**L COETZEE (LUKE)**

**23676469**

**J SITTIG (JACO)**

**27483533**

**ITRW 212**

**Bread & Butter**

**Phase II: Proposal with Inheritance and Methods**

**Lecturer: Me M Zeeman**

**15 March 2017**

Table of Contents

[1. The objective of Bread & Butter 1](#_Toc477266175)

[2. Boundaries 1](#_Toc477266176)

[3. Specifications 1](#_Toc477266177)

[4. Flow of events 3](#_Toc477266178)

[5. Context Diagram 4](#_Toc477266179)

[6. Inheritance Diagram 5](#_Toc477266180)

[7. Classes and Methods 6](#_Toc477266181)

# The objective of Bread & Butter

The focus of our project is to assist any school owned tuck shop (*snoepie*) in handling the day-to-day of running the own shop. The objective is to make the personnel’s life easier, and more organized.

Bread & Butter is not just a point-of-sale system, but an automated ordering and stock taking program. The automated ordering of stock, will help eliminate the chances of being out of stock of a specific product, which in turn will help to increase sales profit and overall customer satisfaction.

This system will consist of three main pages: POS, Stock and Manager administration.

# Boundaries

* Intended for school situated tuck shops only.
* Desktop only – no mobile application or access.
* The aim is not to be a financial application, but to help keep track of day-to-day income and expenditures.
* Personnel will still need to enter stock manually, not a fully automated system.
* Periodically manual stock taking will be required, to identify missing stock through theft.
* Either Windows or Linux is needed to run the application.
* Input is limited to a keyboard and/or a mouse. Touchscreen or voice control will not be supported.
* Employees have access to only the POS, while approved users (such as managers) have access to the whole system.

# Specifications

* Managers can register new users, and choose what access they’ll have.
* Registered users will log in with only their employee code. Managers log in with employee code and password.
* The application will consist of a simple interface.
* The interface will feature dropdown interface, arranged into different categories.
* Users can view current stock, add new stock or delete and update current stock.
* Low stock will trigger an automated stock ordering method, to notify the manager next time they log in.
* Current supplier details can be changed, or new suppliers can be added.
* Simplified POS interface with dropdown categories.
* The option between cash- and account sales is available.
* All student account information will be stored in a text file.
* Manager log in required for financial access.
* Autogenerated income and expenditure statements.

# Flow of events



# Context Diagram

**Text files**

**Textbox**

Read finance reports

Add & update student records

Generate message for new orders

Retrieve information

Interact with POS

**Text files**

**Users**

Create text files for log in details and employee codes

Stock admin

# C:\Users\lodew\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Inheritance.jpgInheritance Diagram

# Classes and Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Super Class | 1. Info | | |
| Sub Classes | * 1. POSInfo | * 1. AdminInfo | * 1. StockInfo |
| Sub Classes | * + 1. AccountSales     2. CashSales | * + 1. Finance     2. Employee     3. Student | 1.3.1. Add/Remove  1.3.2. OrderNew |

## Info

* Private instance fields:
  + String username
  + String password
  + String ID
* Methods
  + Get- and set methods for all instance fields

## POSInfo

* Private instance fields:
  + ArrayList <Stock> arrStock
* Methods
  + **determineStock(arrStock)**

# AccountSales

* Private instance fields:
* ArrayList <Student> arrStudent
* String name
* String accNumber
* ArrayList <Finance> arrStudentAcc
* Array [ ] arrTotalSales
* Array [ ] arrAccSales
* Methods:
* **addNewAcc(arrStudent)**
* This method will add a new student account in to the ArrayList called arrStudent, the ArrayList will then add this new account into a text file called StudentAcc.txt.
* **addDeposit(String name, int accNum, arrStudentAcc, arrTotalSales)**
* The addDeposit method will find the student who wants to add money to his/her account in the ArrayList arrStudent and will update the balance of the student’s account, this will also update the text file StudentAcc.
* **checkBalance(String name, int accNum, arrStudentAcc, arrTotalSales)**
* This method will be called when a student wants to see how much money he/she still have left in their account. It will call the ArrayList arrStudent to find the student’s balance.
* **calcAccSale(arrStudentAcc)**
* calcAccSale will add all the account sales into an array called arrStudentAcc.

# CashSales

* Private instance fields:
* ArrayList Stock [ ] arrStock
* Array arrCashSales
* Methods:
  + - **calcCashSales(arrTotalSales)**
      * This method will add all the cash sales into an array called arrCashSales.

# adminInfo

* Private instance fields:
  + - String username
    - String password
* Methods:
  + - **checkLogIn(String username, String password)**
* The method will check if the user’s login info is correct, if it is incorrect a error message will display or if the login info is correct we user will have access.

# Finance

* Private instance fields:
  + Array [ ] arrTotalSales
  + Array [ ] arrCashSales
  + Array [ ] arrAccSales
* Methods:
  + **calcTotalAccSales()**
    - This will call the method calcAccSale from the AccountSales class to store all the sales into the array arrAccSales.
  + **calcTotalCashSales()**
    - This will call the method calcCashSales from the AccountSales class to store all the sales into the array arrAccSales.
  + **calcTotalSales()**
    - It will add the two above methods together into an array arrTotalSales

# Employee

* Private instance field:
  + String nameID
  + String username
  + String password
  + Array [ ] arrLogIn
* Methods:
  + **createEmpLogIn()**
    - The method will be used by the manager to create a login and password for a employee. It will use an array called arrLogin to store the name, ID, username and password.

# Student

* Private instance field:
  + ArrayList Student [ ] arrStudent
* Method:
  + **toString()**
    - A normal toString method will be used to display all the student account information very neatly.

# StockInfo

* Private instance field:
  + ArrayList <Stock> arrStock
* Methods:
  + Get- and set methods for all instance fields

# AddRemove

* Private instance field:
  + String category
  + String name
* Methods:
  + **addStock(arrStock, String category, String name)**
    - This method will use the category and name of a product to determine where to add the new stock in the ArrayList arrStock.
  + **removeStock(arrStock, String category, String name)**
    - The method will find the product by using the name and category as criteria to remove it from the arrStock.

# OrderNew

* Private instance field:
  + String category
  + String message
* Methods:
  + **determineOrder(arrStock)**
    - determineOrder will use the method determineLow() to make a order list for the manager to order new stock from the suppliers.
  + **determineLow(arrStock)**
    - This method will check every amount of stock in the array to see wich product amount is low.