**Universal Point of Sale (POS)**

Ian Prince, Justin Hawkins, Michael Franklin, Austin Williamon

Course Requirement CS321

11/28/17

Copyright 2017 AUTHORS: Ian Prince, Justin Hawkins, Michael Franklin, Austin Williamon

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3, (3 November 2008) or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found at <https://www.gnu.org/licenses/fdl.html>.

**1.0 Project Description:**

Our Group is doing a cash register program whereby the user can customize and determine items for a preset cash register. Which during operation mode lists the items chosen for that transaction, calculates relevant tax, and presents a total to the user based on items selected. Users should have some control over the look of the cash register as well as the state tax rates used in calculations. User choices should be savable in some form or fashion.

This project consists of two applications, an editor to configure the template for the cash register, and an emulator for the user to interact with the register. The editor can customize the buttons and their corresponding parameters within a predetermined preset of buttons. They would also be able to change the name, price, font, and color of each button. Presets will be saved to an xml file, and the user would be able to edit a previous register.

The emulator would run use this preset to build a functioning register. During operation mode, the register would be able to list the items for a given transaction, calculate the total based on relevant tax, and calculate change based on an amount of received cash.

**2.0 Project Management:**

**History:**

Initially we started with a cash register simulation of sorts. As time went on this devolved into a customizable point of sale setup. Initially the user could customize every aspect of the register and emulate the function of the register. As more time passed various features were removed and processes streamlined. What was left was a simple yet still customizable interface where the user could customize various buttons with a name, price, stock quantity, and stock item photo. They will also be able to set the tax rate that corresponds with their region as well as font and color.

**Personnel:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Members** | **Educational Background** | **Relevant Experience** | **Project Roles** |
| Justin Hawkins | Bachelors of Science Business Administration  Bachelors of Science Computer Science (in progress) | CS221 Data Structures  CS 121 Programing C++  CS102 Intro to C  CS 103 Intro to Java  Retail Experience working with a POS | MVC implementation  Documentation |
| Ian Prince | Bachelors of Science Computer Science (in progress) | CS221 Data Structures  CS 121 Programing C++  CS102 Intro to C  Retail Experience working with a POS | Code Model  MVC implementation  XML Reader/Writer |
| Austin Williamon | Bachelors of Science Computer Science (in progress) | CS221 Data Structures  CS 121 Programing C++  CS102 Intro to C  Retail Experience working with a POS | Graphics and Animation  Text Formatting |
| Michael Franklin | Bachelors of Science Computer Science (in progress) | CS221 Data Structures  CS102 Intro to C  CS 121 Programing C++  Retail Experience working with a POS | GUI Design  GUI Programing and Implementation |

**Effort:**

|  |  |  |
| --- | --- | --- |
| **Meeting** | **Time** | **Running Time Total** |
| Project subject selected: Universal Cash Register | 2hrs | 2hrs |
| Outlined specific programmable features we want to include | 3hrs | 5hrs |
| Use cases and CRC cards made | 3.5hrs | 8.5hrs |
| Formal graphics for CRC and use cases | 2.7hrs | 11.2hrs |
| Google slides made for presentation 1 | 2hrs | 13.2hrs |
| Design and implementation for the model | 5hrs | 18.2hrs |
| Design and implementation for the view | 4.5hrs | 22.7hrs |
| GUI layout and design | 4hrs | 26.7hrs |
| Google slides made for presentation 2 | 3hrs | 29.7hrs |
| Overarching coding meeting | 6hrs | 35.7hrs |
| Overarching coding meeting | 3.5hrs | 39.2hrs |
| Coding for the controller | 5.0hrs | 44.2hrs |
| Completion of editor coding | 5.0hrs | 49.2hrs |
| Coding of emulator | 4.5hrs | 53.7hrs |
| Coding of emulator | 3.5hrs | 57.2hrs |
| Debugging and testing | 3.5hrs | 60.7hrs |
| Debugging and testing | 3hrs | 63.7hrs |

**3.0 Use Cases:**

Our goal is to provide a simple, customizable, and flexible point of sale that can be used for various types of business.

**Examples:** Grocery Stores, Restaurants, Specialty Retail, General Merchandise

This could be extremely useful for upcoming businesses that want full customization over their point of sale system. Additionally, as well as businesses who want to do a full re haul of their current point of sale systems.

**4.0 Requirements:**

**Functional Requirements:**

* The program is expected to allow the user to configure a custom POS and test it with an emulator.

**User Interface Requirements:**

* From a visual standpoint our program will display a template the user can edit and provide the inputs shown below.
* User will input a string for the item name as well as a price and quantity. They will also set the tax rate that corresponds with their region. If the user inputs an invalid entry a dialogue box will pop up saying invalid entry and ask them to reenter specifically a string, real number or integer.
* The users template selections will be written to an XML file which will be loaded from later when using the emulator. Users can choose from a multitude of created XML files to load the specific one for emulation.

**Future Modification and Extension:**

For future modifications we will allow a more ground up personalization and customization of our program. Specifically, the ability to add button objects to the register as well as add the users own image instead of choosing from a preselected set of images. Additionally, we would like to add another trackable attribute to the buttons that would track individual sales for the item. Another feature would be the ability of the user to apply discounts or sales to specific items to overall sales like a buy two get one free.

**Summary:**

|  |  |
| --- | --- |
| **Requirement** | **Satisfies Requirement** |
| Allow users to edit their custom template through editor gui | Satisfied by:   * Editor\_Visual\_2.java * Color\_Text\_Dialog.java * Edit\_Button\_Dialog.java |
| Reflect users selections in editor model | Satisfied by:   * Button.java * ButtonBeanInfo.java * TemplateBeanInfo.java |
| Update editor model view to reflect changes | Satisfied by:   * UserInfoBus.java |
| Save user selections in XML format | Satisfied by:   * XMLManager.java |
| Loads user selections for emulation from XML file | Satisfied by:   * XMLManager.java |
| Reflect user selections in emulator model | Satisfied by:   * XMLManager.java * Template.java |
| Update emulator model view to reflect changes | Satisfied by:   * Emulator\_Visual.java (using observer pattern) |
| Allow users to interact with POS emulator | Satisfied by:   * Emulator\_Visual.java * DataHolder.java (holds button values) |

**Associated Tests:**

|  |  |
| --- | --- |
| **Requirement** | **Test** |
| Allow users to edit their custom template through editor gui |  |
| Reflect users selections in editor model |  |
| Update editor model view to reflect changes |  |
| Save user selections in XML format |  |
| Loads user selections for emulation from XML file |  |
| Reflect user selections in emulator model |  |
| Update emulator model view to reflect changes |  |
| Allow users to interact with POS emulator |  |

**5.0 Design:**

**CRC Cards:**

**Package: Editor\_Gui**

|  |  |
| --- | --- |
| **Color\_Text\_Dialog** | |
| * Visual representation of color and font selection portion of the gui |  |

|  |  |
| --- | --- |
| **Edit\_Button\_Dialog** | |
| * Visual representation of the button edit portion of the gui |  |

|  |  |
| --- | --- |
| **Editor\_Visual\_2** | |
| * Visual representation of the base template editor |  |

|  |  |
| --- | --- |
| **ImagePanel** | |
| * Necessary for editor package to function |  |

|  |  |
| --- | --- |
| **UserInfoBus** | |
| * Acts as a go between or controller for the editor view and editor model |  |

|  |  |
| --- | --- |
| **XMLManager** | |
| * Writes data to be saved from template | * **Template** |

**Package: Editor\_Model**

|  |  |
| --- | --- |
| **Button** | |
| * Set price * Set name | * **Template** |

|  |  |
| --- | --- |
| **ButtonBeanInfo** | |
| * Grabs info from user selections in editor gui | * **TemplateBeanInfo** |

|  |  |
| --- | --- |
| **Template** | |
| * Store button values * Change tax rate | * **Button** |

|  |  |
| --- | --- |
| **TemplateBeanInfo** | |
| * Using observer pattern updates model based on user selections |  |

|  |  |
| --- | --- |
| **XMLManager** | |
| * Writes data to be saved from template | * **Template** |

**Package: Emulator\_Controller**

|  |  |
| --- | --- |
| **Controller** | |
|  |  |

**Package: Emulator\_Gui**

|  |  |
| --- | --- |
| **Emulator\_Visual** | |
| * Displays the visuals for the emulator |  |

|  |  |
| --- | --- |
| **Template\_Emulator** | |
|  |  |

**Package: Emulator\_Model**

|  |  |
| --- | --- |
| **Bank** | |
| * Receives starting value * Adds money from subsequent transactions * Reports current value |  |

|  |  |
| --- | --- |
| **Button** | |
| * Adds item to the dataholder | * **Dataholder** |

|  |  |
| --- | --- |
| **Calculator** | |
| * Keeps track of subtotal of items added to transaction * Applies tax and displays total with sales tax | * **Dataholder** |

|  |  |
| --- | --- |
| **DataHolder** | |
| * Record an item to be processed when button is pressed | * **Button** |

|  |  |
| --- | --- |
| **Template** | |
| * Contains data to be fed into XMLManager | * **XMLManager** |

|  |  |
| --- | --- |
| **XMLManager** | |
| * Reads data in to be emulated | * **Template** |

**Model Design:**

Our program implements two separate models. We have an editor model and an emulator model. The model for the editor contains the Button code, Template, and the XMLManager. When the user updates the view by selecting their preferences on the Template the model will update. The variables that will change will be button variables of price, stock quantity, and name. The XMLManager will save the user template choices in XML format and load the template for emulation.

**View Design:**

Our view begins with a blank template. As the user makes decisions/choices and this information is passed and stored into the model. Using the observer pattern the view is then updated in turn to display these changes.

**Control Design:**

We chose to implement an observer pattern to monitor and update between the model and view for the editor and the emulator.

**Communication Design:**

Thebasic idea of communication between elements is passing a reference to what needs to be accessed to the accessing element.

**6.0 Implementation:**

**Packages and Classes:**

* **Editor\_Gui Package**
  + Color\_Text\_Dialog.java
  + Edit\_Button\_Dialog.java
  + Editor\_Visual\_2.java
  + ImagePanel.java
  + Template\_Editor.java
  + UserInfoBus.java
  + XMLManager.java
* **Editor\_Model Package**
  + Button.java
  + ButtonBeanInfo.java
  + ImagePanel.java
  + Template.java
  + TemplateBeanInfo.java
  + XmlManager.java
* **Emulator\_Controller Package**
  + Controller.java
* **Emulator\_Gui Package**
  + Emulator\_visual.java
  + Template\_Emulator.java
* **Emulator\_Model Package**
  + Bank.java
  + Button.java
  + Calculator.java
  + DataHolder.java
  + Template.java
  + XMLManager.java

**Utility Classes and Packages:**

**Test Plan:**

**Tested Functionality:**

**Untested Functionality:**

**7.0 Discussion:**