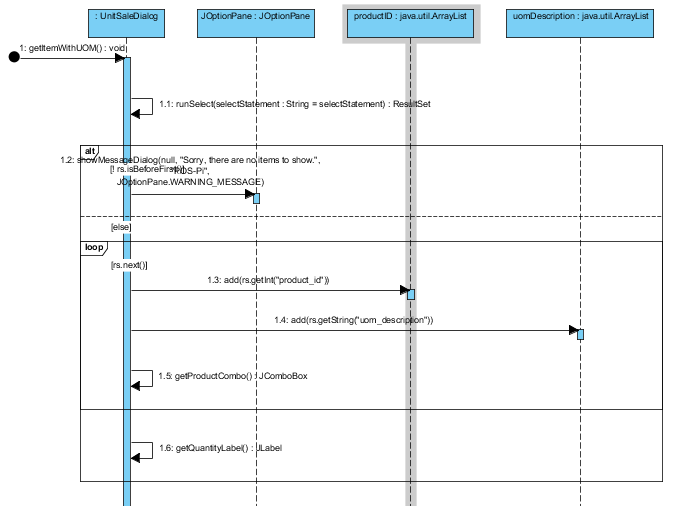
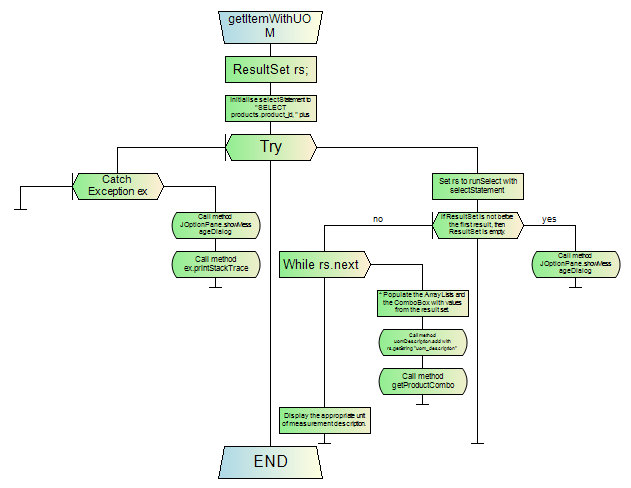


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**getItemWithUOM() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**UnitSaleDialog class code listing:**

package pospi;

import java.awt.BorderLayout;

import java.awt.FlowLayout;

import java.awt.Robot;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.border.EtchedBorder;

import javax.swing.JComboBox;

import java.awt.Font;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JTextField;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.event.KeyEvent;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.UIManager;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: <br>[1] Constructs and initializes the components and graphical user interface

\* of the unit-sale custom dialog screen.<br>[2] Handles the logic and events of the unit-sale custom dialog screen.

\* \*/

@SuppressWarnings("serial")

public class UnitSaleDialog extends JDialog implements ActionListener, ItemListener{

private String dataConnection;

private String dataUser;

private String dataPassword;

private boolean dialogOk;

private double quantity;

private ArrayList <Integer> productID;

private ArrayList <String> uomDescription;

private final JPanel contentPanel = new JPanel();

private JTextField quantityTextField;

@SuppressWarnings("rawtypes")

private JComboBox productCombo;

private JLabel quantityLabel;

private JButton fourButton;

private JButton sevenButton;

private JButton eightButton;

private JButton nineButton;

private JButton fiveButton;

private JButton sixButton;

private JButton oneButton;

private JButton twoButton;

private JButton threeButton;

private JButton dotButton;

private JButton zeroButton;

private JButton clearButton;

private JButton okButton;

private JButton cancelButton;

/\*\*

\* Constructor.

\* @param frame JFrame

\* @param modal boolean

\* @param connection String

\* @param user String

\* @param password String

\*\*/

@SuppressWarnings("rawtypes")

public UnitSaleDialog(JFrame frame, boolean modal, String connection, String user, String password) {

super(frame, "Select Product", modal);

this.dataConnection = connection;

this.dataPassword = password;

this.dataUser = user;

productID = new ArrayList <Integer>();

uomDescription = new ArrayList <String>();

//setTitle("Select Product");

setBounds(100, 100, 316, 212);

getContentPane().setLayout(new BorderLayout());

contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

getContentPane().add(contentPanel, BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.setBackground(UIManager.getColor("Button.background"));

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(null);

fourButton = new JButton("4");

fourButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sevenButton = new JButton("7");

sevenButton.setForeground(Color.BLACK);

sevenButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

eightButton = new JButton("8");

eightButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

nineButton = new JButton("9");

nineButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

fiveButton = new JButton("5");

fiveButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

sixButton = new JButton("6");

sixButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

oneButton = new JButton("1");

oneButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

twoButton = new JButton("2");

twoButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

threeButton = new JButton("3");

threeButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

dotButton = new JButton(".");

dotButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

zeroButton = new JButton("0");

zeroButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

clearButton = new JButton("x");

clearButton.setFont(new Font("Tahoma", Font.PLAIN, 10));

okButton = new JButton("OK");

cancelButton = new JButton("Cancel");

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING, false)

.addComponent(fourButton, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(sevenButton, Alignment.LEADING))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addComponent(fiveButton)

.addComponent(eightButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addComponent(nineButton)

.addComponent(sixButton)))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(oneButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(twoButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(threeButton))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(dotButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(zeroButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(clearButton)))))

.addContainerGap())

);

gl\_panel\_1.setVerticalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(sevenButton)

.addComponent(eightButton)

.addComponent(nineButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(fourButton)

.addComponent(fiveButton)

.addComponent(sixButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(oneButton)

.addComponent(twoButton)

.addComponent(threeButton))

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(dotButton)

.addComponent(zeroButton)

.addComponent(clearButton))

.addGap(59)

.addContainerGap(GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

panel\_1.setLayout(gl\_panel\_1);

GroupLayout gl\_contentPanel = new GroupLayout(contentPanel);

gl\_contentPanel.setHorizontalGroup(

gl\_contentPanel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_contentPanel.createSequentialGroup()

.addComponent(panel, GroupLayout.PREFERRED\_SIZE, 151, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(panel\_1, GroupLayout.PREFERRED\_SIZE, 138, Short.MAX\_VALUE)

.addContainerGap())

);

gl\_contentPanel.setVerticalGroup(

gl\_contentPanel.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, gl\_contentPanel.createSequentialGroup()

.addGroup(gl\_contentPanel.createParallelGroup(Alignment.TRAILING)

.addComponent(panel, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 119, Short.MAX\_VALUE)

.addComponent(panel\_1, GroupLayout.PREFERRED\_SIZE, 119, Short.MAX\_VALUE))

.addContainerGap())

);

JLabel lblProducts = new JLabel("Products");

productCombo = new JComboBox();

quantityLabel = new JLabel("Enter quantity in: ");

quantityTextField = new JTextField();

quantityTextField.setForeground(Color.BLACK);

quantityTextField.setBackground(Color.WHITE);

quantityTextField.setFont(new Font("Tahoma", Font.BOLD, 12));

quantityTextField.setColumns(10);

GroupLayout gl\_panel = new GroupLayout(panel);

gl\_panel.setHorizontalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.TRAILING)

.addComponent(lblProducts, Alignment.LEADING)

.addComponent(quantityLabel, Alignment.LEADING))

.addContainerGap(87, Short.MAX\_VALUE))

.addComponent(productCombo, 0, 147, Short.MAX\_VALUE)

.addComponent(quantityTextField, GroupLayout.DEFAULT\_SIZE, 147, Short.MAX\_VALUE)

);

gl\_panel.setVerticalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addComponent(lblProducts)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(productCombo, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(quantityLabel)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(quantityTextField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(28, Short.MAX\_VALUE))

);

panel.setLayout(gl\_panel);

contentPanel.setLayout(gl\_contentPanel);

{

JPanel buttonPane = new JPanel();

buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));

getContentPane().add(buttonPane, BorderLayout.SOUTH);

{

buttonPane.add(okButton);

//getRootPane().setDefaultButton(okButton);

}

{

buttonPane.add(cancelButton);

}

}

initialize();

getItemWithUOM();

}//end constructor

//Getters.

public JPanel getContentPanel() {

return contentPanel;

}

public JTextField getQuantityTextField() {

return quantityTextField;

}

@SuppressWarnings("rawtypes")

public JComboBox getProductCombo() {

return productCombo;

}

public JButton getFourButton() {

return fourButton;

}

public JButton getSevenButton() {

return sevenButton;

}

public JButton getEightButton() {

return eightButton;

}

public JButton getNineButton() {

return nineButton;

}

public JButton getFiveButton() {

return fiveButton;

}

public JButton getSixButton() {

return sixButton;

}

public JButton getOneButton() {

return oneButton;

}

public JButton getTwoButton() {

return twoButton;

}

public JButton getThreeButton() {

return threeButton;

}

public JButton getDotButton() {

return dotButton;

}

public JButton getZeroButton() {

return zeroButton;

}

public JButton getClearButton() {

return clearButton;

}

public JButton getOkButton(){

return okButton;

}

public JButton getCancelButton(){

return cancelButton;

}

public boolean isDialogOk() {

return dialogOk;

}

public double getQuantity(){

return this.quantity;

}

public JLabel getQuantityLabel(){

return this.quantityLabel;

}

public int getProductID(){

return this.productID.get(getProductCombo().getSelectedIndex());

}

//Setters

public void setQuantityTextField(JTextField quantityTextField) {

this.quantityTextField = quantityTextField;

}

public void setProductCombo(@SuppressWarnings("rawtypes") JComboBox productCombo) {

this.productCombo = productCombo;

}

public void setFourButton(JButton fourButton) {

this.fourButton = fourButton;

}

public void setSevenButton(JButton sevenButton) {

this.sevenButton = sevenButton;

}

public void setEightButton(JButton eightButton) {

this.eightButton = eightButton;

}

public void setNineButton(JButton nineButton) {

this.nineButton = nineButton;

}

public void setFiveButton(JButton fiveButton) {

this.fiveButton = fiveButton;

}

public void setSixButton(JButton sixButton) {

this.sixButton = sixButton;

}

public void setOneButton(JButton oneButton) {

this.oneButton = oneButton;

}

public void setTwoButton(JButton twoButton) {

this.twoButton = twoButton;

}

public void setThreeButton(JButton threeButton) {

this.threeButton = threeButton;

}

public void setDotButton(JButton dotButton) {

this.dotButton = dotButton;

}

public void setZeroButton(JButton zeroButton) {

this.zeroButton = zeroButton;

}

public void setClearButton(JButton backButton) {

this.clearButton = backButton;

}

public void setOkButton(JButton okButton){

this.okButton = okButton;

}

public void setCancelButton(JButton cancelButton){

this.cancelButton = cancelButton;

}

public void setDialogOk(boolean dialogOk) {

this.dialogOk = dialogOk;

}

public void setQuantity(double quantity){

this.quantity = quantity;

}

public void setQuantityLabel(JLabel quantityLabel){

this.quantityLabel = quantityLabel;

}

/\*\*

\* Initialize the event listener of the components.

\*\*/

public void initialize(){

getSevenButton().addActionListener(this);

getSevenButton().setActionCommand("seven");

getSevenButton().setFocusable(false);

getEightButton().addActionListener(this);

getEightButton().setActionCommand("eight");

getEightButton().setFocusable(false);

getNineButton().addActionListener(this);

getNineButton().setActionCommand("nine");

getNineButton().setFocusable(false);

getFourButton().addActionListener(this);

getFourButton().setActionCommand("four");

getFourButton().setFocusable(false);

getFiveButton().addActionListener(this);

getFiveButton().setActionCommand("five");

getFiveButton().setFocusable(false);

getSixButton().addActionListener(this);

getSixButton().setActionCommand("six");

getSixButton().setFocusable(false);

getOneButton().addActionListener(this);

getOneButton().setActionCommand("one");

getOneButton().setFocusable(false);

getTwoButton().addActionListener(this);

getTwoButton().setActionCommand("two");

getTwoButton().setFocusable(false);

getThreeButton().addActionListener(this);

getThreeButton().setActionCommand("three");

getThreeButton().setFocusable(false);

getZeroButton().addActionListener(this);

getZeroButton().setActionCommand("zero");

getZeroButton().setFocusable(false);

getClearButton().addActionListener(this);

getClearButton().setActionCommand("clear");

getClearButton().setFocusable(false);

getDotButton().addActionListener(this);

getDotButton().setActionCommand("dot");

getDotButton().setFocusable(false);

getOkButton().addActionListener(this);

getOkButton().setActionCommand("ok");

//getOkButton().setFocusable(false);

getCancelButton().addActionListener(this);

getCancelButton().setActionCommand("cancel");

//getCancelButton().setFocusable(false);

getProductCombo().addItemListener(this);

}//end initialize

/\*\*

\* Implemented ActionListener abstract methods.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

try{

Robot robot = new Robot();

switch(command){

case "seven":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_7);

robot.keyRelease(KeyEvent.VK\_7);

break;

case "eight":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_8);

robot.keyRelease(KeyEvent.VK\_8);

break;

case "nine":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_9);

robot.keyRelease(KeyEvent.VK\_9);

break;

case "four":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_4);

robot.keyRelease(KeyEvent.VK\_4);

break;

case "five":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_5);

robot.keyRelease(KeyEvent.VK\_5);

break;

case "six":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_6);

robot.keyRelease(KeyEvent.VK\_6);

break;

case "one":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_1);

robot.keyRelease(KeyEvent.VK\_1);

break;

case "two":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_2);

robot.keyRelease(KeyEvent.VK\_2);

break;

case "three":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_3);

robot.keyRelease(KeyEvent.VK\_3);

break;

case "zero":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_0);

robot.keyRelease(KeyEvent.VK\_0);

break;

case "clear":

getQuantityTextField().requestFocusInWindow();

getQuantityTextField().setText("");

break;

case "dot":

getQuantityTextField().requestFocusInWindow();

//Simulate a key press

robot.keyPress(KeyEvent.VK\_PERIOD);

robot.keyRelease(KeyEvent.VK\_PERIOD);

break;

case "ok":

/\*

\* There are no selected items in the ComboBox, or the ComboBox is empty.

\* \*/

if(getProductCombo().getSelectedIndex() == -1){

JOptionPane.showMessageDialog(null, "You have pressed ok.\nNo item was selected\n" +

"Please press cancel.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

dialogOk = false;

this.setVisible(false);

}

else{

/\*

\* An item was selected in the ComboBox. Set a boolean attribute to true to

\* indicate to the calling class that an item was selected and the ok button

\* was pressed.

\* \*/

if(isValidNumber(getQuantityTextField().getText())){

this.quantity = Double.valueOf(getQuantityTextField().getText());

dialogOk = true;

this.setVisible(false);

}

else{

JOptionPane.showMessageDialog(null, "The quantity is not\na valid number.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}//end if

}//end if

break;

case "cancel":

dialogOk = false;

this.setVisible(false);

break;

}//end switch

robot = null;

}//end try

catch(Exception ex){

ex.printStackTrace();

}//end catch

}//end actionPerformed

/\*\*

\* Implemented ItemListener abstract methods.<br>Handles the event when an item is selected from the combobox.

\* @param e ItemEvent

\*\*/

public void itemStateChanged(ItemEvent e){

if(e.getSource() == getProductCombo()){

//Display the appropriate unit of measurement description.

getQuantityLabel().setText("Enter quantity in: " + uomDescription.get(getProductCombo().getSelectedIndex()));

}//end if

}//end itemStateChanged

/\*\*

\* Executes a select query on the database to fetch records that has a uom\_id instead of a barcode.

\* @param selectStatement String

\* @return ResultSet

\*\*/

public ResultSet runSelect(String selectStatement){

Connection dbConnection = null;

Statement dbStatement = null;

ResultSet rs = null;

try{

dbConnection = DriverManager.getConnection(dataConnection, dataUser, dataPassword);

dbStatement = dbConnection.createStatement();

rs = dbStatement.executeQuery(selectStatement);

}//end try

catch(SQLException sqlEx){

JOptionPane.showMessageDialog(null, sqlEx.toString(),

"Error in connecting to database.",

JOptionPane.WARNING\_MESSAGE);

}//end catch

catch(Exception eEx){

JOptionPane.showMessageDialog(null, eEx.toString(),

"Error in connecting to database",

JOptionPane.WARNING\_MESSAGE);

}//end catch

finally{

dbConnection = null;

dbStatement = null;

}//end finally

return rs;

}//end runSelect

/\*\*

\* Retrieves items from the products table that has a unit of measurement.

\*\*/

@SuppressWarnings("unchecked")

public void getItemWithUOM(){

ResultSet rs;

String selectStatement = "SELECT products.product\_id, " +

"products.uom\_id, " +

"unit\_of\_measurement.description as uom\_description, " +

"unit\_of\_measurement.uom\_code as uom\_code, " +

"products.description as product\_description " +

"FROM products LEFT JOIN unit\_of\_measurement " +

"ON products.uom\_id = unit\_of\_measurement.uom\_id " +

"WHERE uom\_code != 'N/A'";

try{

rs = runSelect(selectStatement);

//if ResultSet is not before the first result, then ResultSet is empty.

if(! rs.isBeforeFirst()){

JOptionPane.showMessageDialog(null, "Sorry, there are no items to show.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

//ResultSet is before the first result, record/s found.

else{

while(rs.next()){

/\*

\* Populate the ArrayLists and the ComboBox with values from the result set.

\* \*/

productID.add(rs.getInt("product\_id"));

uomDescription.add(rs.getString("uom\_description"));

getProductCombo().addItem(rs.getString("product\_description"));

}//end while

//Display the appropriate unit of measurement description.

getQuantityLabel().setText("Enter quantity in: " + uomDescription.get(0));

}//end if

}//end try

catch(Exception ex){

JOptionPane.showMessageDialog(null, ex.toString(),

"Error in fetching records.",

JOptionPane.WARNING\_MESSAGE);

ex.printStackTrace();

}//end catch

}//end getItemWithUOM

/\*\*

\* Test if a String value is a valid double. Return true if it is, false otherwise.

\* @param number String

\* @return boolean

\*\*/

public boolean isValidNumber(String number){

try{

Double.parseDouble(number);

return true;

}

catch(NumberFormatException ex){

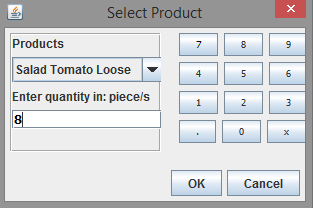
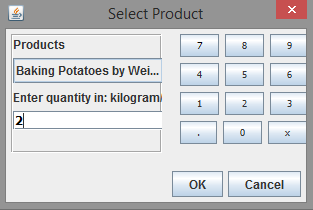
return false;

}//end try/catch

}//end isValidNumber

}//end class

**Screenshot:**



**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class ProductOrder**

java.lang.Object

**pospi.ProductOrder**

public class **ProductOrder**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: Holds the product model that includes the VAT rate and quantity ordered. Used to display product information on the item display TextArea.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ProductOrder**](about:blank../pospi/ProductOrder.html#ProductOrder())()           Constructor. |  |
| [**ProductOrder**](about:blank../pospi/ProductOrder.html#ProductOrder(int, java.lang.String, int, java.lang.String, java.lang.String, double, byte, double))(int productID, java.lang.String barcode, int uomID, java.lang.String uomCode, java.lang.String description, double price, byte vatRate, double quantity)           Constructor, overloaded |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| java.lang.String | [**getBarcode**](about:blank../pospi/ProductOrder.html#getBarcode())() |
| java.lang.String | [**getDescription**](about:blank../pospi/ProductOrder.html#getDescription())() |
| double | [**getPrice**](about:blank../pospi/ProductOrder.html#getPrice())() |
| int | [**getProductID**](about:blank../pospi/ProductOrder.html#getProductID())() |
| double | [**getQuantity**](about:blank../pospi/ProductOrder.html#getQuantity())() |
| java.lang.String | [**getUomCode**](about:blank../pospi/ProductOrder.html#getUomCode())() |
| int | [**getUomID**](about:blank../pospi/ProductOrder.html#getUomID())() |
| byte | [**getVatRate**](about:blank../pospi/ProductOrder.html#getVatRate())() |
| void | [**setBarcode**](about:blank../pospi/ProductOrder.html#setBarcode(java.lang.String))(java.lang.String barcode) |
| void | [**setDescription**](about:blank../pospi/ProductOrder.html#setDescription(java.lang.String))(java.lang.String description) |
| void | [**setPrice**](about:blank../pospi/ProductOrder.html#setPrice(double))(double price) |
| void | [**setProductID**](about:blank../pospi/ProductOrder.html#setProductID(int))(int productID) |
| void | [**setQuantity**](about:blank../pospi/ProductOrder.html#setQuantity(double))(double quantity) |
| void | [**setUomCode**](about:blank../pospi/ProductOrder.html#setUomCode(java.lang.String))(java.lang.String uomCode) |
| void | [**setUomID**](about:blank../pospi/ProductOrder.html#setUomID(int))(int uomID) |
| void | [**setVatRate**](about:blank../pospi/ProductOrder.html#setVatRate(byte))(byte vatRate) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### ProductOrder

public **ProductOrder**()

Constructor.

### ProductOrder

public **ProductOrder**(int productID,

java.lang.String barcode,

int uomID,

java.lang.String uomCode,

java.lang.String description,

double price,

byte vatRate,

double quantity)

Constructor, overloaded

**Parameters:**

productID - int

barcode - String

uomID - uomID

uomCode - uomCode

description - String

price - double

vatRate - byte

quantity - quantity

|  |
| --- |
| **Method Detail** |

### getBarcode

public java.lang.String **getBarcode**()

### getDescription

public java.lang.String **getDescription**()

### getPrice

public double **getPrice**()

### getProductID

public int **getProductID**()

### getQuantity

public double **getQuantity**()

### getUomCode

public java.lang.String **getUomCode**()

### getUomID

public int **getUomID**()

### getVatRate

public byte **getVatRate**()

### setBarcode

public void **setBarcode**(java.lang.String barcode)

### setDescription

public void **setDescription**(java.lang.String description)

### setPrice

public void **setPrice**(double price)

### setProductID

public void **setProductID**(int productID)

### setQuantity

public void **setQuantity**(double quantity)

### setUomCode

public void **setUomCode**(java.lang.String uomCode)

### setUomID

public void **setUomID**(int uomID)

### setVatRate

public void **setVatRate**(byte vatRate)

**ProductOrder class code listing:**

package pospi;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Holds the product model that includes the VAT rate and quantity ordered. Used to display

\* product information on the item display TextArea.

\* \*/

public class ProductOrder{

private int productID;

private String barcode;

private int uomID;

private String uomCode;

private String description;

private double price;

private byte vatRate;

private double quantity;/\*

\*declared as double, quantity can also depend on

\*unit of measurement such as gram eg. 0.5grams.

\*/

/\*\*

\* Constructor.

\*\*/

public ProductOrder(){

productID = 0;

barcode = "";

uomID = 0;

uomCode = "";

description = "";

price = 0D;

vatRate = 0;

quantity = 0d;

}//end constructor

/\*\*

\* Constructor, overloaded

\* @param productID int

\* @param barcode String

\* @param uomID uomID

\* @param uomCode uomCode

\* @param description String

\* @param price double

\* @param vatRate byte

\* @param quantity quantity

\*\*/

public ProductOrder(int productID, String barcode, int uomID, String uomCode, String description,

double price, byte vatRate, double quantity){

this.productID = productID;

this.barcode = barcode;

this.uomID = uomID;

this.uomCode = uomCode;

this.description = description;

this.price = price;

this.vatRate = vatRate;

this.quantity = quantity;

}//end constructor

//Getters.

public int getProductID() {

return productID;

}

public String getBarcode() {

return barcode;

}

public int getUomID() {

return uomID;

}

public String getUomCode(){

return this.uomCode;

}

public String getDescription() {

return description;

}

public double getPrice() {

return price;

}

public byte getVatRate(){

return vatRate;

}

public double getQuantity(){

return quantity;

}

//Setters.

public void setProductID(int productID) {

this.productID = productID;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public void setUomID(int uomID) {

this.uomID = uomID;

}

public void setUomCode(String uomCode){

this.uomCode = uomCode;

}

public void setDescription(String description) {

this.description = description;

}

public void setPrice(double price) {

this.price = price;

}

public void setVatRate(byte vatRate){

this.vatRate = vatRate;

}

public void setQuantity(double quantity){

this.quantity = quantity;

}

}//end class

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class ProductView**

java.lang.Object

**pospi.ProductView**

public class **ProductView**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: Constructs and initializes the components and graphical user interface of the user product view screen.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ProductView**](about:blank../pospi/ProductView.html#ProductView())()           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**createTable**](about:blank../pospi/ProductView.html#createTable(int))(int numberOfRows)           Constructs the table that holds the product list. |
| javax.swing.JButton | [**getDeleteButton**](about:blank../pospi/ProductView.html#getDeleteButton())() |
| javax.swing.JFrame | [**getFrmProducts**](about:blank../pospi/ProductView.html#getFrmProducts())() |
| javax.swing.JButton | [**getInsertButton**](about:blank../pospi/ProductView.html#getInsertButton())() |
| javax.swing.JTable | [**getProductViewTable**](about:blank../pospi/ProductView.html#getProductViewTable())() |
| javax.swing.JButton | [**getUpdateButton**](about:blank../pospi/ProductView.html#getUpdateButton())() |
| void | [**setDeleteButton**](about:blank../pospi/ProductView.html#setDeleteButton(javax.swing.JButton))(javax.swing.JButton deleteButton) |
| void | [**setFrmProducts**](about:blank../pospi/ProductView.html#setFrmProducts(javax.swing.JFrame))(javax.swing.JFrame frmProducts) |
| void | [**setInsertButton**](about:blank../pospi/ProductView.html#setInsertButton(javax.swing.JButton))(javax.swing.JButton insertButton) |
| void | [**setProductViewTable**](about:blank../pospi/ProductView.html#setProductViewTable(javax.swing.JTable))(javax.swing.JTable productViewTable) |
| void | [**setUpdateButton**](about:blank../pospi/ProductView.html#setUpdateButton(javax.swing.JButton))(javax.swing.JButton updateButton) |
| void | [**showGUI**](about:blank../pospi/ProductView.html#showGUI())()           Make the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### ProductView

public **ProductView**()

Constructor.

|  |
| --- |
| **Method Detail** |

### createTable

public void **createTable**(int numberOfRows)

Constructs the table that holds the product list.  
This method is needed by the controller class to construct a JTable object with the correct number of rows from the database.

**Parameters:**

numberOfRows - int

### getDeleteButton

public javax.swing.JButton **getDeleteButton**()

### getFrmProducts

public javax.swing.JFrame **getFrmProducts**()

### getInsertButton

public javax.swing.JButton **getInsertButton**()

### getProductViewTable

public javax.swing.JTable **getProductViewTable**()

### getUpdateButton

public javax.swing.JButton **getUpdateButton**()

### setDeleteButton

public void **setDeleteButton**(javax.swing.JButton deleteButton)

### setFrmProducts

public void **setFrmProducts**(javax.swing.JFrame frmProducts)

### setInsertButton

public void **setInsertButton**(javax.swing.JButton insertButton)

### setProductViewTable

public void **setProductViewTable**(javax.swing.JTable productViewTable)

### setUpdateButton

public void **setUpdateButton**(javax.swing.JButton updateButton)

### showGUI

public void **showGUI**()

Make the frame visible.

**ProductView class code listing:**

package pospi;

import javax.swing.JFrame;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JPanel;

import javax.swing.UIManager;

import javax.swing.JTable;

import javax.swing.LayoutStyle.ComponentPlacement;

import java.awt.FlowLayout;

import javax.swing.JButton;

import javax.swing.JScrollPane;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Constructs and initializes the components and graphical user interface

\* of the user product view screen.

\* \*/

public class ProductView {

private JFrame mainFrame;

private JTable productViewTable;

private JButton insertButton;

private JButton updateButton;

private JButton deleteButton;

private JScrollPane scrollPane;

/\*\*

\* Constructor.

\*/

public ProductView() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

mainFrame = new JFrame();

mainFrame.setTitle("Products");

mainFrame.setBounds(100, 100, 320, 240);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setBackground(UIManager.getColor("Button.background"));

JPanel panel\_1 = new JPanel();

GroupLayout groupLayout = new GroupLayout(mainFrame.getContentPane());

groupLayout.setHorizontalGroup(

groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, groupLayout.createSequentialGroup()

.addContainerGap()

.addGroup(groupLayout.createParallelGroup(Alignment.TRAILING)

.addComponent(panel, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE)

.addComponent(panel\_1, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE))

.addContainerGap())

);

groupLayout.setVerticalGroup(

groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, groupLayout.createSequentialGroup()

.addContainerGap()

.addComponent(panel, GroupLayout.DEFAULT\_SIZE, 151, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(panel\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

);

panel\_1.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

insertButton = new JButton("Insert");

panel\_1.add(insertButton);

updateButton = new JButton("Update");

panel\_1.add(updateButton);

deleteButton = new JButton("Delete");

panel\_1.add(deleteButton);

scrollPane = new JScrollPane();

GroupLayout gl\_panel = new GroupLayout(panel);

gl\_panel.setHorizontalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addComponent(scrollPane, GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE)

);

gl\_panel.setVerticalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addComponent(scrollPane, GroupLayout.DEFAULT\_SIZE, 151, Short.MAX\_VALUE)

);

panel.setLayout(gl\_panel);

mainFrame.getContentPane().setLayout(groupLayout);

}

//Getters.

public JFrame getFrmProducts() {

return mainFrame;

}

public JTable getProductViewTable() {

return productViewTable;

}

public JButton getInsertButton() {

return insertButton;

}

public JButton getUpdateButton() {

return updateButton;

}

public JButton getDeleteButton() {

return deleteButton;

}

//Setters.

public void setFrmProducts(JFrame frmProducts) {

this.mainFrame = frmProducts;

}

public void setProductViewTable(JTable productViewTable) {

this.productViewTable = productViewTable;

}

public void setInsertButton(JButton insertButton) {

this.insertButton = insertButton;

}

public void setUpdateButton(JButton updateButton) {

this.updateButton = updateButton;

}

public void setDeleteButton(JButton deleteButton) {

this.deleteButton = deleteButton;

}

/\*\*

\* Constructs the table that holds the product list.

\* <br>This method is needed by the controller class to construct a JTable object

\* with the correct number of rows from the database.

\* @param numberOfRows int

\*/

public void createTable(int numberOfRows){

String[] columnHead = {"ID", "Product"};

TableModel tModel = new DefaultTableModel(columnHead,numberOfRows);

productViewTable = new JTable(tModel);

productViewTable.getColumnModel().getColumn(0).setMinWidth(40);

productViewTable.getColumnModel().getColumn(0).setMaxWidth(40);

scrollPane.setViewportView(productViewTable);

productViewTable.getColumnModel().getColumn(0).setMinWidth(40);

productViewTable.getColumnModel().getColumn(0).setMaxWidth(40);

scrollPane.setViewportView(productViewTable);

}//end createTable

/\*\*

\* Make the frame visible.

\*/

public void showGUI(){

this.mainFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.mainFrame.pack();

this.mainFrame.setLocationRelativeTo(null);

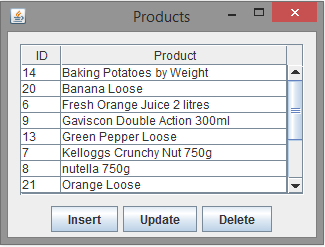
this.mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

this.mainFrame.setVisible(true);

}//end showGUI().

}

**Screenshot:**



**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class ProductController**

java.lang.Object

**pospi.ProductController**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

public class **ProductController**extends java.lang.Objectimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description: Handles the logic and events of the ProductView class.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ProductController**](about:blank../pospi/ProductController.html#ProductController(pospi.ProductView, java.lang.String, java.lang.String, java.lang.String))(pospi.ProductView view, java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/ProductController.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract method. Handles the events when a button is pressed. |
| boolean | [**deleteProduct**](about:blank../pospi/ProductController.html#deleteProduct(int))(int productID)           Deletes a product from the products table. |
| void | [**displayProducts**](about:blank../pospi/ProductController.html#displayProducts(java.sql.ResultSet))(java.sql.ResultSet rs)           Displays the records of the products table in a JTable |
| java.lang.String | [**getDataConnection**](about:blank../pospi/ProductController.html#getDataConnection())() |
| java.lang.String | [**getDataPassword**](about:blank../pospi/ProductController.html#getDataPassword())() |
| java.lang.String | [**getDataUser**](about:blank../pospi/ProductController.html#getDataUser())() |
| java.sql.ResultSet | [**getProductSet**](about:blank../pospi/ProductController.html#getProductSet(java.lang.String))(java.lang.String sql)           Retrieves records for the products table. |
| pospi.ProductView | [**getpView**](about:blank../pospi/ProductController.html#getpView())() |
| int | [**getTotalRows**](about:blank../pospi/ProductController.html#getTotalRows())()           Gets the total number of records in the product table to set the number of rows of the JTable. |
| void | [**initialize**](about:blank../pospi/ProductController.html#initialize())()           Initialize component's event-listeners. |
| void | [**setDataConnection**](about:blank../pospi/ProductController.html#setDataConnection(java.lang.String))(java.lang.String dataConnection) |
| void | [**setDataPassword**](about:blank../pospi/ProductController.html#setDataPassword(java.lang.String))(java.lang.String dataPassword) |
| void | [**setDataUser**](about:blank../pospi/ProductController.html#setDataUser(java.lang.String))(java.lang.String dataUser) |
| void | [**setpView**](about:blank../pospi/ProductController.html#setpView(pospi.ProductView))(pospi.ProductView pView) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### ProductController

public **ProductController**(pospi.ProductView view,

java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor. Calls the displayProducts() method.

**Parameters:**

view - ProductView

connection - String

user - String

password - String

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract method.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### deleteProduct

public boolean **deleteProduct**(int productID)

Deletes a product from the products table.

**Parameters:**

productID - int

**Returns:**

boolean true if delete was successful, false otherwise.

### displayProducts

public void **displayProducts**(java.sql.ResultSet rs)

Displays the records of the products table in a JTable

**Parameters:**

rs - ResultSet

### getDataConnection

public java.lang.String **getDataConnection**()

### getDataPassword

public java.lang.String **getDataPassword**()

### getDataUser

public java.lang.String **getDataUser**()

### getProductSet

public java.sql.ResultSet **getProductSet**(java.lang.String sql)

Retrieves records for the products table.

**Parameters:**

sql - String

**Returns:**

ResultSet

### getpView

public pospi.ProductView **getpView**()

### getTotalRows

public int **getTotalRows**()

Gets the total number of records in the product table to set the number of rows of the JTable.

**Returns:**

int total number of records.

### initialize

public void **initialize**()

Initialize component's event-listeners.

### setDataConnection

public void **setDataConnection**(java.lang.String dataConnection)

### setDataPassword

public void **setDataPassword**(java.lang.String dataPassword)

### setDataUser

public void **setDataUser**(java.lang.String dataUser)

### setpView

public void **setpView**(pospi.ProductView pView)

**displayProducts() – Sequence Diagram**

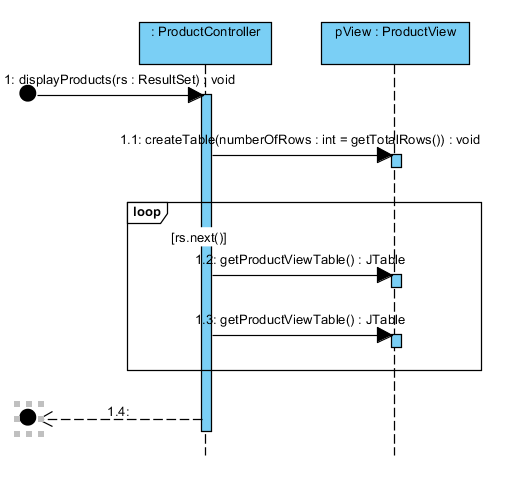
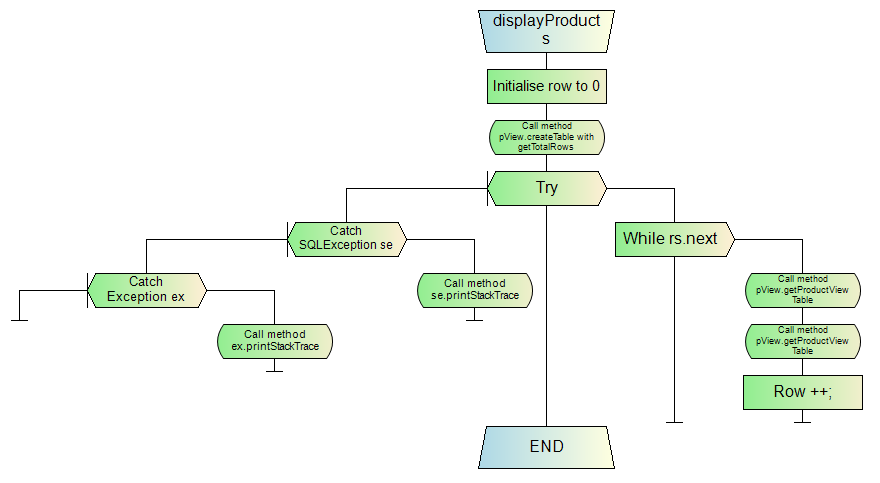


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**displayProducts() – Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**ProductController class code listing:**

package pospi;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.sql.\*;

import javax.swing.JOptionPane;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events of the ProductView class.

\* \*/

public class ProductController implements ActionListener{

private ProductView pView;

private String dataConnection;

private String dataUser;

private String dataPassword;

/\*\*

\* Constructor. Calls the displayProducts() method.

\* @param view ProductView

\* @param connection String

\* @param user String

\* @param password String

\*/

public ProductController(ProductView view, String connection, String user, String password){

this.pView = view;

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

initialize();

this.pView.showGUI();

displayProducts(getProductSet("SELECT product\_id, description FROM products " +

"ORDER BY description;"));

}//end constructor.

//Getters.

public ProductView getpView() {

return pView;

}

public String getDataConnection() {

return dataConnection;

}

public String getDataUser() {

return dataUser;

}

public String getDataPassword() {

return dataPassword;

}

//Setters.

public void setpView(ProductView pView) {

this.pView = pView;

}

public void setDataConnection(String dataConnection) {

this.dataConnection = dataConnection;

}

public void setDataUser(String dataUser) {

this.dataUser = dataUser;

}

public void setDataPassword(String dataPassword) {

this.dataPassword = dataPassword;

}

/\*\*

\* Initialize component's event-listeners.

\*/

public void initialize(){

this.pView.getInsertButton().addActionListener(this);

this.pView.getInsertButton().setActionCommand("insert");

this.pView.getUpdateButton().addActionListener(this);

this.pView.getUpdateButton().setActionCommand("update");

this.pView.getDeleteButton().addActionListener(this);

this.pView.getDeleteButton().setActionCommand("delete");

}//end initialize().

/\*\*

\* Implemented ActionListener abstract method.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

switch(command){

case "insert":

@SuppressWarnings("unused")

ProductMaintenanceController pmControlI = new ProductMaintenanceController(new ProductMaintenanceView(), this.dataConnection,

this.dataUser, this.dataPassword);

break;

case "update":

ProductMaintenanceView pmView = new ProductMaintenanceView();

@SuppressWarnings("unused")

ProductMaintenanceController pmControl = new ProductMaintenanceController(pmView, this.dataConnection, this.dataUser,

this.dataPassword,

//Gets the product\_id from the currently selected row on the JTable.

Integer.valueOf(this.pView.getProductViewTable().getModel().getValueAt(this.pView.getProductViewTable().getSelectedRow(), 0).toString()));

pmView.showGUI();

break;

case "delete":

if(deleteProduct(Integer.valueOf(this.pView.getProductViewTable().getModel().getValueAt(this.pView.getProductViewTable().getSelectedRow(), 0).toString())) == true){

JOptionPane.showMessageDialog(null, "Product was successfully\ndeleted.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

displayProducts(getProductSet("SELECT product\_id, description FROM products " +

"ORDER BY description;"));

}

else{

JOptionPane.showMessageDialog(null, "An error was encountered\nin deleting the product.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

break;

}

}//end actionPerformed()

/\*\*

\* Retrieves records for the products table.

\* @param sql String

\* @return ResultSet

\*/

public ResultSet getProductSet(String sql){

Connection connection;

Statement statement;

try{

connection = DriverManager.getConnection(this.dataConnection, this.dataUser, this.dataPassword);

statement = connection.createStatement();

return statement.executeQuery(sql);

}//end try

catch(SQLException se){

se.printStackTrace();

return null;

}//end catch.

catch(Exception ex){

ex.printStackTrace();

return null;

}//end catch.

finally{

//Do some clean up.

statement = null;

connection = null;

}//end finally

}//end getProductSet().

/\*\*

\* Gets the total number of records in the product table to set the number of rows of the JTable.

\* @return int total number of records.

\*/

public int getTotalRows(){

ResultSet rs = getProductSet("SELECT COUNT(\*) AS total\_number FROM products;");

int totalRows = 0;

try{

if(rs.next()){

totalRows = rs.getInt("total\_number");

}

}

catch(SQLException se){

se.printStackTrace();

}//end catch.

catch(Exception ex){

ex.printStackTrace();

}//end catch.

return totalRows;

}//

/\*\*

\* Displays the records of the products table in a JTable

\* @param rs ResultSet

\*/

public void displayProducts(ResultSet rs){

int row = 0;

pView.createTable(getTotalRows());//Construct JTable with specified number of rows.

try{

while(rs.next()){

pView.getProductViewTable().setValueAt(rs.getString("product\_id"), row, 0);

pView.getProductViewTable().setValueAt(rs.getString("description"), row, 1);

row ++;

}//end while

}//end try

catch(SQLException se){

se.printStackTrace();

}//end catch

catch(Exception ex){

ex.printStackTrace();

}//end catch

}//end displayProduct();

/\*\*

\* Deletes a product from the products table.

\* @param productID int

\* @return boolean true if delete was successful, false otherwise.

\*/

public boolean deleteProduct(int productID){

Connection dbConnection = null;

Statement deleteStatement = null;

try{

//Indicate manual transaction.

dbConnection = DriverManager.getConnection(this.dataConnection, this.dataUser, this.dataPassword);

dbConnection.setAutoCommit(false);

deleteStatement = dbConnection.createStatement();

deleteStatement.execute("DELETE FROM product\_inventory WHERE product\_id = " + productID);

deleteStatement.execute("DELETE FROM products WHERE product\_id = " + productID);

//commit changes.

dbConnection.commit();

return true;

}//end try.

catch(SQLException se){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e) {

e.printStackTrace();

}

se.printStackTrace();

return false;

}//end catch.

catch(Exception e){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

return false;

}//end catch.

finally{

//Do some clean-up.

dbConnection = null;

deleteStatement = null;

}//end finally.

}//end deleteProduct()

}//end class.

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class ProductMaintenanceView**

java.lang.Object

**pospi.ProductMaintenanceView**

public class **ProductMaintenanceView**extends java.lang.Object

Author: Angelo Romel Lopez  
Description: Constructs and initializes the components and graphical user interface of the user product maintenancer view screen.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ProductMaintenanceView**](about:blank../pospi/ProductMaintenanceView.html#ProductMaintenanceView())()           Constructor. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| javax.swing.JTextField | [**getBarCodeTextField**](about:blank../pospi/ProductMaintenanceView.html#getBarCodeTextField())() |
| javax.swing.JButton | [**getCancelButton**](about:blank../pospi/ProductMaintenanceView.html#getCancelButton())() |
| javax.swing.JTextField | [**getDescriptionTextField**](about:blank../pospi/ProductMaintenanceView.html#getDescriptionTextField())() |
| javax.swing.JFrame | [**getMainFrame**](about:blank../pospi/ProductMaintenanceView.html#getMainFrame())() |
| javax.swing.JTextField | [**getPriceTextField**](about:blank../pospi/ProductMaintenanceView.html#getPriceTextField())() |
| javax.swing.JTextField | [**getQuantityTextField**](about:blank../pospi/ProductMaintenanceView.html#getQuantityTextField())() |
| javax.swing.JButton | [**getSaveButton**](about:blank../pospi/ProductMaintenanceView.html#getSaveButton())() |
| javax.swing.JComboBox | [**getUomCombo**](about:blank../pospi/ProductMaintenanceView.html#getUomCombo())() |
| javax.swing.JComboBox | [**getVatCombo**](about:blank../pospi/ProductMaintenanceView.html#getVatCombo())() |
| void | [**setBarCodeTextField**](about:blank../pospi/ProductMaintenanceView.html#setBarCodeTextField(javax.swing.JTextField))(javax.swing.JTextField barCodeTextField) |
| void | [**setCancelButton**](about:blank../pospi/ProductMaintenanceView.html#setCancelButton(javax.swing.JButton))(javax.swing.JButton cancelButton) |
| void | [**setDescriptionTextField**](about:blank../pospi/ProductMaintenanceView.html#setDescriptionTextField(javax.swing.JTextField))(javax.swing.JTextField descriptionTextField) |
| void | [**setMainFrame**](about:blank../pospi/ProductMaintenanceView.html#setMainFrame(javax.swing.JFrame))(javax.swing.JFrame mainFrame) |
| void | [**setPriceTextField**](about:blank../pospi/ProductMaintenanceView.html#setPriceTextField(javax.swing.JTextField))(javax.swing.JTextField priceTextField) |
| void | [**setQuantityTextField**](about:blank../pospi/ProductMaintenanceView.html#setQuantityTextField(javax.swing.JTextField))(javax.swing.JTextField quantityTextField) |
| void | [**setSaveButton**](about:blank../pospi/ProductMaintenanceView.html#setSaveButton(javax.swing.JButton))(javax.swing.JButton saveButton) |
| void | [**setUomCombo**](about:blank../pospi/ProductMaintenanceView.html#setUomCombo(javax.swing.JComboBox))(javax.swing.JComboBox uomCombo) |
| void | [**setVatCombo**](about:blank../pospi/ProductMaintenanceView.html#setVatCombo(javax.swing.JComboBox))(javax.swing.JComboBox vatCombo) |
| void | [**showGUI**](about:blank../pospi/ProductMaintenanceView.html#showGUI())()           Make the frame visible. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### ProductMaintenanceView

public **ProductMaintenanceView**()

Constructor.

|  |
| --- |
| **Method Detail** |

### getBarCodeTextField

public javax.swing.JTextField **getBarCodeTextField**()

### getCancelButton

public javax.swing.JButton **getCancelButton**()

### getDescriptionTextField

public javax.swing.JTextField **getDescriptionTextField**()

### getMainFrame

public javax.swing.JFrame **getMainFrame**()

### getPriceTextField

public javax.swing.JTextField **getPriceTextField**()

### getQuantityTextField

public javax.swing.JTextField **getQuantityTextField**()

### getSaveButton

public javax.swing.JButton **getSaveButton**()

### getUomCombo

public javax.swing.JComboBox **getUomCombo**()

### getVatCombo

public javax.swing.JComboBox **getVatCombo**()

### setBarCodeTextField

public void **setBarCodeTextField**(javax.swing.JTextField barCodeTextField)

### setCancelButton

public void **setCancelButton**(javax.swing.JButton cancelButton)

### setDescriptionTextField

public void **setDescriptionTextField**(javax.swing.JTextField descriptionTextField)

### setMainFrame

public void **setMainFrame**(javax.swing.JFrame mainFrame)

### setPriceTextField

public void **setPriceTextField**(javax.swing.JTextField priceTextField)

### setQuantityTextField

public void **setQuantityTextField**(javax.swing.JTextField quantityTextField)

### setSaveButton

public void **setSaveButton**(javax.swing.JButton saveButton)

### setUomCombo

public void **setUomCombo**(javax.swing.JComboBox uomCombo)

### setVatCombo

public void **setVatCombo**(javax.swing.JComboBox vatCombo)

### showGUI

public void **showGUI**()

Make the frame visible.

**ProductMaintenanceView class code listing:**

package pospi;

import javax.swing.JFrame;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JPanel;

import java.awt.GridBagLayout;

import javax.swing.JLabel;

import java.awt.GridBagConstraints;

import javax.swing.JTextField;

import java.awt.Insets;

import javax.swing.JComboBox;

import javax.swing.LayoutStyle.ComponentPlacement;

import java.awt.FlowLayout;

import javax.swing.JButton;

import java.awt.Font;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Constructs and initializes the components and graphical user interface

\* of the user product maintenancer view screen.

\* \*/

public class ProductMaintenanceView {

private JFrame mainFrame;

private JTextField barCodeTextField;

private JTextField descriptionTextField;

private JTextField priceTextField;

private JTextField quantityTextField;

private JButton saveButton;

private JButton cancelButton;

@SuppressWarnings("rawtypes")

private JComboBox uomCombo;

@SuppressWarnings("rawtypes")

private JComboBox vatCombo;

/\*\*

\* Constructor.

\*/

public ProductMaintenanceView() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*\*/

@SuppressWarnings("rawtypes")

private void initialize() {

mainFrame = new JFrame();

mainFrame.setTitle("Product Info");

mainFrame.setBounds(100, 100, 320, 233);

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

JPanel panel\_1 = new JPanel();

GroupLayout groupLayout = new GroupLayout(mainFrame.getContentPane());

groupLayout.setHorizontalGroup(

groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, groupLayout.createSequentialGroup()

.addContainerGap()

.addGroup(groupLayout.createParallelGroup(Alignment.TRAILING)

.addComponent(panel, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE)

.addComponent(panel\_1, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE))

.addContainerGap())

);

groupLayout.setVerticalGroup(

groupLayout.createParallelGroup(Alignment.TRAILING)

.addGroup(groupLayout.createSequentialGroup()

.addContainerGap()

.addComponent(panel, GroupLayout.DEFAULT\_SIZE, 140, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(panel\_1, GroupLayout.PREFERRED\_SIZE, 26, GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

panel\_1.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

saveButton = new JButton("Save");

saveButton.setFont(new Font("Tahoma", Font.PLAIN, 9));

panel\_1.add(saveButton);

cancelButton = new JButton("Cancel");

cancelButton.setFont(new Font("Tahoma", Font.PLAIN, 9));

panel\_1.add(cancelButton);

GridBagLayout gbl\_panel = new GridBagLayout();

gbl\_panel.columnWidths = new int[]{83, 189, 0};

gbl\_panel.rowHeights = new int[]{19, 19, 19, 19, 19, 19, 0};

gbl\_panel.columnWeights = new double[]{0.0, 0.0, Double.MIN\_VALUE};

gbl\_panel.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, Double.MIN\_VALUE};

panel.setLayout(gbl\_panel);

JLabel lblNewLabel = new JLabel("Barcode:");

lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblNewLabel = new GridBagConstraints();

gbc\_lblNewLabel.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel.gridx = 0;

gbc\_lblNewLabel.gridy = 0;

panel.add(lblNewLabel, gbc\_lblNewLabel);

barCodeTextField = new JTextField();

barCodeTextField.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_barCodeTextField = new GridBagConstraints();

gbc\_barCodeTextField.anchor = GridBagConstraints.NORTH;

gbc\_barCodeTextField.fill = GridBagConstraints.HORIZONTAL;

gbc\_barCodeTextField.insets = new Insets(0, 0, 5, 0);

gbc\_barCodeTextField.gridx = 1;

gbc\_barCodeTextField.gridy = 0;

panel.add(barCodeTextField, gbc\_barCodeTextField);

barCodeTextField.setColumns(10);

JLabel lblDescription = new JLabel("Description:");

lblDescription.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblDescription = new GridBagConstraints();

gbc\_lblDescription.anchor = GridBagConstraints.EAST;

gbc\_lblDescription.insets = new Insets(0, 0, 5, 5);

gbc\_lblDescription.gridx = 0;

gbc\_lblDescription.gridy = 1;

panel.add(lblDescription, gbc\_lblDescription);

descriptionTextField = new JTextField();

descriptionTextField.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_descriptionTextField = new GridBagConstraints();

gbc\_descriptionTextField.anchor = GridBagConstraints.NORTH;

gbc\_descriptionTextField.fill = GridBagConstraints.HORIZONTAL;

gbc\_descriptionTextField.insets = new Insets(0, 0, 5, 0);

gbc\_descriptionTextField.gridx = 1;

gbc\_descriptionTextField.gridy = 1;

panel.add(descriptionTextField, gbc\_descriptionTextField);

descriptionTextField.setColumns(10);

JLabel lblNewLabel\_1 = new JLabel("Price:");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblNewLabel\_1 = new GridBagConstraints();

gbc\_lblNewLabel\_1.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_1.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_1.gridx = 0;

gbc\_lblNewLabel\_1.gridy = 2;

panel.add(lblNewLabel\_1, gbc\_lblNewLabel\_1);

priceTextField = new JTextField();

priceTextField.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_priceTextField = new GridBagConstraints();

gbc\_priceTextField.anchor = GridBagConstraints.NORTH;

gbc\_priceTextField.fill = GridBagConstraints.HORIZONTAL;

gbc\_priceTextField.insets = new Insets(0, 0, 5, 0);

gbc\_priceTextField.gridx = 1;

gbc\_priceTextField.gridy = 2;

panel.add(priceTextField, gbc\_priceTextField);

priceTextField.setColumns(10);

JLabel lblUom = new JLabel("UOM:");

lblUom.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblUom = new GridBagConstraints();

gbc\_lblUom.anchor = GridBagConstraints.EAST;

gbc\_lblUom.insets = new Insets(0, 0, 5, 5);

gbc\_lblUom.gridx = 0;

gbc\_lblUom.gridy = 3;

panel.add(lblUom, gbc\_lblUom);

uomCombo = new JComboBox();

uomCombo.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_uomCombo = new GridBagConstraints();

gbc\_uomCombo.anchor = GridBagConstraints.NORTH;

gbc\_uomCombo.fill = GridBagConstraints.HORIZONTAL;

gbc\_uomCombo.insets = new Insets(0, 0, 5, 0);

gbc\_uomCombo.gridx = 1;

gbc\_uomCombo.gridy = 3;

panel.add(uomCombo, gbc\_uomCombo);

JLabel lblVat = new JLabel("VAT:");

lblVat.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblVat = new GridBagConstraints();

gbc\_lblVat.anchor = GridBagConstraints.EAST;

gbc\_lblVat.insets = new Insets(0, 0, 5, 5);

gbc\_lblVat.gridx = 0;

gbc\_lblVat.gridy = 4;

panel.add(lblVat, gbc\_lblVat);

vatCombo = new JComboBox();

vatCombo.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_vatCombo = new GridBagConstraints();

gbc\_vatCombo.anchor = GridBagConstraints.NORTH;

gbc\_vatCombo.fill = GridBagConstraints.HORIZONTAL;

gbc\_vatCombo.insets = new Insets(0, 0, 5, 0);

gbc\_vatCombo.gridx = 1;

gbc\_vatCombo.gridy = 4;

panel.add(vatCombo, gbc\_vatCombo);

JLabel lblNewLabel\_2 = new JLabel("Quantity in Stock:");

lblNewLabel\_2.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_lblNewLabel\_2 = new GridBagConstraints();

gbc\_lblNewLabel\_2.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_2.insets = new Insets(0, 0, 0, 5);

gbc\_lblNewLabel\_2.gridx = 0;

gbc\_lblNewLabel\_2.gridy = 5;

panel.add(lblNewLabel\_2, gbc\_lblNewLabel\_2);

quantityTextField = new JTextField();

quantityTextField.setFont(new Font("Tahoma", Font.PLAIN, 10));

GridBagConstraints gbc\_quantityTextField = new GridBagConstraints();

gbc\_quantityTextField.anchor = GridBagConstraints.NORTH;

gbc\_quantityTextField.fill = GridBagConstraints.HORIZONTAL;

gbc\_quantityTextField.gridx = 1;

gbc\_quantityTextField.gridy = 5;

panel.add(quantityTextField, gbc\_quantityTextField);

quantityTextField.setColumns(10);

mainFrame.getContentPane().setLayout(groupLayout);

}

//Getters.

public JFrame getMainFrame() {

return mainFrame;

}

public JTextField getBarCodeTextField() {

return barCodeTextField;

}

public JTextField getDescriptionTextField() {

return descriptionTextField;

}

public JTextField getPriceTextField() {

return priceTextField;

}

public JTextField getQuantityTextField() {

return quantityTextField;

}

public JButton getSaveButton() {

return saveButton;

}

public JButton getCancelButton() {

return cancelButton;

}

@SuppressWarnings("rawtypes")

public JComboBox getUomCombo() {

return uomCombo;

}

@SuppressWarnings("rawtypes")

public JComboBox getVatCombo() {

return vatCombo;

}

//Setters.

public void setMainFrame(JFrame mainFrame) {

this.mainFrame = mainFrame;

}

public void setBarCodeTextField(JTextField barCodeTextField) {

this.barCodeTextField = barCodeTextField;

}

public void setDescriptionTextField(JTextField descriptionTextField) {

this.descriptionTextField = descriptionTextField;

}

public void setPriceTextField(JTextField priceTextField) {

this.priceTextField = priceTextField;

}

public void setQuantityTextField(JTextField quantityTextField) {

this.quantityTextField = quantityTextField;

}

public void setSaveButton(JButton saveButton) {

this.saveButton = saveButton;

}

public void setCancelButton(JButton cancelButton) {

this.cancelButton = cancelButton;

}

@SuppressWarnings("rawtypes")

public void setUomCombo(JComboBox uomCombo) {

this.uomCombo = uomCombo;

}

@SuppressWarnings("rawtypes")

public void setVatCombo(JComboBox vatCombo) {

this.vatCombo = vatCombo;

}

/\*\*

\* Make the frame visible.

\*\*/

public void showGUI(){

this.mainFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.mainFrame.pack();

this.mainFrame.setLocationRelativeTo(null);

this.mainFrame.setExtendedState(java.awt.Frame.MAXIMIZED\_BOTH);

this.mainFrame.setVisible(true);

}//end showGUI().

}

**Javadoc Generated API Documentation**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

## **pospi Class ProductMaintenanceController**

java.lang.Object

**pospi.ProductMaintenanceController**

**All Implemented Interfaces:**

java.awt.event.ActionListener, java.util.EventListener

public class **ProductMaintenanceController**extends java.lang.Objectimplements java.awt.event.ActionListener

Author: Angelo Romel Lopez  
Description: Handles the logic and events of the ProductMaintenanceView class.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**ProductMaintenanceController**](about:blank../pospi/ProductMaintenanceController.html#ProductMaintenanceController(pospi.ProductMaintenanceView, java.lang.String, java.lang.String, java.lang.String))(pospi.ProductMaintenanceView view, java.lang.String connection, java.lang.String user, java.lang.String password)           Constructor for the ProductMaintenanceController class. |  |
| [**ProductMaintenanceController**](about:blank../pospi/ProductMaintenanceController.html#ProductMaintenanceController(pospi.ProductMaintenanceView, java.lang.String, java.lang.String, java.lang.String, int))(pospi.ProductMaintenanceView view, java.lang.String connection, java.lang.String user, java.lang.String password, int productID)           Constructor for the ProductMaintenanceController class. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](about:blank../pospi/ProductMaintenanceController.html#actionPerformed(java.awt.event.ActionEvent))(java.awt.event.ActionEvent e)           Implemented ActionListener abstract method. Handles the events when a button is pressed. |
| void | [**displayProducts**](about:blank../pospi/ProductMaintenanceController.html#displayProducts(java.sql.ResultSet))(java.sql.ResultSet rs)           Displays a list of products on a JTable |
| void | [**fetchRecord**](about:blank../pospi/ProductMaintenanceController.html#fetchRecord())()           Gets the record to be updated and displays it on the frame. |
| java.lang.String | [**getDataConnection**](about:blank../pospi/ProductMaintenanceController.html#getDataConnection())() |
| java.lang.String | [**getDataPassword**](about:blank../pospi/ProductMaintenanceController.html#getDataPassword())() |
| java.lang.String | [**getDataUser**](about:blank../pospi/ProductMaintenanceController.html#getDataUser())() |
| boolean | [**getIsUpdate**](about:blank../pospi/ProductMaintenanceController.html#getIsUpdate())() |
| pospi.ProductMaintenanceView | [**getPmView**](about:blank../pospi/ProductMaintenanceController.html#getPmView())() |
| java.sql.ResultSet | [**getProductSet**](about:blank../pospi/ProductMaintenanceController.html#getProductSet(java.lang.String))(java.lang.String sql)           Retrieves records from the products table. |
| int | [**getUomID**](about:blank../pospi/ProductMaintenanceController.html#getUomID(java.lang.String))(java.lang.String uomCode)           Retrieves the uom\_id from the unit\_of\_measurement given the uom\_code. |
| void | [**initialize**](about:blank../pospi/ProductMaintenanceController.html#initialize())()           Initialize component's event-listeners. |
| boolean | [**insertProduct**](about:blank../pospi/ProductMaintenanceController.html#insertProduct(int))(int reorderLevel)           Inserts a new record to the products and product inventory table. |
| boolean | [**isValidNumber**](about:blank../pospi/ProductMaintenanceController.html#isValidNumber(java.lang.String))(java.lang.String number)           Returns true for a valid number, false otherwise. |
| void | [**populateUomComboBox**](about:blank../pospi/ProductMaintenanceController.html#populateUomComboBox())()           Populates the unit-of-measurement combobox with values from the table. |
| void | [**populateVatComboBox**](about:blank../pospi/ProductMaintenanceController.html#populateVatComboBox())()           Populates the VAT combobox with values from the table. |
| void | [**setDataConnection**](about:blank../pospi/ProductMaintenanceController.html#setDataConnection(java.lang.String))(java.lang.String dataConnection) |
| void | [**setDataPassword**](about:blank../pospi/ProductMaintenanceController.html#setDataPassword(java.lang.String))(java.lang.String dataPassword) |
| void | [**setDataUser**](about:blank../pospi/ProductMaintenanceController.html#setDataUser(java.lang.String))(java.lang.String dataUser) |
| void | [**setIsUpdate**](about:blank../pospi/ProductMaintenanceController.html#setIsUpdate(boolean))(boolean isUpdate) |
| void | [**setPmView**](about:blank../pospi/ProductMaintenanceController.html#setPmView(pospi.ProductMaintenanceView))(pospi.ProductMaintenanceView pmView) |
| boolean | [**updateProduct**](about:blank../pospi/ProductMaintenanceController.html#updateProduct())()           Updates the product record. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### ProductMaintenanceController

public **ProductMaintenanceController**(pospi.ProductMaintenanceView view,

java.lang.String connection,

java.lang.String user,

java.lang.String password)

Constructor for the ProductMaintenanceController class. Called if product is to be inserted, isUpdate is set to false.

**Parameters:**

view - (instance of ProductMaintenanceView class)

connection - (database connection string.)

user - (user name to connect to the database.)

password - (password for the supplied user name.)

### ProductMaintenanceController

public **ProductMaintenanceController**(pospi.ProductMaintenanceView view,

java.lang.String connection,

java.lang.String user,

java.lang.String password,

int productID)

Constructor for the ProductMaintenanceController class. Called if product is to be updated, isUpdate is set to true.

**Parameters:**

view - (instance of ProductMaintenanceView class)

connection - (database connection string.)

user - (user name to connect to the database.)

password - (password for the supplied user name.)

productID - (product ID of the product to be updated)

|  |
| --- |
| **Method Detail** |

### actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implemented ActionListener abstract method.  
Handles the events when a button is pressed.

**Specified by:**

actionPerformed in interface java.awt.event.ActionListener

**Parameters:**

e - ActionEvent

### displayProducts

public void **displayProducts**(java.sql.ResultSet rs)

Displays a list of products on a JTable

**Parameters:**

rs - ResultSet

### fetchRecord

public void **fetchRecord**()

Gets the record to be updated and displays it on the frame.

### getDataConnection

public java.lang.String **getDataConnection**()

### getDataPassword

public java.lang.String **getDataPassword**()

### getDataUser

public java.lang.String **getDataUser**()

### getIsUpdate

public boolean **getIsUpdate**()

### getPmView

public pospi.ProductMaintenanceView **getPmView**()

### getProductSet

public java.sql.ResultSet **getProductSet**(java.lang.String sql)

Retrieves records from the products table.

**Parameters:**

sql - String

**Returns:**

ResultSet

### getUomID

public int **getUomID**(java.lang.String uomCode)

Retrieves the uom\_id from the unit\_of\_measurement given the uom\_code.

**Parameters:**

uomCode - String

**Returns:**

int uom\_id

### initialize

public void **initialize**()

Initialize component's event-listeners.

### insertProduct

public boolean **insertProduct**(int reorderLevel)

Inserts a new record to the products and product inventory table.

**Returns:**

boolean true if the insert operation was successful, false otherwise.

### isValidNumber

public boolean **isValidNumber**(java.lang.String number)

Returns true for a valid number, false otherwise.

**Parameters:**

number - String

**Returns:**

boolean

### populateUomComboBox

public void **populateUomComboBox**()

Populates the unit-of-measurement combobox with values from the table.

### populateVatComboBox

public void **populateVatComboBox**()

Populates the VAT combobox with values from the table.

### setDataConnection

public void **setDataConnection**(java.lang.String dataConnection)

### setDataPassword

public void **setDataPassword**(java.lang.String dataPassword)

### setDataUser

public void **setDataUser**(java.lang.String dataUser)

### setIsUpdate

public void **setIsUpdate**(boolean isUpdate)

### setPmView

public void **setPmView**(pospi.ProductMaintenanceView pmView)

### updateProduct

public boolean **updateProduct**()

Updates the product record.

**Returns:**

boolean true if the update operation was successful, false otherwise.

**getProductSet() – Sequence diagram**

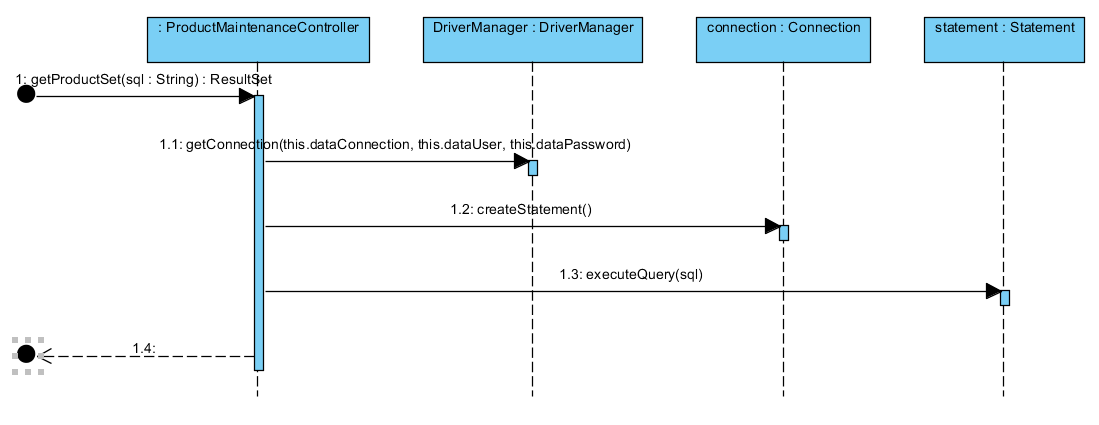
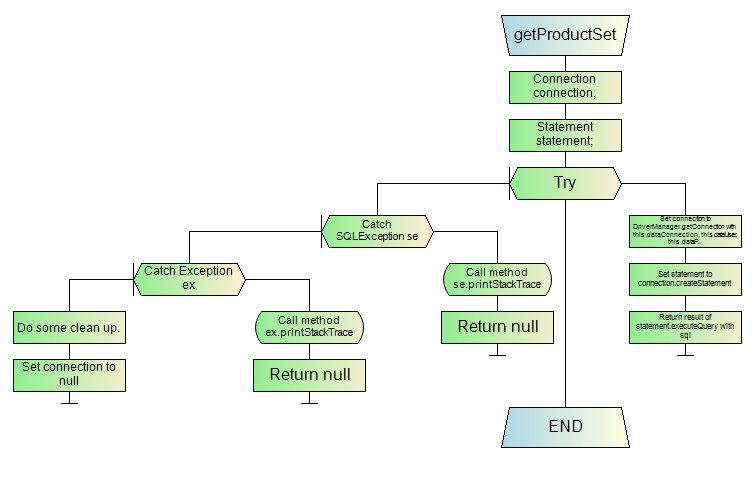


Diagram created with Visual Paradigm (<http://www.visual-paradigm.com/>)

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. [Accessed April 29, 2015].

**getProductSet() - Flowchart**



Auto-generated diagram created with Code Rocket Flowchart Editor (<http://www.rapidqualitysystems.com/>)

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. [Accessed April 29, 2015]

**ProductMaintenanceController class code listing:**

package pospi;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.JOptionPane;

/\*\*

\* Author: Angelo Romel Lopez

\* <br>Description: Handles the logic and events of the ProductMaintenanceView class.

\* \*/

public class ProductMaintenanceController implements ActionListener {

private ProductMaintenanceView pmView;

private String dataConnection;

private String dataUser;

private String dataPassword;

private boolean isUpdate;//if update is true then record is updated, otherwise record is inserted.

private int productID;

/\*\*

\* Constructor for the ProductMaintenanceController class. Called if product is to be updated, isUpdate is set to true.

\* @param view (instance of ProductMaintenanceView class)

\* @param connection (database connection string.)

\* @param user (user name to connect to the database.)

\* @param password (password for the supplied user name.)

\* @param productID (product ID of the product to be updated)

\*\*/

public ProductMaintenanceController(ProductMaintenanceView view, String connection, String user, String password, int productID){

this.pmView = view;

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

this.isUpdate = true;

this.productID = productID;

this.pmView.showGUI();

initialize();

//Populate the comboboxes.

populateUomComboBox();

populateVatComboBox();

this.pmView.getUomCombo().setSelectedItem("N/A");

//Display product information if product is to be updated.

if(this.isUpdate){

fetchRecord();

}

}//end constructor.

/\*\*

\* Constructor for the ProductMaintenanceController class. Called if product is to be inserted, isUpdate is set to false.

\* @param view (instance of ProductMaintenanceView class)

\* @param connection (database connection string.)

\* @param user (user name to connect to the database.)

\* @param password (password for the supplied user name.)

\*\*/

public ProductMaintenanceController(ProductMaintenanceView view, String connection, String user, String password){

this.pmView = view;

this.dataConnection = connection;

this.dataUser = user;

this.dataPassword = password;

this.isUpdate = false;

initialize();

this.pmView.showGUI();

//Populate the comboboxes.

populateUomComboBox();

populateVatComboBox();

this.pmView.getUomCombo().setSelectedItem("N/A");

}//end constructor.

//Getters.

public ProductMaintenanceView getPmView() {

return pmView;

}

public String getDataConnection() {

return dataConnection;

}

public String getDataUser() {

return dataUser;

}

public String getDataPassword() {

return dataPassword;

}

public boolean getIsUpdate(){

return this.isUpdate;

}

//Setters.

public void setPmView(ProductMaintenanceView pmView) {

this.pmView = pmView;

}

public void setDataConnection(String dataConnection) {

this.dataConnection = dataConnection;

}

public void setDataUser(String dataUser) {

this.dataUser = dataUser;

}

public void setDataPassword(String dataPassword) {

this.dataPassword = dataPassword;

}

public void setIsUpdate(boolean isUpdate){

this.isUpdate = isUpdate;

}

/\*\*

\* Initialize component's event-listeners.

\*\*/

public void initialize(){

this.pmView.getSaveButton().addActionListener(this);

this.pmView.getSaveButton().setActionCommand("save");

this.pmView.getCancelButton().addActionListener(this);

this.pmView.getCancelButton().setActionCommand("cancel");

}//end initialize().

/\*\*

\* Implemented ActionListener abstract method.<br>Handles the events when a button is pressed.

\* @param e ActionEvent

\*\*/

public void actionPerformed(ActionEvent e){

String command = e.getActionCommand();

switch(command){

case "save":

//Record is to be updated.

if(isUpdate == true){

if(updateProduct() == true){

JOptionPane.showMessageDialog(null, "Product information was\nupdated successfully.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

this.pmView.getMainFrame().dispose();

}

else{

JOptionPane.showMessageDialog(null, "Unable to update product\ninformation.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

}

//Record is to be inserted.

else{

//A product either have a barcode or a unit-of-measurement code, but not both.

if(!this.pmView.getBarCodeTextField().getText().equals("") && !this.pmView.getUomCombo().getSelectedItem().equals("N/A")){

JOptionPane.showMessageDialog(null, "Sorry, cannot have a\nvalue for both barcode\nand UOM",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

else{

String reorderLevel = JOptionPane.showInputDialog(null, "Please enter the reorder\nlevel for this product:",

"POS-Pi", JOptionPane.QUESTION\_MESSAGE);

if(isValidNumber(reorderLevel) == true){

if(insertProduct(Integer.valueOf(reorderLevel)) == true){

JOptionPane.showMessageDialog(null, "Product information was\nadded successfully.",

"POS-Pi.",

JOptionPane.WARNING\_MESSAGE);

this.pmView.getMainFrame().dispose();

}

else{

JOptionPane.showMessageDialog(null, "Unable to update product\ninformation.",

"POS-Pi",

JOptionPane.WARNING\_MESSAGE);

}

}

else{

JOptionPane.showMessageDialog(null, "Invalid number,\nplease try again.",

"Invalid number format.",

JOptionPane.WARNING\_MESSAGE);

}//end if

}

}//end if

break;

case "cancel":

this.pmView.getMainFrame().dispose();

break;

}

}//end actionPerformed().

/\*\*

\* Gets the record to be updated and displays it on the frame.

\*\*/

public void fetchRecord(){

String sql = "SELECT products.product\_id, products.barcode, products.uom\_id, unit\_of\_measurement.uom\_code, " +

"products.description, products.price, products.vat\_code, product\_inventory.quantity\_in\_stock " +

"FROM products LEFT JOIN unit\_of\_measurement " +

"ON products.uom\_id = unit\_of\_measurement.uom\_id LEFT JOIN " +

"product\_inventory ON products.product\_id = product\_inventory.product\_id " +

"WHERE products.product\_id = " + this.productID + ";";

displayProducts(getProductSet(sql));

}//end fetchRecord().

/\*\*

\* Retrieves records from the products table.

\* @param sql String

\* @return ResultSet

\*\*/

public ResultSet getProductSet(String sql){

Connection connection;

Statement statement;

try{

connection = DriverManager.getConnection(this.dataConnection, this.dataUser, this.dataPassword);

statement = connection.createStatement();

return statement.executeQuery(sql);

}//end try

catch(SQLException se){

se.printStackTrace();

return null;

}//end catch.

catch(Exception ex){

ex.printStackTrace();

return null;

}//end catch.

finally{

//Do some clean up.

statement = null;

connection = null;

}//end finally

}//end displayProduct().

/\*\*

\* Displays a list of products on a JTable

\* @param rs ResultSet

\*\*/

public void displayProducts(ResultSet rs){

try{

this.pmView.getUomCombo().setSelectedItem("(None)");

this.pmView.getVatCombo().setSelectedItem("(None)");

if(rs.next()){

this.pmView.getBarCodeTextField().setText(rs.getString("products.barcode"));

this.pmView.getDescriptionTextField().setText(rs.getString("products.description"));

this.pmView.getPriceTextField().setText(rs.getString("products.price"));

this.pmView.getUomCombo().setSelectedItem(rs.getString("unit\_of\_measurement.uom\_code"));

this.pmView.getVatCombo().setSelectedItem(rs.getString("products.vat\_code"));

this.pmView.getQuantityTextField().setText(rs.getString("product\_inventory.quantity\_in\_stock"));

}//end while

}//end try

catch(SQLException se){

se.printStackTrace();

}//end catch

catch(Exception ex){

ex.printStackTrace();

}//end catch

}//end displayProduct();

/\*\*

\* Populates the unit-of-measurement combobox with values from the table.

\*\*/

@SuppressWarnings("unchecked")

public void populateUomComboBox(){

ResultSet rs = getProductSet("SELECT uom\_code FROM unit\_of\_measurement;");

try {

while(rs.next()){

this.pmView.getUomCombo().addItem(rs.getString("uom\_code"));

}//end while.

} catch (SQLException e) {

e.printStackTrace();

}catch (Exception e) {

e.printStackTrace();

}//end catch.

}//end populateUomComboBox().

/\*\*

\* Populates the VAT combobox with values from the table.

\*\*/

@SuppressWarnings("unchecked")

public void populateVatComboBox(){

ResultSet rs = getProductSet("SELECT vat\_code FROM vat;");

try {

while(rs.next()){

this.pmView.getVatCombo().addItem(rs.getString("vat\_code"));

}//end while.

} catch (SQLException e) {

e.printStackTrace();

}catch (Exception e) {

e.printStackTrace();

}//end catch.

}//end populateVatComboBox().

/\*\*

\* Retrieves the uom\_id from the unit\_of\_measurement given the uom\_code.

\* @param uomCode String

\* @return int uom\_id

\*\*/

public int getUomID(String uomCode){

ResultSet rs = getProductSet("SELECT uom\_id FROM unit\_of\_measurement WHERE uom\_code = '" +

uomCode + "'");

try {

if(!rs.isBeforeFirst()){

//if ResultSet is not before the first result, then ResultSet is empty. Return -1.

return -1;

}

else{

rs.next();

return rs.getInt("uom\_id");

}

} catch (SQLException e) {

e.printStackTrace();

return -1;

}catch (Exception e) {

e.printStackTrace();

return -1;

}//end catch.

}//end getUomID()

/\*\*

\* Updates the product record.

\* @return boolean true if the update operation was successful, false otherwise.

\*\*/

public boolean updateProduct(){

Connection dbConnection;

PreparedStatement updateProduct = null,

updateStock = null;

try{

dbConnection = DriverManager.getConnection(this.dataConnection, this.dataUser, this.dataPassword);

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//Start transaction.

//Insert a value for the barcode if a value was specified, insert null otherwise.

if(!this.pmView.getBarCodeTextField().getText().equals("")){

updateProduct = dbConnection.prepareStatement("UPDATE products SET barcode = ?, uom\_id = ?" +

", description = ?, price = ?, vat\_code = ? " +

"WHERE product\_id = ?");

updateProduct.setString(1, this.pmView.getBarCodeTextField().getText());

updateProduct.setInt(2, getUomID(this.pmView.getUomCombo().getSelectedItem().toString()));

updateProduct.setString(3, this.pmView.getDescriptionTextField().getText());

updateProduct.setDouble(4, Double.valueOf(this.pmView.getPriceTextField().getText()));

updateProduct.setString(5, this.pmView.getVatCombo().getSelectedItem().toString());

updateProduct.setInt(6, this.productID);

}

else{

updateProduct = dbConnection.prepareStatement("UPDATE products SET uom\_id = ?" +

", description = ?, price = ?, vat\_code = ? " +

"WHERE product\_id = ?");

updateProduct.setInt(1, getUomID(this.pmView.getUomCombo().getSelectedItem().toString()));

updateProduct.setString(2, this.pmView.getDescriptionTextField().getText());

updateProduct.setDouble(3, Double.valueOf(this.pmView.getPriceTextField().getText()));

updateProduct.setString(4, this.pmView.getVatCombo().getSelectedItem().toString());

updateProduct.setInt(5, this.productID);

}

updateStock = dbConnection.prepareStatement("UPDATE product\_inventory SET quantity\_in\_stock = ? " +

"WHERE product\_id = ?");

updateStock.setDouble(1, Double.valueOf(this.pmView.getQuantityTextField().getText()));

updateStock.setInt(2, this.productID);

//Update product table.

updateProduct.executeUpdate();

//Update product\_inventory table.

updateStock.executeUpdate();

//Commit transaction

dbConnection.commit();

return true;

}//end try

catch(SQLException se){

se.printStackTrace();

return false;

}catch(Exception e){

e.printStackTrace();

return false;

}//end catch

finally{

//Do some clean-up.

dbConnection = null;

updateProduct = null;

updateStock = null;

}//end finally

}//end updateProduct().

/\*\*

\* Inserts a new record to the products and product inventory table.

\* @return boolean true if the insert operation was successful, false otherwise.

\*\*/

public boolean insertProduct(int reorderLevel){

Connection dbConnection = null;

Statement insertProduct = null;

PreparedStatement insertStock = null;

ResultSet rs = null;//For holding auto-generated keys.

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

int productPK;

try{

dbConnection = DriverManager.getConnection(this.dataConnection, this.dataUser, this.dataPassword);

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//Start transaction.

//Insert a value for the barcode if a value was specified, insert null otherwise.

if(!this.pmView.getBarCodeTextField().getText().equals("")){

insertProduct = dbConnection.createStatement();

insertProduct.executeUpdate("INSERT INTO products (barcode, uom\_id, description, price, vat\_code) VALUES ('" +

this.pmView.getBarCodeTextField().getText() + "', " +

getUomID(this.pmView.getUomCombo().getSelectedItem().toString()) + ", '" +

this.pmView.getDescriptionTextField().getText() + "', " +

Double.valueOf(this.pmView.getPriceTextField().getText()) + ", '" +

this.pmView.getVatCombo().getSelectedItem().toString() + "')", Statement.RETURN\_GENERATED\_KEYS);

}

else{//Inserts null for the barcode.

insertProduct = dbConnection.createStatement();

insertProduct.executeUpdate("INSERT INTO products (uom\_id, description, price, vat\_code) VALUES (" +

getUomID(this.pmView.getUomCombo().getSelectedItem().toString()) + ", '" +

this.pmView.getDescriptionTextField().getText() + "', " +

Double.valueOf(this.pmView.getPriceTextField().getText()) + ", '" +

this.pmView.getVatCombo().getSelectedItem().toString() + "')", Statement.RETURN\_GENERATED\_KEYS);

}

rs = insertProduct.getGeneratedKeys();//Retrieve the auto-number primary key.

if(rs.next()){

productPK = rs.getInt(1);

insertStock = dbConnection.prepareStatement("INSERT INTO product\_inventory (product\_id, inventory\_date, " +

"quantity\_in\_stock, reorder\_level) VALUES (?, ?, ?, ?)");

insertStock.setInt(1, productPK);

insertStock.setString(2, dateFormat.format(new Date()));

insertStock.setDouble(3, Double.valueOf(this.pmView.getQuantityTextField().getText()));

insertStock.setInt(4, reorderLevel);

insertStock.executeUpdate();

//Commit transaction.

dbConnection.commit();

}

else{//Error in generating primary key, throw exception to rollback changes.

throw new SQLException();

}

return true;

}//end try.

catch(SQLException se){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

se.printStackTrace();

return false;

}

catch(Exception e){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException ex) {

// TODO Auto-generated catch block

e.printStackTrace();

}

e.printStackTrace();

return false;

}//end catch.

finally{

//Do some clean-up.

dbConnection = null;

insertProduct = null;

insertStock = null;

}//end finally.

}//end insertProduct().

/\*\*

\* Returns true for a valid number, false otherwise.

\* @param number String

\* @return boolean

\*\*/

public boolean isValidNumber(String number){

try{

Double.parseDouble(number);

return true;

}

catch(NumberFormatException ex){

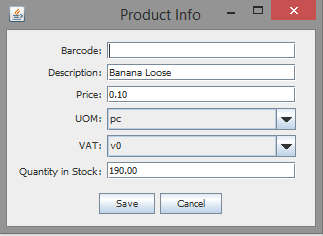
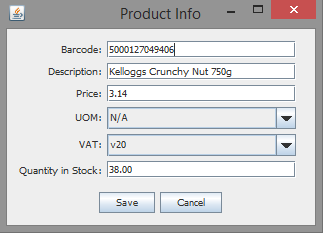
return false;

}//end try/catch

}//end isValidNumber

}//end class.

**Screenshots:**



**Database Design and Development Documentation**

The database design for the POS-Pi application is based on information gathered and researched on what a point of sale database design should look like.

The database is stored on a MySQL Community Server (GPL) version: 5.6.21 both running on Windows 8.1 and Raspbian (Debian Wheezy) Kernel version: 3.18. Both are accessed via phpMyAdmin web interface version: 4.2.11 and MySQL Workbench 6.3.

List of sources that I have used when designing the point of sale database:

-Industry Data Models: http://www.databaseanswers.org/data\_models/

-System Processes Document: http://eecs.vanderbilt.edu/courses/eece295/2005\_2006/u-club\_pos/documents.htm

-SQL by Design: http://sqlmag.com/database-performance-tuning/sql-design-why-you-need-database-normalization

-POS Database Model: http://wiki.openbravo.com/wiki/POS/2.30/Database\_Model

-Data Mode, Customer, Inventory and POS: http://www.databaseanswers.org/data\_models/customers\_inventory\_and\_pos/index.htm

-Database Normalisation Basics: http://databases.about.com/od/specificproducts/a/normalization.htm

-Database Normalisation Resources: http://www.hitechcoach.com/index.php?option=com\_content&view=article&id=34:database-normalization-resources&catid=53:designing

-Database Design and Normalisation: http://www.jpmartel.com/bu12ce03.htm

**Database Design**

The database design for the POS-Pi application is very similar to the design of other sales and order databases. Tables are needed to store product information from the stock and order details (invoice) for every transaction. I have decided not to include a table for the customer information as the application that it is designed for is meant to be used on a point of sale terminal/counter that has no immediate need to store information about customers.

Tables:

**Products Table**

This table needs to store the id, name or description of the product, as well as the selling price. It will also store a barcode if one is needed, or a unit-of-measurement code if a barcode is not applicable. A field that contains a reference to a VAT amount and a unit-of-measurement code is also needed.

**Product Inventory Table**

Contains a reference to a product stored in the Products table. It also needs to store how much products are left in stock, the re-order level and the date of the last inventory check.

**Unit of Measurement Table**

This table contains will act as a look-up table and contains a code and a description for the measurement. Examples of unit-of-measurement: by piece, by gram, by kilogram, etc. The unit-of-measurement code will used when a barcode is not available or possible to be used by a product.

**VAT Table**

Just like the Unit of Measurement table, this table will also act as a look-up table. It needs to contain a code, the rate (in whole number) of the VAT amount and the description.

**Orders Table**

Contains a unique order ID and the date the sale was made.

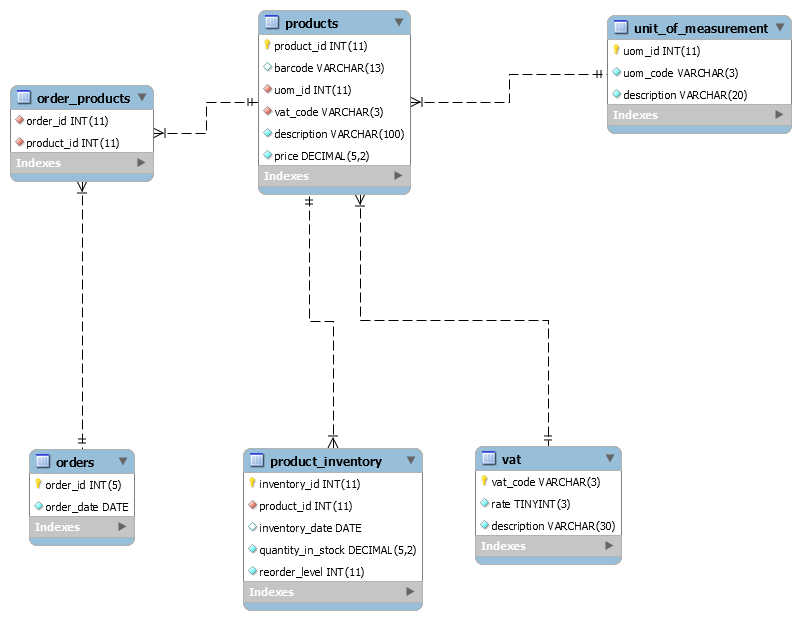
**Order Products or Order Details Table**

Contains a reference to the Orders table and the Products table and will contain all the products or items that was sold as part of the order or invoice.

**Users Table**

This table contains login information such as username and password for people that needs to access the database.

**POS-Pi (Point of Sale) Database ERD**

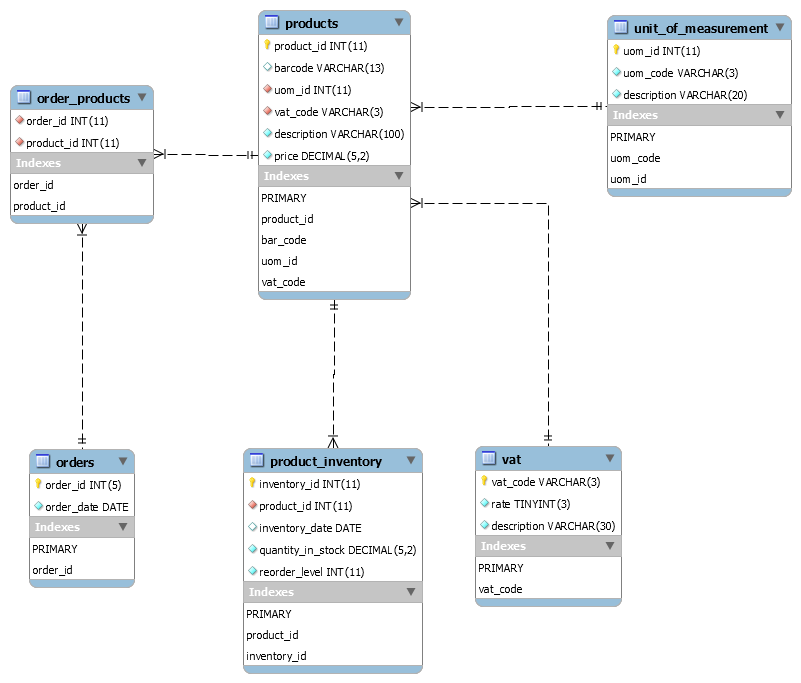


ERD diagram created with MySQL Workbench (<https://www.mysql.com/products/workbench/>)

MySQL, nd. MySQL Workbench (6.3). [computer program] MySQL. Available at < <https://www.mysql.com/products/workbench/>> [Accessed May 04, 2015].

**POS-Pi (Point of Sale) Database ERD**

This diagram shows the indexes associated with each table.



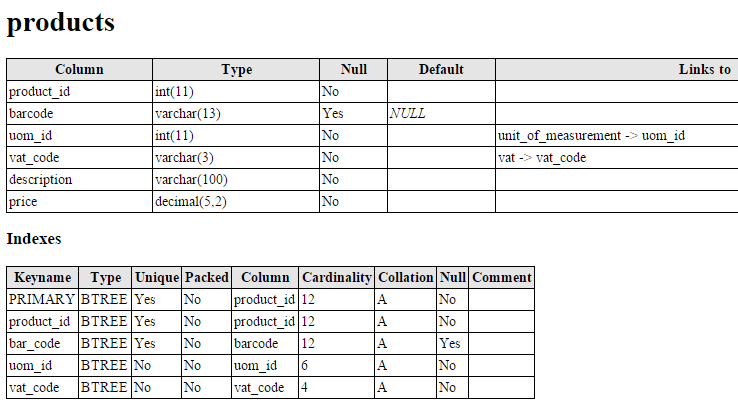
ERD diagram created with MySQL Workbench (<https://www.mysql.com/products/workbench/>)

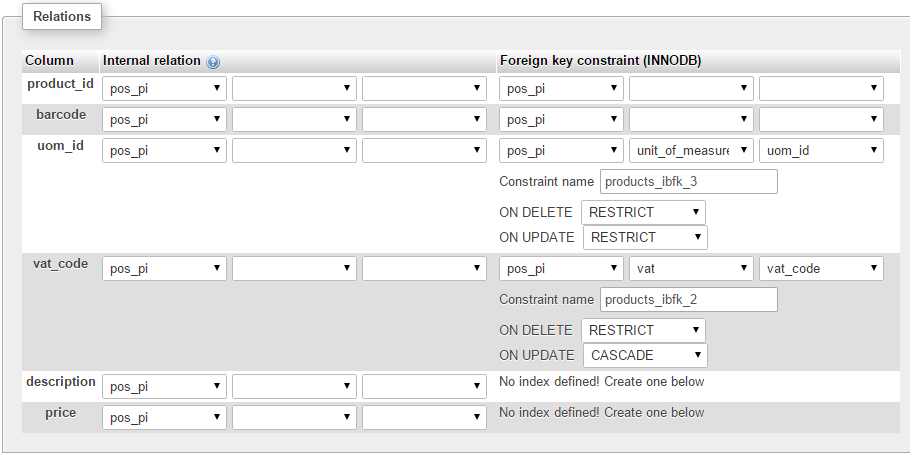
MySQL, nd. MySQL Workbench (6.3). [computer program] MySQL. Available at < <https://www.mysql.com/products/workbench/>> [Accessed May 04, 2015].

**Database: pos\_pi**

**Data Dictionary/Table Structure on MySQL Server (thru phpMyAdmin):**

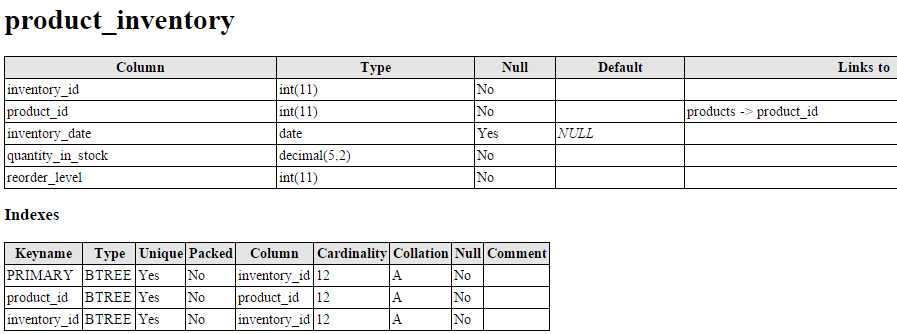
**Table: products**

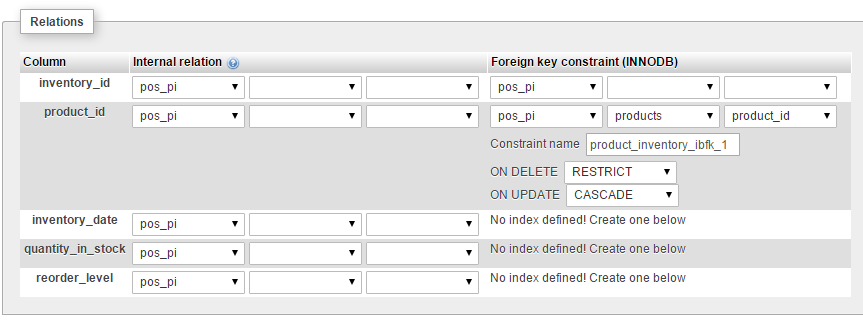




Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

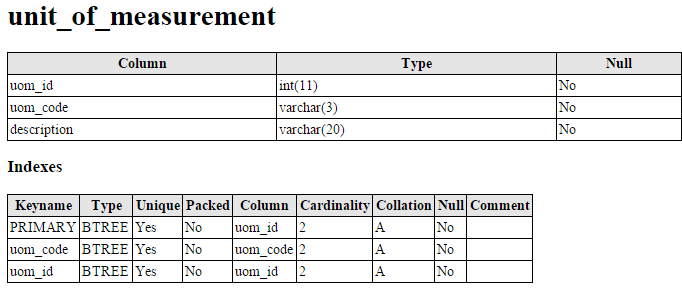
**Table: product\_inventory**



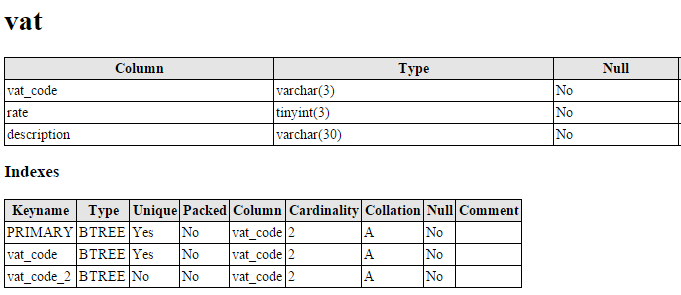


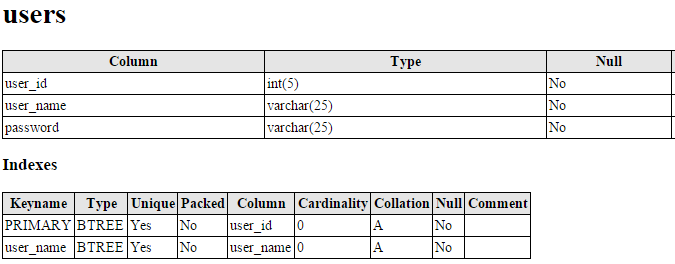
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

**Table: unit\_of\_measurement**



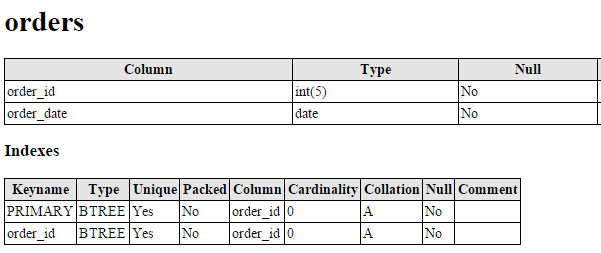
**Table: vat**



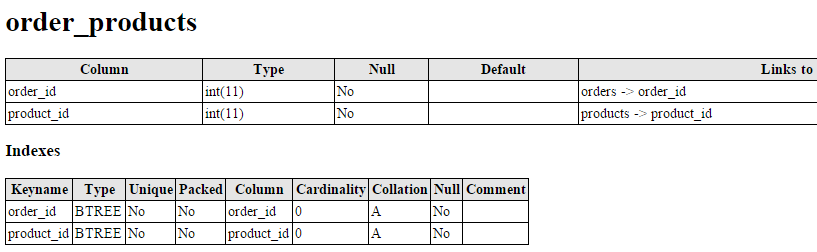


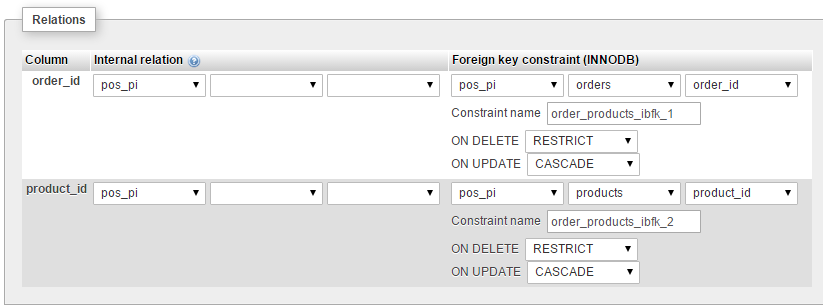
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

**Table: orders**



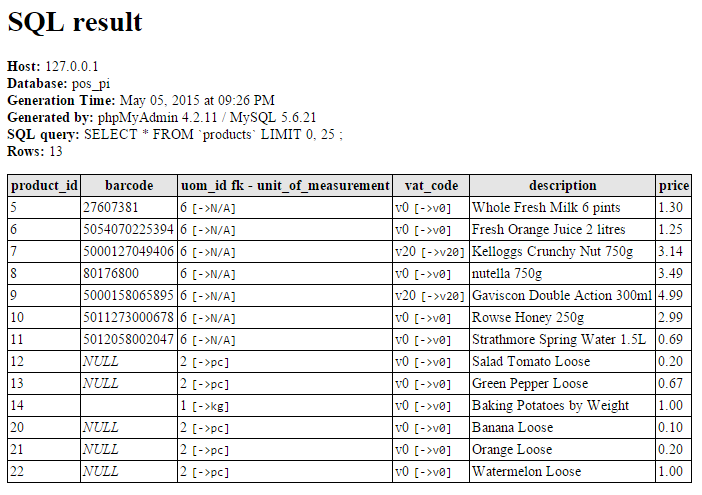
**Table: order\_products**





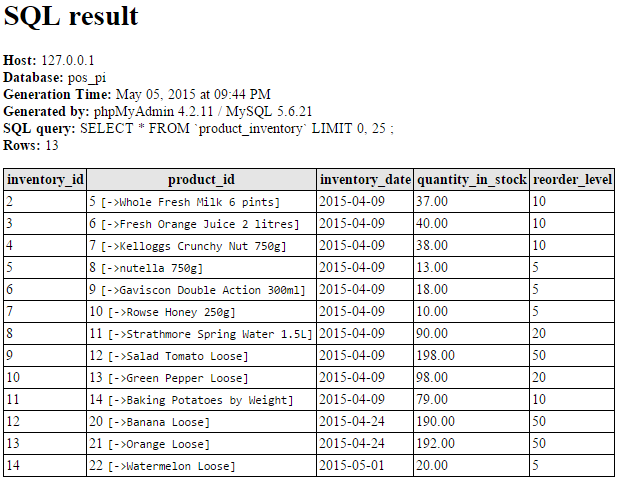
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

**Database tables populated with mock-data:**



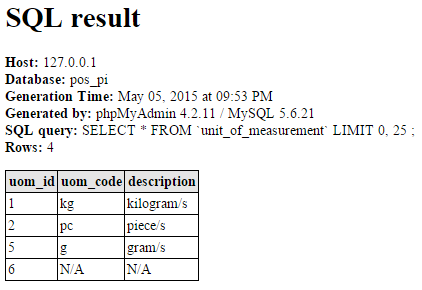
Note: The records for both uom\_id and vat\_code columns shows values from two columns from their respected tables. The value ‘6 [- >N/A]’ from the column uom\_id is a combination of the uom\_id = ‘6’ and uom\_code=’N/A’. They are shown this way to make the foreign key columns easier to read. In reality, only the uom\_id and the vat\_code is actually stored in the table.

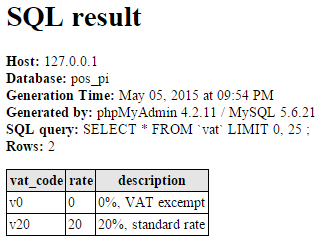
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

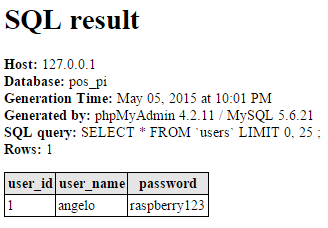


Note: Just like from the previous table, the product\_id displays the product\_id and description from the products table. Only the product\_id is actually stored in the product\_inventory table.

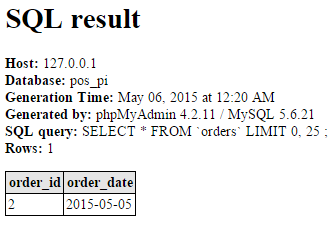
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

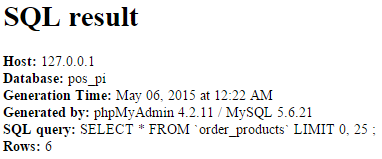


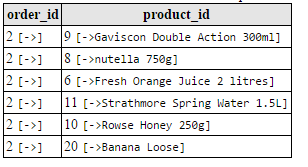




Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]





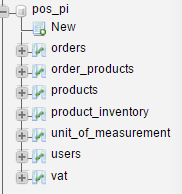
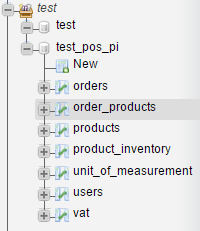


## 3.2. testing/implementation

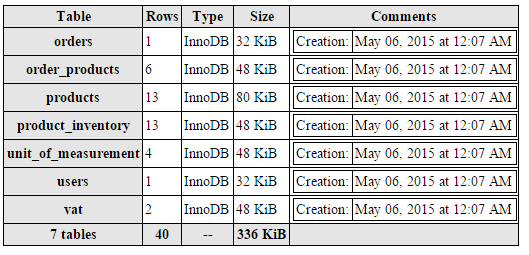
A backup copy of the database named ‘**test\_pos\_pi**’ was created for the purpose of testing the application. A test code will be created and executed to test for the following:

1. Database connectivity – database connection will be tested by verifying that an appropriate database driver is installed and registered, and that login credentials are correct.
2. Product insertion, update and deletion – verify that information is correctly inserted, updated and deleted.
3. Database transaction – verify that information is correctly commited after a successful transaction, or that changes to the database are being rolled-back after a failed transaction.

The original database (phpMyadmin): The cloned database (phpMyAdmin):

Data structure of the test database from phpMyadmin:



Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Test code for database registration and connection:**

**package** pospi;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**public** **class** TestDataConnection {

**private** **static** **boolean** isConnectionValid(){

**try**{

//Register JDBC driver (done once for the entire application).

Class.*forName*("com.mysql.jdbc.Driver");

//Test connection to database

@SuppressWarnings("unused")

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost/test\_pos\_pi",

"root", "raspberry");

**return** **true**;

}//end try

**catch**(Exception ex){

ex.printStackTrace();

**return** **false**;

}//end catch

}//isConnectionValid

**public** **static** **void** main(String[] args) {

**if**(*isConnectionValid*()){

System.***out***.println("Database registration and connection successful!!!");

}

**else**{

System.***out***.println();

}

}//end main

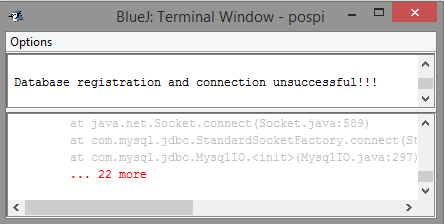
}//end class

**Database registration and connection - test run scenario using XAMPP Control Panel and BlueJ:**

-> With MySQL service stopped:



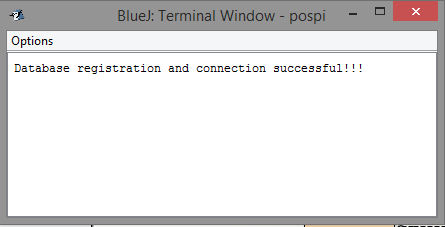
Result:



->With MySQL service started:



Result:



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

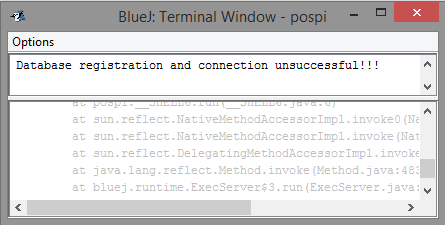
Apache Friends, nd. XAMPP Control Panel, [computer program]. Apache Friends. Available at: <https://www.apachefriends.org/download.html> [Accessed May 06, 2015]

->With incorrect database driver:

*Correct database driver: "com.mysql.jdbc.Driver"*

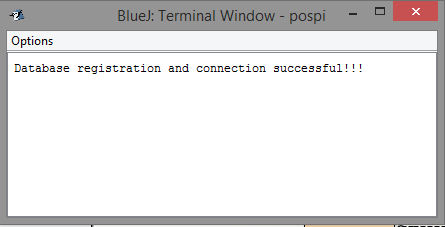
*Incorrect database driver: “com.mysql.odbc.Driver”*

Result:



->With correct database driver:

Result:



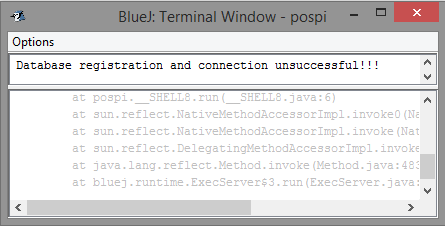
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

->With incorrect login credentials:

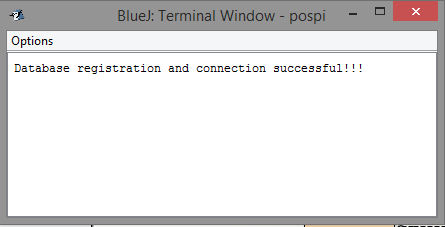
*Correct login credentials: username=”root”, password=”raspberry”*

*Incorrect login credentials: username=”gingerroot”, password=”raspberrypie”*

Result:

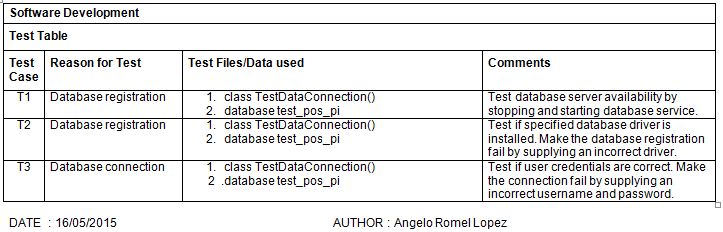


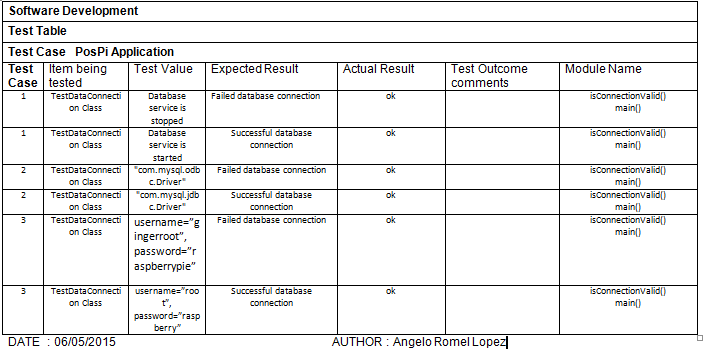
->With correct login credentials:



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Test Log:**





**Test code for product insertion using transaction:**

**The transaction includes inserting a new product record in the products table and product\_inventory.**

package pospi;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Date;

public class TestProductInsert {

//Insert product information with a value for the barcode.

private static boolean isProductWithBarcodeInserted(){

Connection dbConnection = null;

Statement insertProduct = null;

PreparedStatement insertStock = null;

ResultSet rs = null;//For holding auto-generated keys.

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

int productPK;

try{

dbConnection = DriverManager.getConnection("jdbc:mysql://localhost/test\_pos\_pi",

"root", "raspberry");

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//Start transaction.

insertProduct = dbConnection.createStatement();

insertProduct.executeUpdate("INSERT INTO products (barcode, uom\_id, description, price, vat\_code) VALUES ('27607381', "

+ "'6', 'Whole Fresh Milk 6 pints', 1.30, 'v0')", Statement.RETURN\_GENERATED\_KEYS);

rs = insertProduct.getGeneratedKeys();//Retrieve the auto-number primary key.

if(rs.next()){

productPK = rs.getInt(1);

insertStock = dbConnection.prepareStatement("INSERT INTO product\_inventory (product\_id, inventory\_date, " +

"quantity\_in\_stock, reorder\_level) VALUES (?, ?, ?, ?)");

insertStock.setInt(1, productPK);

insertStock.setString(2, dateFormat.format(new Date()));

insertStock.setDouble(3, 50);

insertStock.setInt(4, 10);

insertStock.executeUpdate();

//Commit transaction.

dbConnection.commit();

}

else{//Error in generating primary key, throw exception to rollback changes.

throw new SQLException();

}

return true;

}//end try.

catch(SQLException se){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

se.printStackTrace();

return false;

}

catch(Exception e){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException ex) {

// TODO Auto-generated catch block

e.printStackTrace();

}

e.printStackTrace();

return false;

}//end catch.

finally{

//Do some clean-up.

dbConnection = null;

insertProduct = null;

insertStock = null;

}//end finally.

}//end insertProduct().

public static void main(String[] args) {

System.out.println("New product information will be inserted into tables: products and product\_inventory.");

if(isProductWithBarcodeInserted()){

System.out.println("Product was successfully inserted!!!");

}

else{

System.out.println("Product insertion failed!!!");

}//end if

}//end main

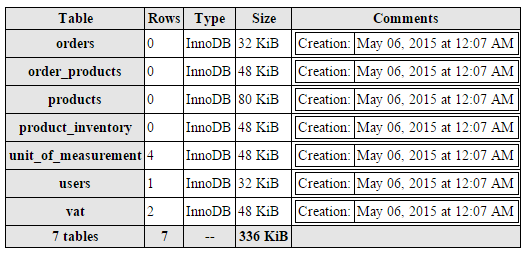
}//end class

**Existing rows for the products and the product\_inventory tables will be deleted before starting the test by issuing the following commands in MySQL:**

*DELETE FROM products;*

*DELETE FROM product\_inventory;*

The figure below from phpMyAdmin shows that both tables now contain zero (0) rows:



Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Product insertion - test run scenario using BlueJ:**

->Insert the following product information:

*barcode='27607381', uom\_id=6; description= 'Whole Fresh Milk 6 pints', price=1.30; vat\_code=’v0’*

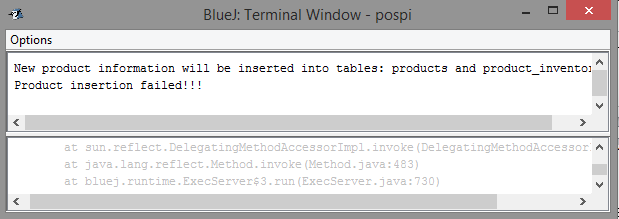
->Set the product\_inventory insert-statement to null to fail the transaction:

*insertStock = null;*

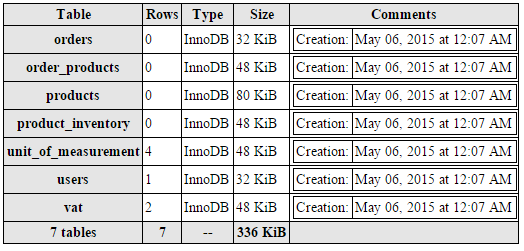
->Insert the following product\_inventory information:

*product\_id=(get auto-generated key from product insert), inventory\_date=(current date); quantity\_in\_stock=50, reorder\_level=10*

Result:



Using phpMyAdmin, the figure below shows that no product information was inserted for both the products and product\_inventory tables. No records were inserted and the transaction was rolledback:



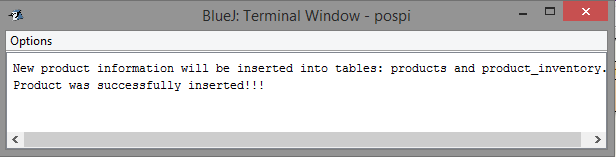
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

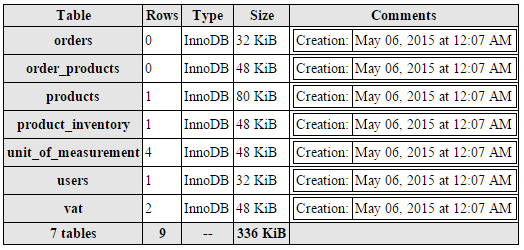
->Remove the statement that made the transaction fail:

// *insertStock = null;*

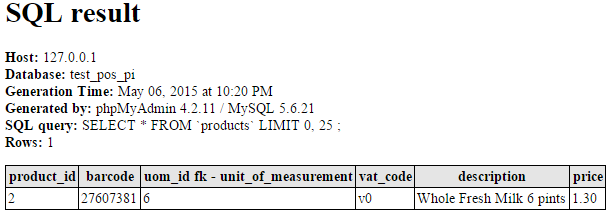
Result:



Using phpMyAdmin, the figure below shows that a product information was inserted for both the products and product\_inventory tables. The transaction was successfully committed and rows were inserted:

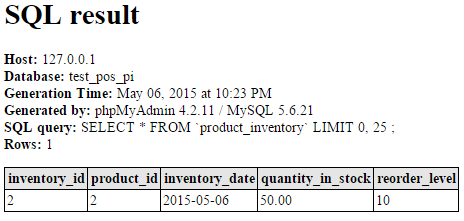


The figures below shows the rows that were inserted for both the products and product\_inventory tables after the transaction was successfully committed to the database:

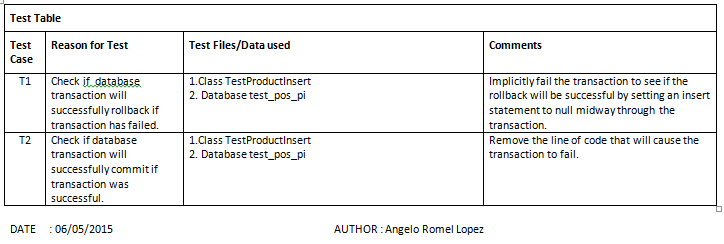


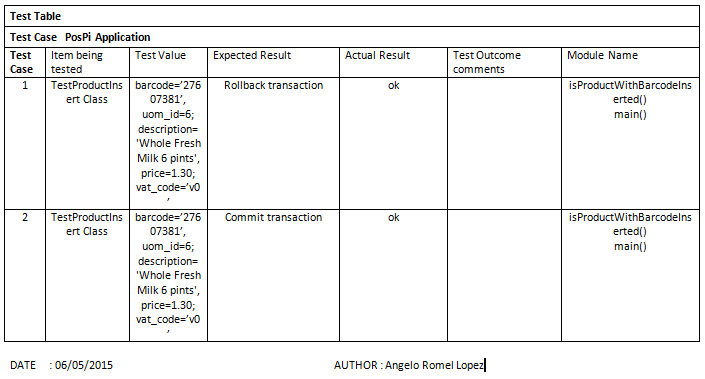
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]



**Test Log:**





Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Test code for product update using transaction:**

**The transaction includes updating an existing product record in the products table and product\_inventory**

package pospi;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class TestProductUpdate {

private static boolean isProductUpdated(){

Connection dbConnection;

PreparedStatement updateProduct = null,

updateStock = null;

try{

dbConnection = DriverManager.getConnection("jdbc:mysql://localhost/test\_pos\_pi",

"root", "raspberry");

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//Start transaction.

updateProduct = dbConnection.prepareStatement("UPDATE products SET barcode = ?, uom\_id = ?" +

", description = ?, price = ?, vat\_code = ? " +

"WHERE product\_id = ?");

updateProduct.setString(1, "27607381");

updateProduct.setInt(2, 6);

updateProduct.setString(3, "Whole Fresh Milk 12 pints");

updateProduct.setDouble(4, 2.00);

updateProduct.setString(5, "v0");

updateProduct.setInt(6, 2);

updateStock = dbConnection.prepareStatement("UPDATE product\_inventory SET quantity\_in\_stock = ? " +

"WHERE product\_id = ?");

updateStock.setDouble(1, 100);

updateStock.setInt(2, 2);

//Update product table.

updateProduct.executeUpdate();

//Update product\_inventory table.

updateStock.executeUpdate();

//Commit transaction

dbConnection.commit();

return true;

}//end try

catch(SQLException se){

se.printStackTrace();

return false;

}catch(Exception e){

e.printStackTrace();

return false;

}//end catch

finally{

//Do some clean-up.

dbConnection = null;

updateProduct = null;

updateStock = null;

}//end finally

}//end isProductUpdated().

public static void main(String[] args) {

System.out.println("Existing product information will be updated from tables: products and product\_inventory.");

if(isProductUpdated()){

System.out.println("Product was successfully updated!!!");

}

else{

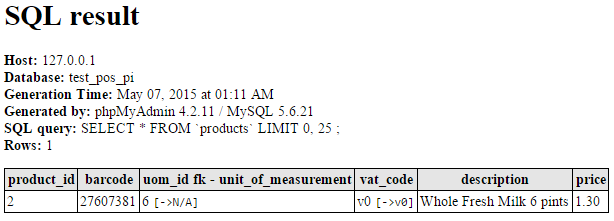
System.out.println("Product update failed!!!");

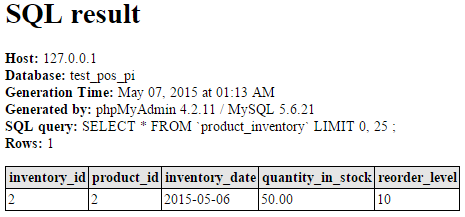
}//end if

}//end main.

}//end class.

The figure below shows the values of the record before the test was run.





Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Product update - test run scenario using BlueJ:**

->Updatethe following product information where product\_id = 2 and update record to:

*description= 'Whole Fresh Milk 12 pints', price=2.00*

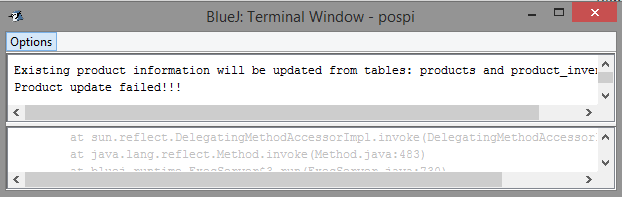
->Set the product\_inventory update-statement to null to fail the transaction:

*updateStock = null;*

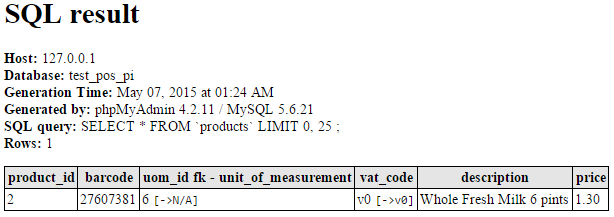
->Update the following product\_inventory information where product\_id = 2 and update record to:

*quantity\_in\_stock=100*

Result:

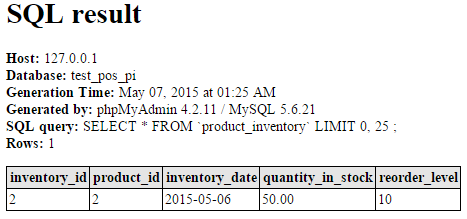


Using phpMyAdmin, the figures below shows that no product information was updated for both the products and product\_inventory tables. The transaction failed and was rolledback:



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

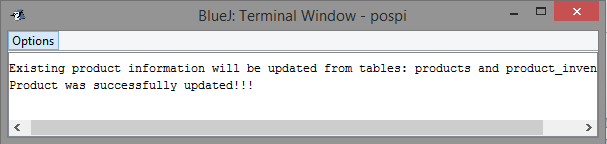
Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]



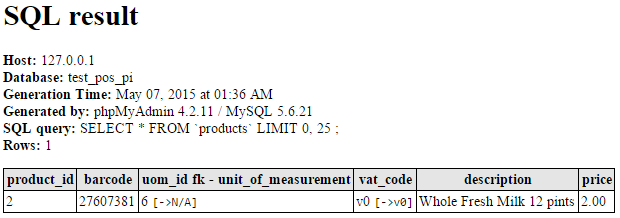
->Remove the statement that made the transaction fail:

// *updateStock = null;*

Result:

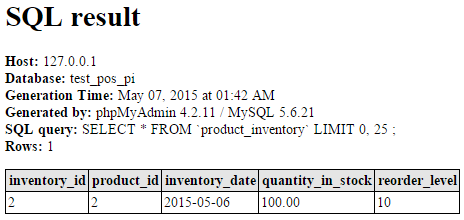


The figures below shows the rows that were updated for both the products and product\_inventory tables after the transaction was successfully committed to the database:

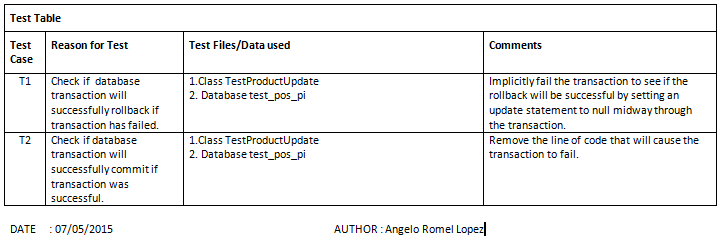


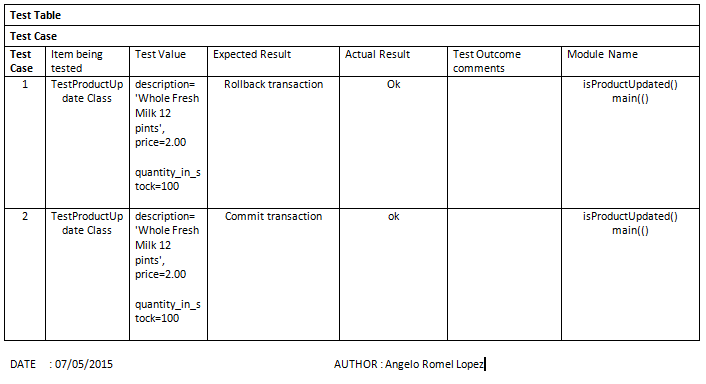
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]



**Test Log:**





Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Test code for product delete using transaction:**

**The transaction includes deleting an existing product record in the products table and product\_inventory**

package pospi;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class TestProductDelete {

private static boolean isProductDeleted(){

Connection dbConnection = null;

Statement deleteStatement = null;

try{

dbConnection = DriverManager.getConnection("jdbc:mysql://localhost/test\_pos\_pi",

"root", "raspberry");

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

deleteStatement = dbConnection.createStatement();

deleteStatement.execute("DELETE FROM product\_inventory WHERE product\_id = 2");

deleteStatement.execute("DELETE FROM products WHERE product\_id = 2");

//commit changes.

dbConnection.commit();

return true;

}//end try.

catch(SQLException se){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e) {

e.printStackTrace();

}

se.printStackTrace();

return false;

}//end catch.

catch(Exception e){

try {

dbConnection.rollback();//Rollback changes if transaction fails.

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

return false;

}//end catch.

finally{

//Do some clean-up.

dbConnection = null;

deleteStatement = null;

}//end finally.

}//end isProductDeleted()

public static void main(String[] args) {

System.out.println("Existing product information will be deleted from tables: products and product\_inventory.");

if(isProductDeleted()){

System.out.println("Product was successfully deleted!!!");

}

else{

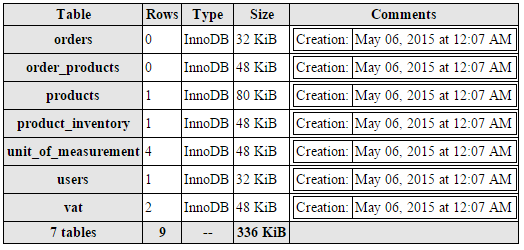
System.out.println("Product deletion failed!!!");

}//end if

}//end main().

}//end class.

The figure below shows that a record exists in the products and product\_inventory table. These are the records that the test code will try to delete.



**Product deletion - test run scenario using BlueJ:**

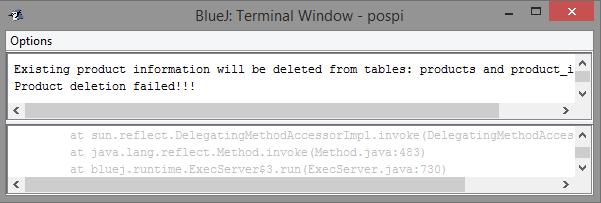
->Delete product information where product\_id = 2

->Set the product\_inventory delete-statement to null to fail the transaction:

*deleteStatement = null;*

->Delete product information where product\_id = 2

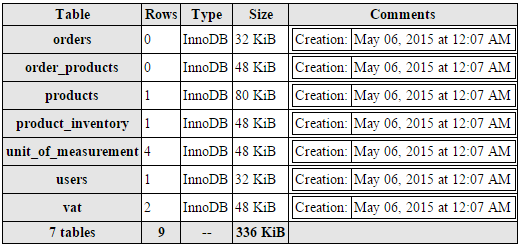
Result:



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

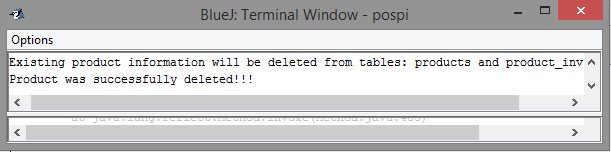
Using phpMyAdmin, the figure below shows that the transaction failed and no rows were deleted for the products and product\_inventory tables.



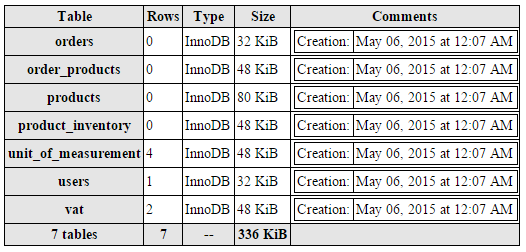
->Remove the statement that made the transaction fail:

// *deleteStatement = null;*

Result:



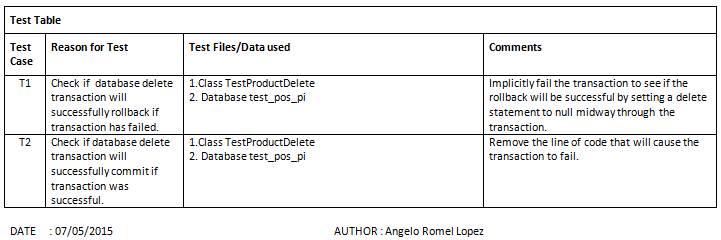
The figure below shows the rows that were successfully deleted for both the products and product\_inventory tables after the transaction was successfully committed to the database:

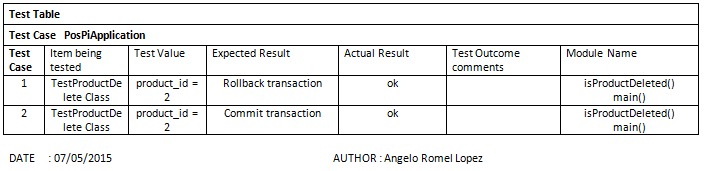


Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Test Log:**





**Test code for order insert using transaction:**

**The transaction includes inserting a new row in the orders and order\_products table.**

package pospi;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.JOptionPane;

public class TestOrderInsert {

private static boolean isSaleTransactionCommitted(){

Connection dbConnection = null;

Statement insertStatement = null;

PreparedStatement updateStatement = null;

int orderPK;//Store auto-generated primary key value.

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

ResultSet rs = null;//For holding auto-generated keys.

try{

dbConnection = DriverManager.getConnection("jdbc:mysql://localhost/test\_pos\_pi",

"root", "raspberry");

//Indicate manual transaction.

dbConnection.setAutoCommit(false);

//----------------Start of transaction.----------------

insertStatement = dbConnection.createStatement();

insertStatement.executeUpdate("INSERT INTO orders " +

" (order\_date) VALUES ('" + dateFormat.format(new Date()) + "')", Statement.RETURN\_GENERATED\_KEYS);

//Get the auto-increment primary key value.

rs = insertStatement.getGeneratedKeys();

if(rs.next()){

orderPK = rs.getInt(1);//Store PK value.

//Insert a new record in the order\_products table.

insertStatement = dbConnection.createStatement();

insertStatement.executeUpdate("INSERT INTO order\_products " +

"(order\_id, product\_id) VALUES (" + orderPK + ", 3)");

//Update inventory table.

updateStatement = dbConnection.prepareStatement("UPDATE product\_inventory " +

"SET quantity\_in\_stock = (quantity\_in\_stock - ?)" +

" WHERE product\_id = ?");

updateStatement.setDouble(1, 1);

updateStatement.setInt(2, 3);

updateStatement.executeUpdate();

//Commit transaction.

dbConnection.commit();

return true;

}

else{

JOptionPane.showMessageDialog(null, "Unable to retrieve orders\nPK value.",

"Error PK retrieval.",

JOptionPane.WARNING\_MESSAGE);

throw new SQLException();//Throw an exception and rollback changes.

}//end if

}//end try

catch(SQLException sqlEx){

sqlEx.printStackTrace();

return false;

}//end catch

catch(Exception eEx){

eEx.printStackTrace();

return false;

}//end catch

finally{

//Do some clean-up.

dbConnection = null;

insertStatement = null;

updateStatement = null;

}//end finally

}//end isSaleTransactionCommitted()

public static void main(String[] args) {

System.out.println("A new record will be inserted in the orders and order\_details table.");

if(isSaleTransactionCommitted()){

System.out.println("Rercord/s were successfully inserted!!!");

}

else{

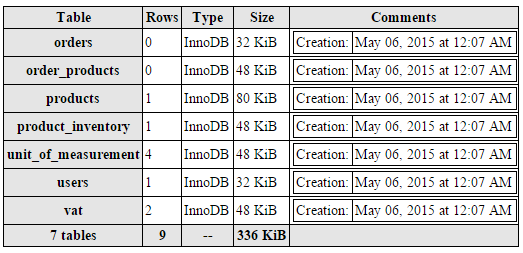
System.out.println("Record insertion failed!!!");

}//end if

}//end main().

}//end class.

The figure below shows that no rows exist in the orders and order\_products tables.



Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Order insertion - test run scenario using BlueJ:**

->Insert the following orders information:

*order\_id = (auto\_generated), order\_date = (new system date)*

->Insert the following order\_products information:

*order\_id = (get auto\_generated id from orders), product\_id = 3*

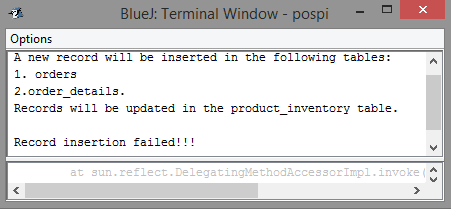
->Set the product\_inventory update-statement to null to fail the transaction:

*updateStatement = null;*

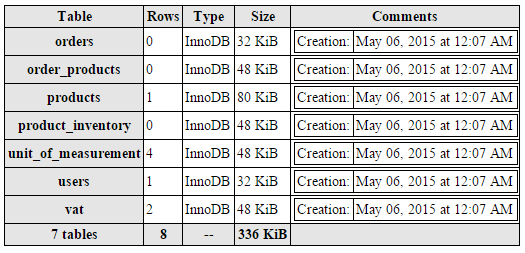
->Update the following product\_inventory information:

*quantity\_in\_stock -1;*

Result:



Using phpMyAdmin, the figure below shows that the transaction failed and no rows were inserted and/or updated in the orders, product\_orders and product\_inventory tables.



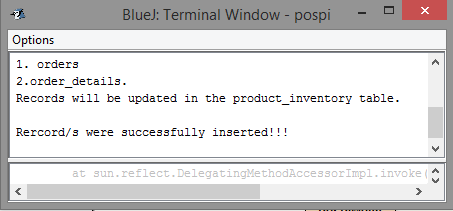
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

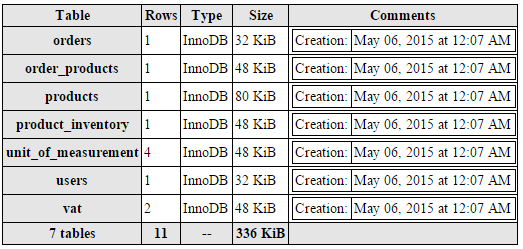
->Remove the statement that made the transaction fail:

// *updateStatement = null;*

Result:



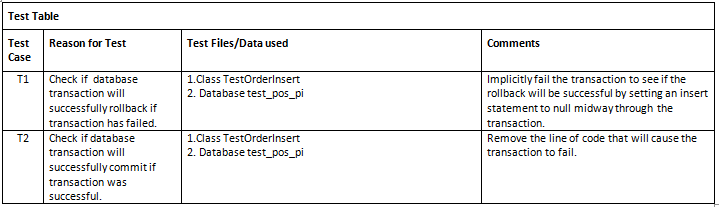
The figure below shows the rows that were successfully inserted/updated for the orders, order\_products and the product\_inventory tables after the transaction was successfully committed to the database:

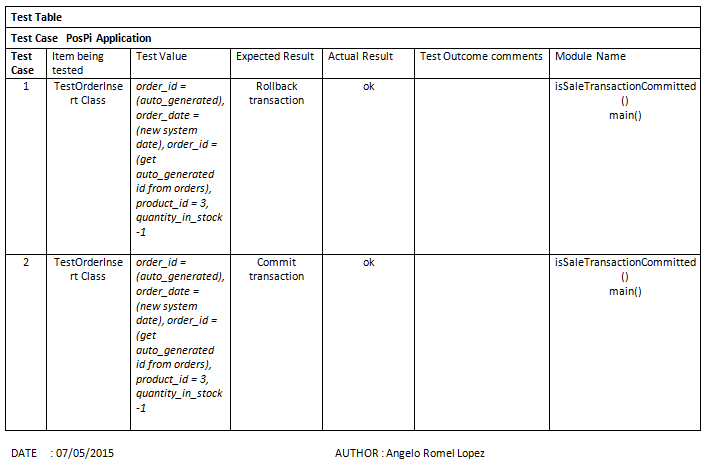


Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 05, 2014]

**Test Log:**





## 3.3. User document/guide

**POS-Pi**

**Point of Sale Application**

(Working Prototype)



**User Guide**

Author: Angelo Romel Lopez

**OVERVIEW**

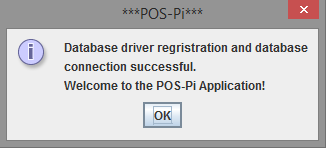
POS-Pi is a prototype for a point of sale application designed to run on a Raspberry Pi device. It has the additional feature of giving the user the ability to add, modify and delete product information. The graphical user interface was designed to be viewed and used on small touch screen (2.8” and 3.2”) form factor. The POS-Pi can accept input from a barcode scanner, finger-touch; stylus-pen, mouse; virtual and physical keyboard.

**GETTING STARTED**

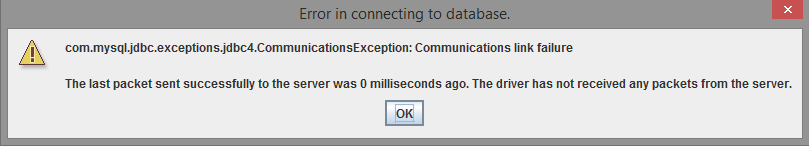
Power on the device by connecting one end (micro-USB) of the USB cable to the 5V micro-USB power port of the Raspberry Pi and the other end to the 2.1A output of the portable USB power bank. Type in the username “pi” and the password “raspberry” when prompted. After a successful login, type in the command “startx” (without the double quotes) to start the Raspbian operating system in GUI mode. Start the POS-Pi application by launching BlueJ (Java IDE) and right-clicking on the PosPiApplication class which is displayed on the BlueJ class diagram screen (the main screen that you see). Click on the “void main(String[] args)” option from the context-menu that pops up.

**Logging On**

The first thing that you will see is the welcome screen and the status of your database connection:



If the database server has failed to start when you boot-up the Raspberry Pi, you will see this message pop-up:



Type this command from a terminal/console window to start the MySQL service:

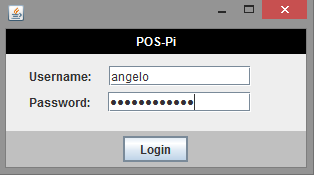
>sudo service mysql start

or

>sudo service mysql restart

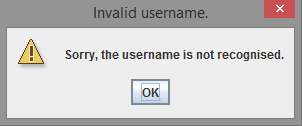
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

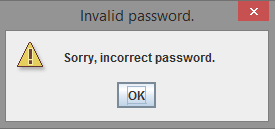
After the database driver has been successfully registered and the connection to the database server established, you will then see the POS-Pi login screen:



Type your username and password (default username=angelo password=raspberry123), then press the “**Login**” button to login.

You will see the following messages if you have typed your username or password incorrectly:



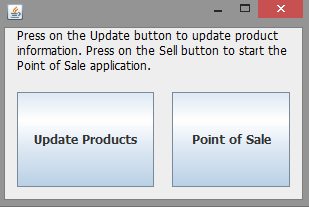


Typing the correct username and password will take you to the main menu.

Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

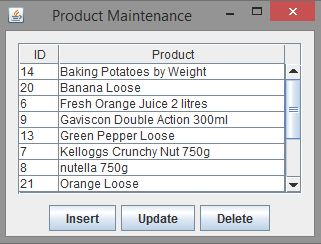
**MAIN MENU**

The main menu (figure below) is the main screen you’ll see after a successful login. The two options provided on the menu allow you to 1) manage product information by adding, modifying and deleting products stored in the database. Click or tap the “**Update Products**” button to start managing product information. 2) You will also be able to access the main point-of-sale application from this screen to start a sale transaction by clicking or tapping on the “**Point of Sale**” button.



**PRODUCT MAINTENANCE**

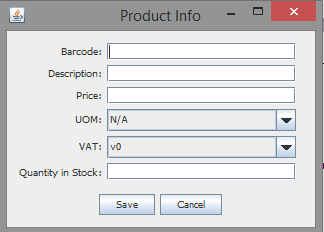
The Product Maintenance screen is accessed by choosing the **Update Products** option from the Main Menu. The products listed on the screen are the products currently stored in the database. You can choose to add, modify or delete a product information by clicking on any of the three buttons at the bottom of the screen.



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Adding a New Product**

Clicking or tapping on the “**Insert**” button at Product Maintenance screen will take you to a data entry screen where you can add a new product information.



Supply the following information by typing directly into the input boxes provided:

1.Barcode – The barcode for the product that you are trying to add to the database. I have used an online tool from [www.scandit.com](http://www.scandit.com) to generate the barcode symbols for this project.

Scandit Inc, nd. Barcode Generator, [online tool], Scandit Inc. Available at: <http://www.scandit.com/barcode-generator/> [Accessed February 10, 2015]

2.Description – A brief description of the product.

3.Price – The selling amount (VAT not yet included).

4.UOM- The unit-of-measurement code for a product if the product cannot have a barcode. Please note that a product can only have a barcode or a uom code, but not both. The application will give you a warning message if you supply both.

5.VAT – The code for the value added tax for a product if applicable ( v20 = 20% vat, v0=vat excempt).

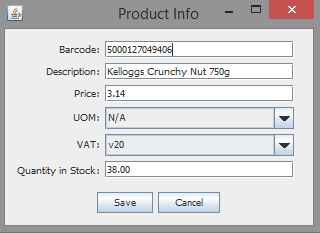
6.Quantity in Stock – The amount of items the product has in stock.

Click or tap on the “**Save**” button to save your entry, or choose “**Cancel**” to cancel and close the data entry screen.

Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Modifying an Existing Product Information**

Choose the product that you wish to edit from the Product Maintenance screen simply by selecting it. Once it is highlighted, click or tap on the “**Update**” button at the bottom of the screen. You will then be taken to the data entry screen where you can make the changes to the product information.



Once you are done editing, you can click or tap on the “**Save**” button to save your entry, or choose “**Cancel**” to cancel and close the data entry screen.

**Deleting an Existing Product**

Please be aware of the following facts before you to decide to delete a product information from the database.

1.Please be warned that products deleted from the database cannot be easily undone.

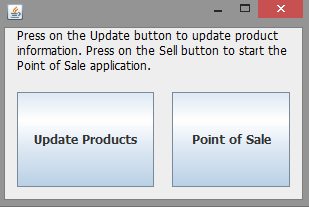
2.You cannot delete a product information if it is being used/referenced in another table like the order\_detail and the product\_inventory table.

If you have decided and are sure that the product information is no longer needed and referenced, then deleting the product information from the database is as easy as selecting it from the product list on the Product Maintenance screen; and clicking or tapping the “**Delete**” button at the bottom of the screen. Press ok to confirm your decision.

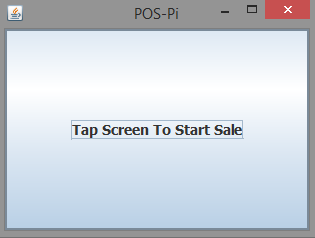
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**POINT of SALE**

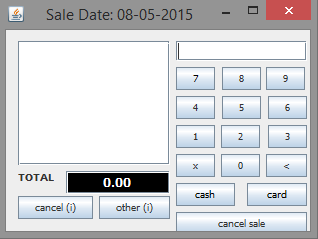
Click or tap on the “**Point of Sale**” button from the Main Menu screen.



Tap on the big button that says “**Tap Screen to Start Sale**” on the next screen that appears…



and you’ll be taken to the Point of Sale transaction screen (see image below).



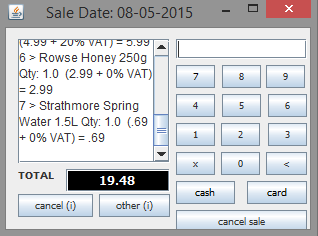
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Adding Items to the Shopping Basket**

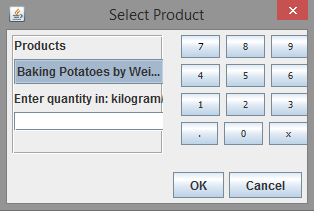
You can use a barcode scanner/reader to start adding items to the shopping basket located at the left-hand side of the point-of-sale screen. Barcodes scanned with the barcode scanner will automatically be recognized by the application (as long as the barcode and the corresponding product information is stored in the database) and the corresponding item will be displayed in the shopping basket and the VAT and total amount calculated and displayed. Alternatively, you can use the numeric pad on the right-hand side of the screen (see image below) to enter a product’s barcode if a barcode scanner/reader is not available.



The figure below shows the shopping basket containing several items/products.

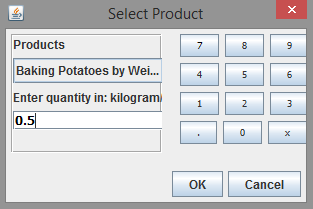


Additional products that do not have any barcode can be accessed by clicking on the “**other (i)**” button. You will then be able to add products to the shopping basket by unit-of-measurement. Select the desired product from the drop-down list.



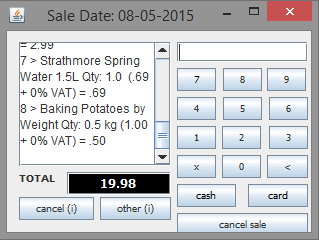
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

Use the numeric keypad on the left-hand side of the screen to enter the amount of unit-of-measurement. We have entered 0.5 (1/2) Kilograms of baking potatoes for this example, as shown in the image below.



Click or tap on the “**OK**” button when you’re done.

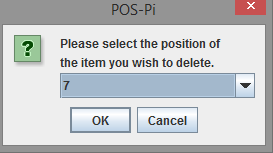
As you can see from the image below, the item/product was added to the shopping basket. The VAT (if applicable) and the total amount is calculated and displayed on the screen. The price for the potatoes is £1 per 1 kilogram (the prices for the products in the database does not reflect real-life actual prices). 0.5kg \* £1.00 = 0.50p.



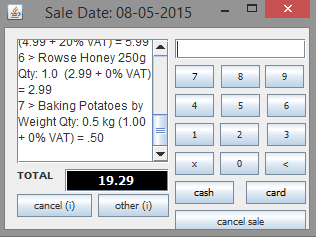
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Item Cancellation**

You can cancel an item by removing it from the shopping basket. The product information displayed on the shopping basket is always preceded by a line number. The item “7>Strathmore Spring Water” is indicated by line number seven (>7). To delete this product from the shopping basket, simply press on the “**cancel (i)**” button and select the number seven (7) from the drop-down list.



Press or tap on the “**OK**” button to remove the product from the shopping basket. You can see the updated shopping basket and total amount from the image below.



**Cancelling a Sale Transaction**

Click or tap on the “**cancel sale**” button at any time during the transaction to cancel the current sale and be taken out of the point-of-sale screen.

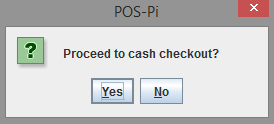
Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

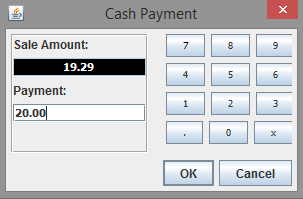
**The Checkout**

If there are no more product/items to be added to the shopping basket, then you can press the “**cash**” button to checkout. At the moment, only cash payments are allowed by the application. The “**card**” button, although not yet implemented; is a feature that will enable the application to accept debit and credit card payments and is in the priority list for future improvements of the application.

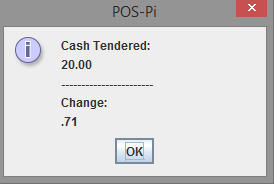
**Cash Payment Checkout**

To finalize the sale using the cash payment method, press the “**cash**” button to start the checkout process. After a confirmation screen has been displayed, a screen will appear displaying the amount to be paid and an input box to enter the amount that the customer has tendered.

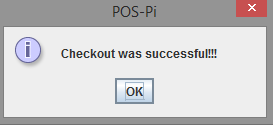




As show from the image above, the amount to be paid is £19.29. Use the numeric keypad located on the right-hand side of the screen to enter the amount tendered. After pressing the “OK” button, a screen will pop up showing the amount that the customer has tendered and the amount of change (if applicable).



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]



A screen will pop up indicating the status of the checkout after pressing the “OK” button. In this case, the checkout process was successful as shown in the image above.

A receipt will automatically be printed out if a printer is connected to the Raspberry Pi device. Otherwise, a warning message will appear if there are no printers available to print the receipt. In either case, the sale transaction will still be committed to the database.

The image below is from an actual receipt printout:



Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

**Resources Used in the Development of the POS-Pi**

**(Point of Sale** A**pplication on the Raspberry Pi)**

[1] **Javadoc**

Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

[2] **Visual Paradigm**

Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. Available at: <http://www.visual-paradigm.com/> [Accessed April 29, 2015].

[3] **ArgoUML**

Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. Available at: <http://sourceforge.net/projects/argouml/> [Accessed April 29, 2015].

[4] **Code Rocket for Eclipse**

Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. Available at: <http://www.rapidqualitysystems.com/> [Accessed April 29, 2015]

[5**] phpMyAdmin**

Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

[6] **BlueJ**

Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

[7] **XAMPP**

Apache Friends, nd. XAMPP Control Panel, [computer program]. Apache Friends. Available at: <https://www.apachefriends.org/download.html> [Accessed May 06, 2015]

[8] **Barcode Generator**

Scandit Inc, nd. Barcode Generator, [online tool], Scandit Inc. Available at: <http://www.scandit.com/barcode-generator/> [Accessed February 10, 2015]

[9] **Eclipse IDE**

Eclipse Foundation, nd. Eclipse IDE Luna Service Release, [computer program]. Eclipse Foundation. Available at: <www.eclipse.org> [Accessed 2014]

# 4. Evaluating

**Evaluation of Software Quality**

In order to produce an impartial evaluation of the software prototype, the original requirement specification defined in the planning stage [2] (page 7 of this report) will be evaluated for this project; against the 6 main software quality characteristics/criteria defined in the ISO 9126 (http://www.sqa.net/iso9126.html) software quality model. Reference will also be made to the source code documentation to ensure that proper naming convention and design pattern (Model-View-Controller) have been observed throughout the development of the software. Reference will also be made to the Testing/Implementation phase [3.2] (page 308 of this report) and to the User Document/ Guide [3.3] on page 338 to provide evidence that the finished software complied with the requirements specification.

Below is an excerpt (table included) from an article on the sqa.net (Software Quality Assurance) website. It provides a detailed description on the international standard used for the evaluation of software:

*“The ISO 9126 documentation itself, from the official ISO 9126 documentation, can only be purchased and is subject to copyright. SQA.net only reproduces the basic structure of the ISO 9126 standard and any descriptions, commentary or guidance are original material based on public domain information as well as our own experience.   
  
The ISO 9126-1 software quality model identifies 6 main quality characteristics, namely:*

*Functionality*

*Reliability*

*Usability*

*Efficiency*

*Maintainability*

*Portability”*

*“The full table of Characteristics and Subcharacteristics for the ISO 9126-1 Quality Model is:-*

|  |  |  |
| --- | --- | --- |
| Characteristics | Subcharacteristics | Definitions |
|  | Suitability | This is the essential Functionality characteristic and refers to the appropriateness (to specification) of the functions of the software. |
|  | Accurateness | This refers to the correctness of the functions, an ATM may provide a cash dispensing function but is the amount correct? |
| Functionality | Interoperability | A given software component or system does not typically function in isolation. This subcharacteristic concerns the ability of a software component to interact with other components or systems. |
|  | Compliance | Where appropriate certain industry (or government) laws and guidelines need to be complied with, i.e. SOX. This subcharacteristic addresses the compliant capability of software. |
|  | Security | This subcharacteristic relates to unauthorized access to the software functions. |
|  | | |
|  | Maturity | This subcharacteristic concerns frequency of failure of the software. |
| Reliability | Fault tolerance | The ability of software to withstand (and recover) from component, or environmental, failure. |
|  | Recoverability | Ability to bring back a failed system to full operation, including data and network connections. |
|  | | |
|  | Understandability | Determines the ease of which the systems functions can be understood, relates to user mental models in Human Computer Interaction methods. |
| Usability | Learnability | Learning effort for different users, i.e. novice, expert, casual etc. |
|  | Operability | Ability of the software to be easily operated by a given user in a given environment. |
|  | | |
| Efficiency | Time behavior | Characterizes response times for a given thru put, i.e. transaction rate. |
|  | Resource behavior | Characterizes resources used, i.e. memory, cpu, disk and network usage. |
|  | | |
|  | Analyzability | Characterizes the ability to identify the root cause of a failure within the software. |
| Maintainability | Changeability | Characterizes the amount of effort to change a system. |
|  | Stability | Characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes. |
|  | Testability | Characterizes the effort needed to verify (test) a system change. |
|  | | |
|  | Adaptability | Characterizes the ability of the system to change to new specifications or operating environments. |
| Portability | Installability | Characterizes the effort required to install the software. |
|  | Conformance | Similar to compliance for functionality, but this characteristic relates to portability. One example would be Open SQL conformance which relates to portability of database used. |
|  | Replaceability | Characterizes the plug and play aspect of software components, that is how easy is it to exchange a given software component within a specified environment. |

“

*Software Quality Assurance (nd.) ISO 9126 Software Quality Characteristics [website] available from: http://www.sqa.net/iso9126.html [Accessed 25/05/2015]*

**EVALUATION**

**Functionality**

Essential functions as described in the requirement specification:

1. **Retrieve username and password** (Planning Stage [2], Initial Investigation [2.1], page 7):

- Query MySQL database and retrieve user name and password where user name and password is equal to the input user name and password.

Implementation:

The software provides a login screen that prompts the user for a valid username and password and searches the MySQL database for a match.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], UserLoginController class on page 49. The source code listing shows how this functionality was implemented.

-**public boolean isUserValid(String userName, char[] password)** on page 58.

1. Refer to the Testing/Implementation [3.2] stage, Database registration and connection test run scenario on page 312. Test case #3 on the test log on page 313 shows a successful outcome.
2. **Retrieve item description and unit price** (Planning Stage [2], Initial Investigation [2.1], page 8):

**-** Query MySQL database and retrieve item description and unit price where item barcode is equal to the input barcode.

Implementation:

The software provides a point-of-sale screen where a barcode scanner/reader or a keypad (physical/virtual keyboard) can be used to read/input a product’s barcode. The software will then check the database stored in MySQL server and return the corresponding product information (including description and unit price) if a match exists.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PointOfSaleController class on page 120. The source code listing shows how this functionality was implemented.

-**public void getItemFromBarcode()** on page 130.

-**public void getItemFromUOM(int productID, double quantity)** on page 132.

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PointOfSaleView class on page 88. The source code listing shows how this functionality was presented.
2. Refer to the Testing/Implementation [3.2] stage, Order insertion test run scenario on page 335. The test code on page 331 was the code used to implement the order and order –details insert transaction in the actual controller class to insert product information from a barcode input.
3. **Manage cash payment** (Planning Stage [2], Initial Investigation [2.1], page 8):

**-** If cash amount is greater than or equal to the amount to be paid then calculate change and save transaction to database.

Implementation:

The software provides a cash payment checkout screen where the total amount to be paid is displayed and an input box to enter the cash payment amount is provided. Once the cash amount has been accepted (cash amount is greater than or equal to the amount to be paid), the amount tendered and the amount of change (if applicable) is displayed. The sale will then be recorded in the database as a result of a successful transaction.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PayByCashDialog class on page 159. The source code listing shows how this functionality was implemented.

-**public boolean isPaymentEnough(double paymentAmount)** on page 179.

1. Refer to the Testing/Implementation [3.2] stage, Order insertion test run scenario on page 335. The test code on page 331 was the code used to implement the order and order –details insert transaction in the actual controller class after a successful cash payment checkout.
2. **Reduce stock amount** (Planning Stage [2], Initial Investigation [2.1], page 9):

**-** Item stock amount – 1.

Implementation:

The stock amount (product\_inventory table) for a particular product will be deducted by the amount of items sold for that product as part of a successful sale transaction.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PointOfSaleController class on page 120. The source code listing shows how this functionality was implemented.

-**public boolean commitSaleTransaction()** on page 137.

1. Refer to the Testing/Implementation [3.2] stage, Order insertion test run scenario on page 335. The test code on page 331 was the code used to implement the order and order –details insert transaction in the actual controller class to update the product\_inventory table. The amount of items sold will be deducted for a particular product as part of a successful sale transaction.
2. **Cancel sale** (Planning Stage [2], Initial Investigation [2.1], page 9):

- Cancel a sale transaction.

Implementation:

Clicking or tapping on the “**cancel sale**” button at any time during the sale transaction will cancel the current sale and be taken out of the point-of-sale screen.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PointOfSaleController class on page 120. The source code listing shows how this functionality was implemented.

-**public void actionPerformed(ActionEvent e)**, case: “cancel sale” on page 129.

1. Refer to the User Document/Guide [3.3,] Cancelling a Sale Transaction on page 347.
2. **Print receipt** (Planning Stage [2], Initial Investigation [2.1], page 9):

**-** Print items sold, sub-total and amount to be paid.

Implementation:

After a successful checkout, a receipt will automatically be printed out if a printer is connected to the Raspberry Pi device.

Evidence:

1. Refer to the Developing [3] stage, Design/Production of Application [3.1], PointOfSaleController class on page 120. The source code listing shows how this functionality was implemented.

-**public void printReceipt()** on page 140.

-**public int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException** on page 141.

1. Refer to the User Document/Guide [3.3,] Cash Payment Checkout on page 348. Page 349 shows an image of an actual receipt printout.
2. **Insert discount on items** (Planning Stage [2], Initial Investigation [2.1], page 10):

- Insert product + discount.

\*This feature was not implemented due to design and time constraints.

1. **Update discount on items** (Planning Stage [2], Initial Investigation [2.1], page 10):

-Update discount.

\*This feature was not implemented due to design and time constraints.

1. **Delete discount on items** (Planning Stage [2], Initial Investigation [2.1], page 10):

**-**Item discount = 0.

**\***This feature was not implemented due to design and time constraints.

**Reliability**

1. **Fault tolerance** – The software currently relies on two external software components to run successfully. These components must be correctly installed, up and running for the software to function. Failure to start these components or services or the ability to maintain its service for a defined period of time will cause the software to fail.
2. Apache HTTP Server – this component must be started and must be able to maintain its state for defined periods of time.
3. MySQL Community Server – this component must be started and the database available for defined periods of time. The database community server supports database replication, database backup and restore. The database backup can be stored and compressed using different formats. However, scheduled backups do not seem to be available with the current version of the database server installed.
4. **Load test** – The server is currently installed locally on the same device where the software is installed. Load testing was performed on a single user connection per account with the max\_connection variable set to 150 (default) and max\_user\_connection set to 0 (dafault). 150 is the maximum amount permitted for simultaneous client connections while the maximum amount for concurrent client connection per account is unlimited.
5. **Feature test** – Software features were tested and checked in the testing stage of the development. Please refer to the Testing/Implementation phase [3.2] starting on page 308 of this document.

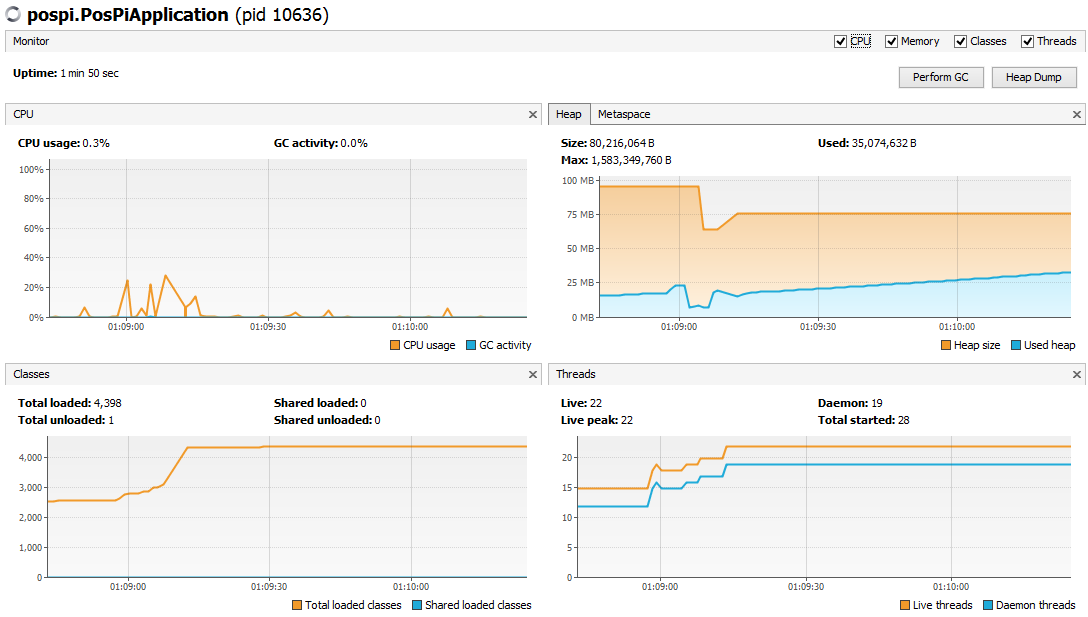
**Usability**

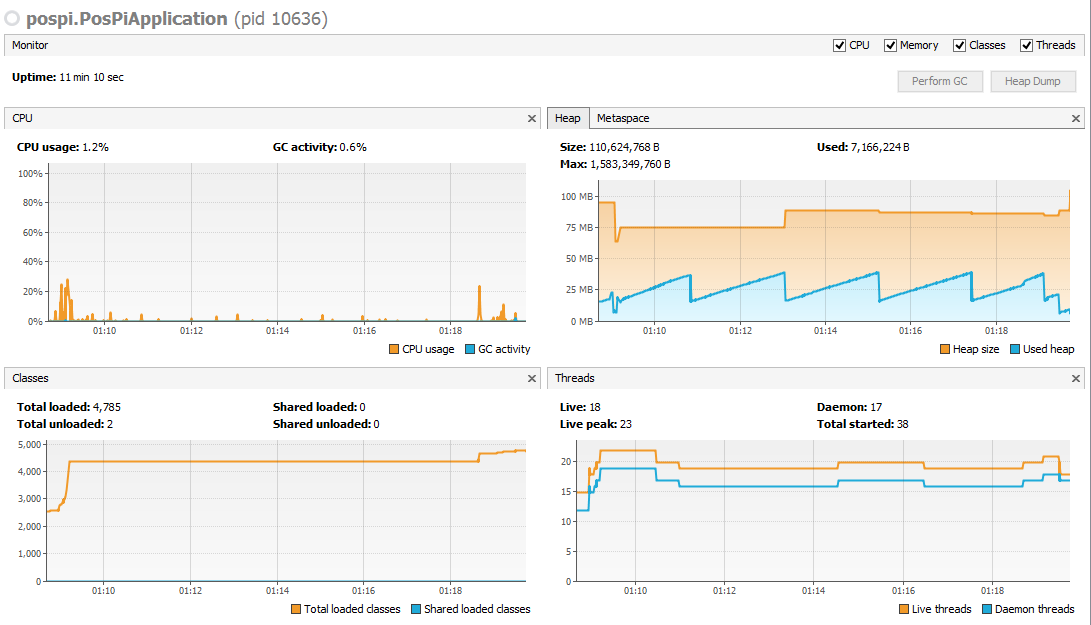
1. The graphical user interface of the software was designed to display information that is easily accessible on a smaller screen.
2. The software can accept input from multiple sources.
3. Information is presented in a clear and concise manner.
4. The design is minimal. Information that is irrelevant is not displayed.
5. Dialog boxes show one information at a time.
6. Input boxes accept one input at a time.
7. Virtual numeric keypad is displayed on-screen during a sale transaction as an alternative way to facilitate the input of the barcode, item quantity, payment amount, etc.
8. Error Handling was used were the code would likely throw an exception.
9. Feedback is displayed where and when appropriate.
10. User document and guide is included within this document to show people step-by-step instruction on how to use the software. The user document includes screenshots to give people a visual guide of the instructions.

**Efficiency**

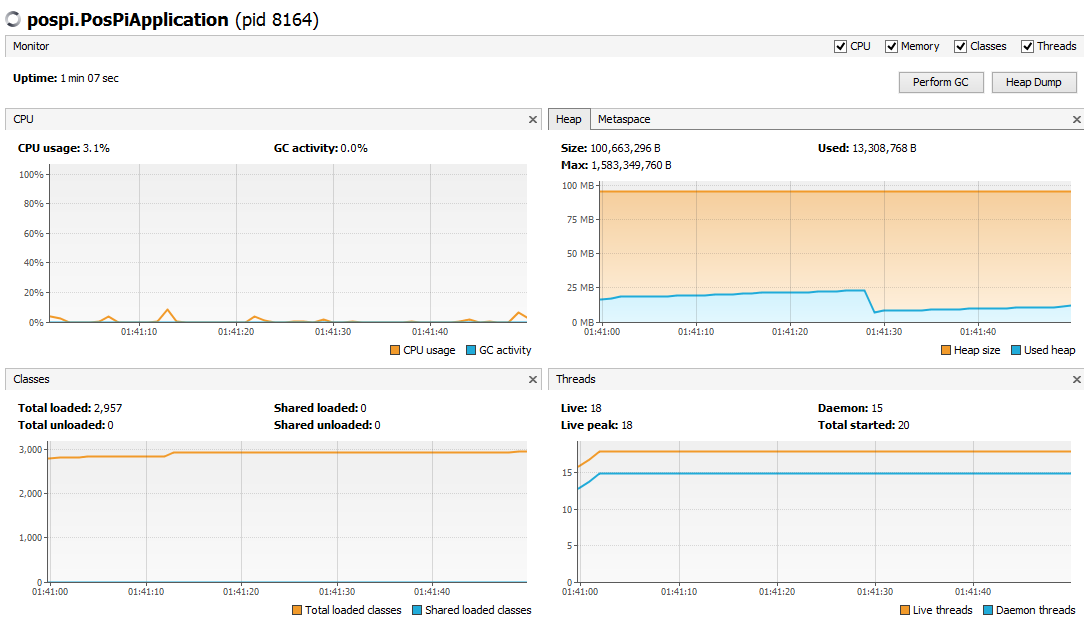
The screenshots below show the resource usage of the POS-Pi application. Data was obtained using a lightweight profiling tool called the “VisualVM – All in One Java Troubleshooting Tool” available at the VisualVM website (https://visualvm.java.net/?VisualVM\_1.3.8). Data was captured over a period of 29 minutes.

[1] Database connection and registration - (CPU usage: 0.3%, Used Heap: 35.075mb)

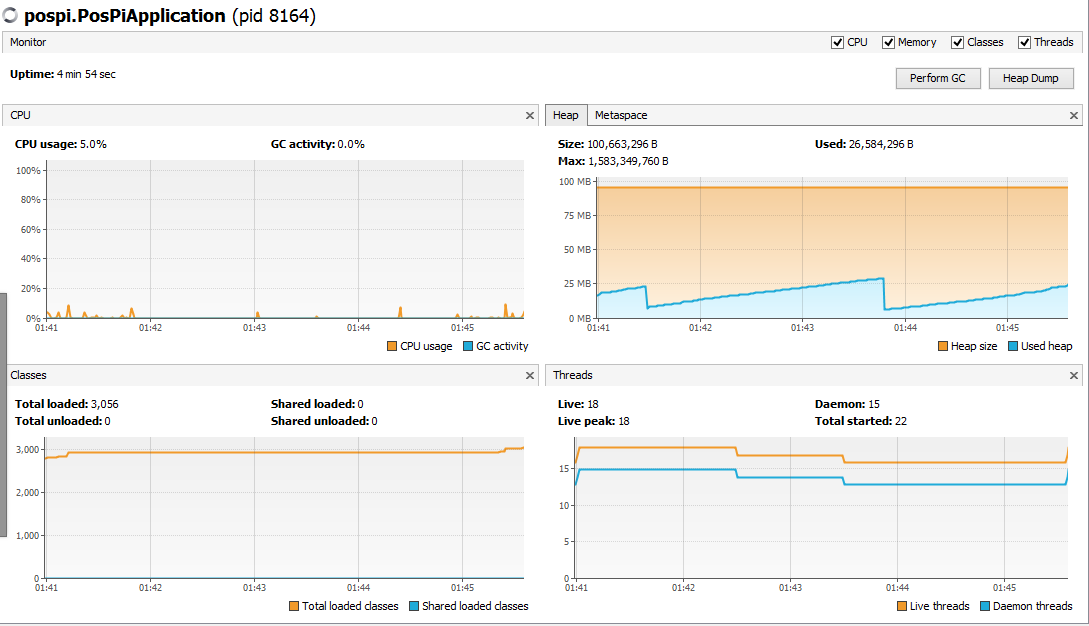


[2] User authentication - (CPU usage: 1.2%, Used Heap: 7.166mb)

[3] Display product list - (CPU usage: 3.1%, Used Heap: 13.309mb)

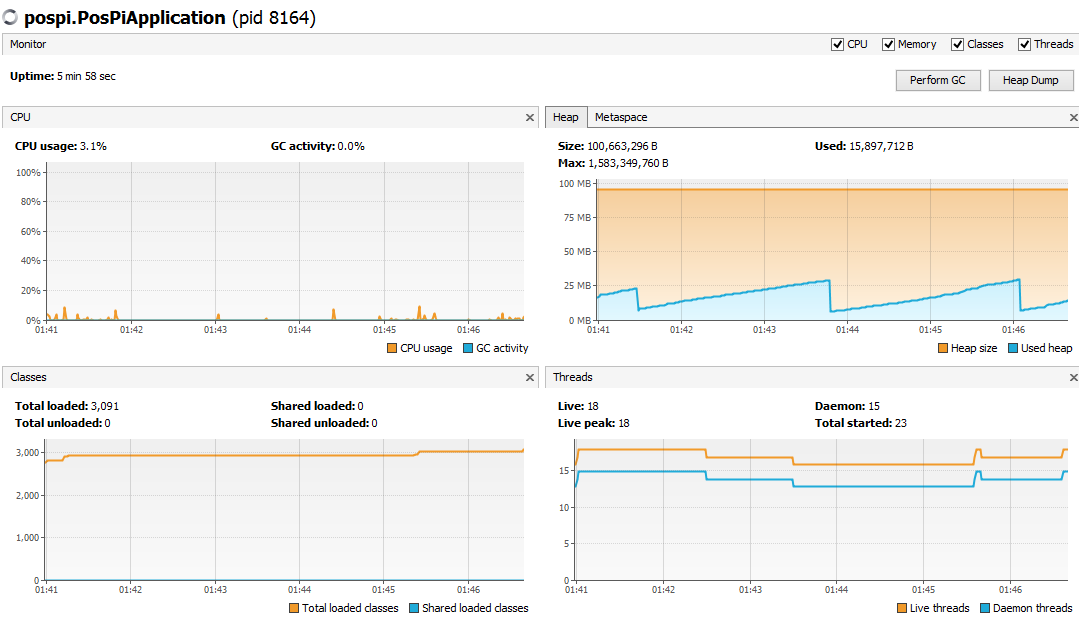


[4] Updating product information - (CPU usage: 5.0%, Used Heap: 26.584mb)

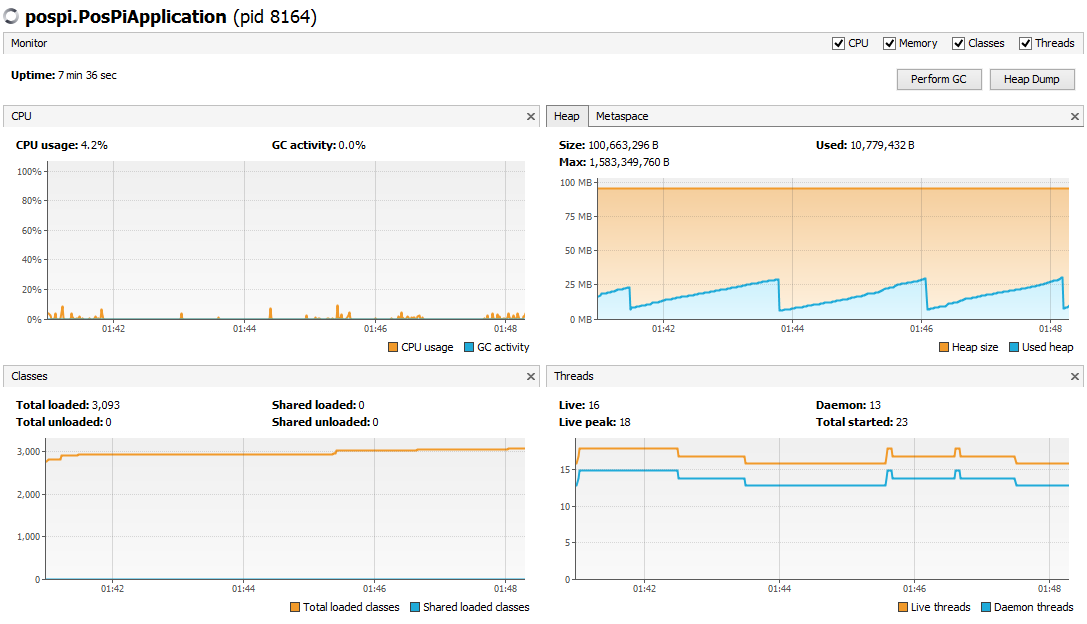


Jiri Sedlacek and Tomas Hurka, nd. VisualVM, [computer program]. Java.net. Available at: <https://visualvm.java.net/?VisualVM\_1.3.8> [Accessed May 26, 2015] [3] Display product list - (CPU usage: 3.1%, Used Heap: 13.309mb)

[5] Inserting product information - (CPU usage: 3.1%, Used Heap: 15.898mb)

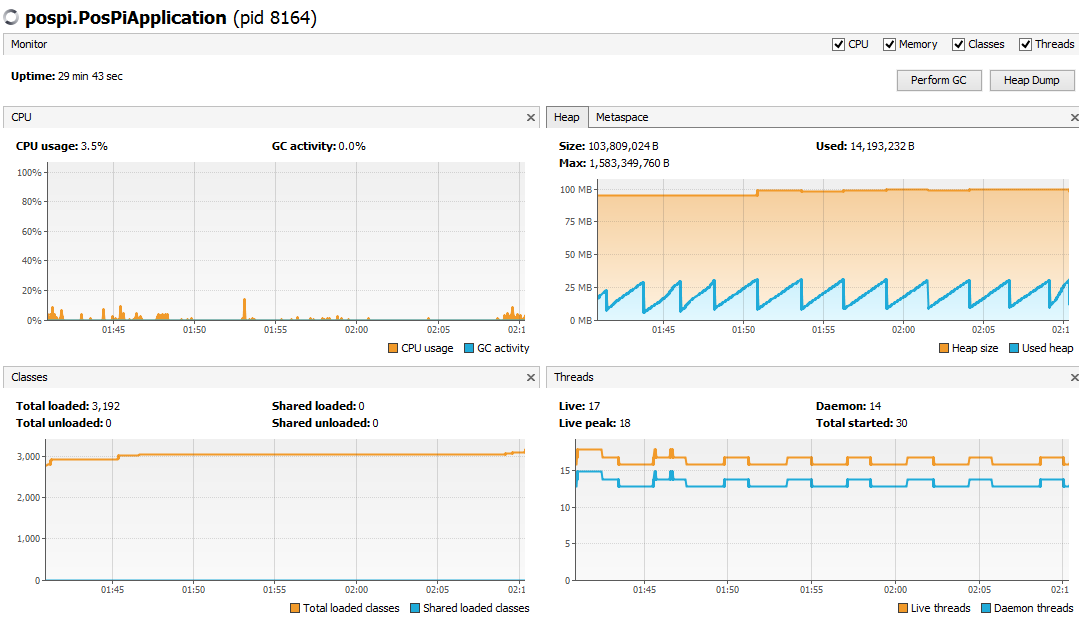


[6] Deleting product information - (CPU usage: 4.2%, Used Heap: 10.779mb)



Jiri Sedlacek and Tomas Hurka, nd. VisualVM, [computer program]. Java.net. Available at: <https://visualvm.java.net/?VisualVM\_1.3.8> [Accessed May 26, 2015] [3] Display product list - (CPU usage: 3.1%, Used Heap: 13.309mb)

[7] Checkout - (CPU usage: 3.5%, Used Heap: 14.193mb)



Maximum CPU usage of 5.0% occurs when product information is being updated, while maximum memory usage of 35.075MB occurs during database connection and registration. Overall performance on a regular computer with 8GB ram and dual core processor is excellent. However, the 512MB maximum memory capacity of the rapberry pi b+ model gave unimpressive result when running the software, although I believe that this is due to the fact that the MySQL Server that resides on the same device also contributed to resource usage.

If I had more time for this project, then I would allocate time to perform a thorough analysis and profiling on the MySQL Server to try and figure out where and how to make performance improvements and where the bottleneck occurs.

Jiri Sedlacek and Tomas Hurka, nd. VisualVM, [computer program]. Java.net. Available at: <https://visualvm.java.net/?VisualVM\_1.3.8> [Accessed May 26, 2015]

**Maintainability**

Since code readability and testability has a direct impact on the software’s level of maintainability, code conventions were observed and implemented throughout the design and development of the software.

Below is a list on how **code conventions** were implemented on the design and development of the POS-Pi application:

* A Javadoc style comment appears at the top of every source file that lists the name of the author and the purpose of the class.
* A whitespace is used between operators.
* If an expression is continued on a newline, then it is aligned on the same indentation level as the first line.
* Single-line comments are used for short comment where the comment does not exceed a single line.
* Block comments are used for longer comments where comments span multiple lines.
* Javadoc block comments are used to describe classes, interfaces, constructors and methods.
* This pattern is used where Javadoc style block comments are used:

/\*\*

\*<Description of the method>

\*<parameter 1>

\*<parameter 2>

\*…

\*…

\*<return>

/\*\*

public int method(int parameter1, String parameter2)

…

…

* One declaration per line is used to declare and define class attributes.
* Class attribures are declared as private.
* Private attributes where appropriate are supplied with getters and setters.
* Local variables are declared and initialized at the beginning of a block where they are used.
* Class attributes are initialized within constructors.
* Camel casing is used for naming packages, classes, variables and methods.
* Class names start with an uppercase letter.
* Packages start with a lowercase letter.
* Attributes and variables start with a lowercase letter.
* Methods start with a lowercase letter.
* Verbs are used to name methods.
* Where a method returns a boolean value, the prefix “is” is used. Example: isUserNameValid().

**Portability**

The POS-Pi application is written in Java and was designed using the MVC (Model-View-Controller) pattern. The model is the database that resides in MySQL Community Server. Below is a list of the view and controller classes:

1. UserLoginView (page 40).
2. UserLoginController (page 49).
3. PointOfSaleView (page 79).
4. PointOfSaleController (page 106).
5. ProductView (page 231).
6. ProductController (page 240).
7. ProductMaintenanceView (page 255).
8. ProductMaintenanceController (page 270).

Since Java is an architecture neutral programming language, the software will run on any platform where the Java Virtual Machine is installed.

List of supported platforms:

1. Solaris x64(64 bit) and x86 (32 bit).
2. Windows 8, 7, Vista, 2008 Server, 2012 Server x64(64 bit) and x86 (32 bit).
3. Linux x64(64 bit) and x86 (32 bit).
4. Redhat Enterprise
5. Oracle Linux
6. Suse Enterprise
7. Ubuntu
8. Debian
9. Raspbian (for the Raspberry Pi)
10. Mac OS X

Source: http://www.oracle.com/technetwork/java/javase/config-417990.html

List of platforms that support MySQL database:

* Oracle Linux x64(64 bit) and x86 (32 bit).
* Oracle Solars x64(64 bit) and x86 (32 bit).
* Radhat Enterprise Linux x64(64 bit) and x86 (32 bit).
* Ubuntu x64(64 bit) and x86 (32 bit).
* Suse Enterprise Linux x64(64 bit) and x86 (32 bit).
* Debian GNU Linux 64(64 bit) and x86 (32 bit).
* Microsoft Windows 8, 7, Vista, XP, 2003 Server, 2008 Server, 2012 Server x64(64 bit) and x86 (32 bit).
* IBM AIX
* Apple OS X, Mac OS X x64(64 bit) and x86 (32 bit).
* FreBSD x64(64 bit) and x86 (32 bit).
* HP UX

Source: http://www.mysql.com/support/supportedplatforms/database.html

**Thoughts**

Most of the software features was implemented according to the specification defined in the planning stage, except for the requirement on product discount. This was not implemented due to time limitation. However, this feature can be included in the next version of the prototype. Other features such as card payment and returns will also be incorporated into the development of the software in the future. The development stage of the project went really well, as there really was no unavoidable or unforeseen event that would have caused serious delays. I believe that I have done a good job in the research and preparation for this project.

# 5. Conclusion and Recommendations

Performance and security is an issue where the Raspberry Pi houses both the POS-Pi application and MySQL Server database server. The Raspberry Pi (Model B+) only has 512 MB memory capacity, and to run both applications on the same device will overtax the hardware resources. A major difference in response time is to be expected when comparing the software’s performance between the Raspberry Pi and a normal PC. I would recommend running the software on the newer Raspberry Pi 2 which has 1GB memory and an upgraded processor. See appendix on page 363 for a direct comparison between the Raspberry Pi B+ and the newer Raspberry Pi 2.

The Raspberry Pi also uses a micro SD card with 8GB capacity to store data. This is neither a reliable nor a safe medium to store sensitive and important business data. I would recommend moving the MySQL database server to a separate and dedicated server. This would considerably improve the performace of the POS-Pi application and provide a safer and more reliable way to store and access data

# 6. Bibliography

[1] Oracle Corporation, nd. Javadoc. [computer program]. Oracle Corporation. [Accessed April 28, 2015].

[2] Visual Paradigm International, nd. Visual Paradigm. [computer program]. Visual Paradigm International. Available at: <http://www.visual-paradigm.com/> [Accessed April 29, 2015].

[3] Jason E. Robbins, nd. ArgoUML. [computer program]. Tigris.org. Available at: <http://sourceforge.net/projects/argouml/> [Accessed April 29, 2015].

[4] Rapid Quality Systems, nd. Code Rocket for Eclipse, [computer program]. Rapid Quality Systems. Available at: <http://www.rapidqualitysystems.com/> [Accessed April 29, 2015]

[5] Olivier Müller, et al, 2001. phpMyAdmin, [computer program]. SourceForge.net. Available at: <http://www.phpmyadmin.net/> [Accessed May 04, 2014]

[6] Michael Kölling, 1999. BlueJ, [computer program]. University of Kent. Available at: <http://www.bluej.org/> [Accessed May 06, 2015]

[7] Apache Friends, nd. XAMPP Control Panel, [computer program]. Apache Friends. Available at: <https://www.apachefriends.org/download.html> [Accessed May 06, 2015]

[8] Scandit Inc, nd. Barcode Generator, [online tool], Scandit Inc. Available at: <http://www.scandit.com/barcode-generator/> [Accessed February 10, 2015]

[9] Eclipse Foundation, nd. Eclipse IDE Luna Service Release, [computer program]. Eclipse Foundation. Available at: <www.eclipse.org> [Accessed 2014]

[10] Jiri Sedlacek and Tomas Hurka, nd. VisualVM, [computer program]. Java.net. Available at: <https://visualvm.java.net/?VisualVM\_1.3.8> [Accessed May 26, 2015]

[11] ComputerWeekly [July 2009][website] Point of Sale (POS) Software – Essential Guide, http://www.computerweekly.com/feature/Point-of-Sale-POS-software-software-Essential-Guide [Accessed January2015]

[12] DigitalDining [no date][website] Handheld POS, http://www.digitaldining.com/handheld-pos/ [Accessed January 2015]

[13] Computer Science and Information Engineering [December 2006] [website] A Case Study – Point of Sale, http://www.csie.ntut.edu.tw/sdrc/files/course/20061201/SoftwareRequirementSpecification.pdf [Accessed January2015]

[14] Chip and Pin Machines [no date][website] WorldPayZinc vs iZettle vs Payleven, http://www.chipandpin-machines.com/category/chip-and-pin/ [Accessed January 2015]

[15] Raspberry Pi [no date][website] Raspberry Pi, http://www.raspberrypi.org/ [Accessed December 2014]

# 7. Appendix

Raspberry Pi Model B+ and Raspberry Pi 2 comparisson:

|  |  |  |
| --- | --- | --- |
| Spec | **Raspberry Pi Model B+** | **Raspberry Pi 2** |
| Processor | Single-core Broadcom BCM2835 ARMv6 (700 Mhz) | Quad-core Broadcom BCM2836 ARM v7 processor (900 Mhz) |
| Memory | 512 mb 350Mhz | 1GB 400Mhz |
| GPU | Videocore - 4 | Videocore - 4 |
| Snappy Ubuntu Core support | No | Yes |
| Future Windows 10 support | No | Yes |
| Price | £35 | £35 |