DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

RETAIL MANAGEMENT SYSTEM

by

Berhan Türkü Ay - 2017510013

Gökhan Göksel Elpeze - 2015510127

Murat Taylan Şahin - 2017510093

Oktay Türkdağlı - 2017510101

CHAPTER ONE INTRODUCTION

Increasing productivity and speeding processes up has a direct effect on customer satisfaction in retailers. Technology plays an important role in terms of more comfortable and intuitive customer experience. Retail Management System (RMS) we develop keeps this principle in mind and lets you manage these processes in real-time. Thanks to this, it is possible to speed up processes easily ranging from receipts to products, inventory management to price management, purchases to promotions, orders to transfers.

RMS is designed with needs of retailers selling a wide range of different products in mind.

Using RMS, you can attract more customers and attend to existing customers in real-time, manage different branches of your shop, analyze your sale data and generate meaningful reports about your company's future. In addition to this you can offer promotions for customers and check your employees' performance from this program.

MAIN FUNCTIONALITIES AND USER GUIDE:

Selling a Product:

- After Logging in with your username and password, click on SALES menu.
- In the screen that opens up, enter product id to the SEARCH bar. Click on the correct product or press ENTER.
- Chosen product will appear on the Cart Preview Window.
- Repeat these steps for all other products in your customer's cart.
- Click on GENERATE RECEIPT button.
- Pick the correct payment option (Cash or Credit Card) and press COMPLETE TRANSACTION button.
- You can press CANCEL on any of these steps in case you want to cancel the transaction.
- Sale successful notification appears upon successful sale.

Adding a New Product:

- After login, if you have admin privileges PRODUCT tab will activate, click on PRODUCT menu.
- If you want to add a new item, leave SEARCH bar on the newly opened screen blank.
- Product's attributes are filled (Example TV attributes):
    - Category
    - Name
    - Size
    - Build Date
    - Name of Manufacturer
    - Name of Distributor
    - Price
    - Quantity
    - Date Product enters company (Passive)
    - Person adding the item (Passive)
- Click on SAVE.

- Item added successfully notification appears upon successful entry.

Removing a Product:

- After login, if you have admin privileges PRODUCT tab will activate, click on PRODUCT menu.
- Type id of the product that needs to be removed on the newly opened screen. Click on the correct item or press ENTER
- Attributes of the product should fill automatically. Click on REMOVE button
- Click YES on the “Are you sure?” popup.
- Item successfully removed notification appears upon successful removal.

Editing an Existing Product:

- After login, if you have admin privileges PRODUCT tab will activate, click on PRODUCT menu.
- Type id of the product that needs to be edited on the newly opened screen. Click on the correct item or press ENTER
- Attributes of the product should fill automatically. Modify the attributes you need to change.
- Click on EDIT button.
- If the edited product is the same as an already existing product, “Would you like to merge these items?” notification appears. If you do not want to merge the items, click no and edit items attributes again.
- Item successfully edited notification appears upon a successful edit..

Transfering Products:

- After login, if you have admin privileges PRODUCT tab will activate, click on PRODUCT menu.
- Pick TRANSFERS tab on the newly opened screen, Type item name or id to the SEARCH bar and pick the correct item or press ENTER
- Pick the origin and destination branches.
- Enter the quantity (Quantity may not exceed origin branch's maximum stock).
- "Transfer request is sent." notification appears upon successful entry.
- An admin should approve of transfer request.
    - If user has logged in with an admin account, MANAGEMENT tab will activate, click on MANAGEMENT tab.
    - Pick TRANSFER REQUESTS tab from newly opened window.
    - Transfer requests should be listed here. Pick the request you want to approve and click APPROVE.
    - Transfer approved notification appears upon successful approval.

Adding a New Employee:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.
- Navigate to EMPLOYEES tab on the newly opened window and pick ADD option. Do not fill in the SEARCH bar.
- Following attributes of the Employee is filled
    - Employee ID (Passive)
    - National Identification Number
    - Taxpayer ID
    - Name
    - Surname
    - Branch id where employee works in
    - Position

    - Phone Number
    - Address
    - E-mail
    - Salary (If this is left empty base salary of the position will be applied)
    - Birth Date
    - Birthplace
    - Start date(Passive)
    - Current user (Passive)
- Click on SAVE button.
- Employee added notification appears upon successful entry.

Editing Employees:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.
- Navigate to EMPLOYEES tab on the newly opened window and pick ADD option. Fill in the SEARCH bar with employee id or name and pick the employee or press ENTER.
- Employee attributes should automatically fill up, edit the attribute you want to change.
- Click EDIT button.
- Edit successful notification appears upon successful edit.

Removing Employees:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.

- Navigate to EMPLOYEES tab on the newly opened window and pick ADD option. Fill in the SEARCH bar with employee id or name and pick the employee or press ENTER.
- Employee attributes should automatically fill up, check if employee is the one you want to remove, then Click EDIT button.
- Answer yes to “Are you sure?” notification.
- Employee successfully removed notification appears upon successful removal.

Adding a new branch:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.
- Navigate to BRANCHES tab on the newly opened window and pick ADD option. Do not fill in the search bar.
- Fill in the following attributes of the new branch:
    - Name
    - Phone number
    - E-mail
    - Address
    - Opening Date (Passive)
- Click on SAVE button
- Successfully saved notification appear upon successful entry.

Removing a Branch:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.
- Navigate to BRANCHES tab on the newly opened window and pick REMOVE option. Fill the search bar with branch id.
- Click on REMOVE button
- Answer yes to “Are you sure?” notification.
- Branch successfully removed notification appears upon successful removal.

Editing a Branch:

- After login, if you have admin privileges MANAGEMENT tab will activate, click on MANAGEMENT menu.
- Navigate to BRANCHES tab on the newly opened window. Fill the search bar with branch id.
- Attributes should fill automatically. Edit the attributes you want to change.
- Answer yes to "Are you sure?" notification.
- Branch successfully edited notification appears upon successful edit.

CHAPTER TWO: REQUIREMENTS

CLASSES

A) MANAGEMENT CLASSES

```
                        Inventory Manager

 Adds an already existing item to mentioned branch
  addItemToShop(Shop shopToAdd, int quantity, Item item): void

 Removes an already existing item from a branch in specified quantity
  removeItemFromShop(Shop shopToAdd, int quantity, Item item): void

 Creates a new Item
  addNewItem(Item item): void

 Removes an existing item from all branches and marks it as removed
  removeItem(Item item): void

 Transfers specified product from origin Shop to destination
  transferBetweenBranches(Shop originShop, Shop destinationShop, Item
transferredItem, int quantity): void

 Orders items in specified quantity from the item's manufacturer
  orderItem(Item item, int quantity): void

 Searches and returns an item from Inventory
  searchItem(Item item): Item
```

```
                              Sales Manager

Sells specified item in given quantity
sellItem(Item item, int quantity): void

Generates final receipt of customer and saves it
generateReceipt(Item[] items): void

Finds and returns specified receipt
searchReceipt(int receiptID, Date date): Receipt

Returns an item and saves a return receipt
returnItem(Item item, int receiptID): void
```

```
                             Reports Manager

Fetches specified report
displayReports(int chosenReport) :void
.
.
.
```

```
                              Staff Manager

Adds a new employee
addEmployee(Employee employee) :void

Removes an employee
removeEmployee(Employee employee) :void

Edits an employee
editEmployee(Employee employee) :void

Finds and returns employee with specified ID
searchEmployee(int employeeID) :Employee
```

B) PRODUCT CLASSES

Note: Attributes of Product classes are not final and they might change later on to be more detailed/realistic or to improve usability.

Our base class for products is "Item"

```
public class Item {
    private String brand;
    private String name;
    private double price;
    private int itemID;
    private Manufacturer manufacturer;
```

We are extending new classes from Item class. First one is "Clothing".

1 - CLOTHING

```
public class Clothing extends Item{
    private String color;
    private String gender;
```

We are extending new classes from Clothing class. They are "Footwear", "Pants" and "Tshirt".

SUBCLASSES:

1.1 FOOTWEAR

```
public class Footwear extends Clothing {
    private double size;
```

1.2 PANTS

```
public class Pants extends Clothing {
    private int sizeWidth;
    private int sizeLength;
```

1.3 T-SHIRT

```
public class Tshirt extends  Clothing {
    private String size;
```

We are extending new classes from Item class. Second one is “Electronics”.

2 - ELECTRONICS

```
public class Electronics extends Item {
    private double size;
    private String resolution;
```

We are extending new classes from Clothing class. They are “Computer”, “Phone” and “Television”.

2.1 COMPUTER

```
public class Computer extends Electronics {
  private String processor;
  private String graphicsCard;
  private String graphicsCapacity;
  private String ramType;
  private String ramSpeed;
  private String ramCapacity;
  private String storage;
```

2.2 PHONE

```
public class Phone extends Electronics{
    private String memory;
    private String ram;
```

2.3 TELEVISION

```
public class Television extends Electronics {
    private boolean smart;
    private String screenType;
```

We are extending new classes from Item class. Third one is “OtherProducts”. Other products is for all the items and categories that don’t have a specific class.

3-) OTHER PRODUCTS

```
public class OtherProducts extends Item {
    private String categoryName;
```

C) COMPANY CLASSES:

MAIN CLASS: COMPANY

1 – COMPANY

```
public class Company {
  private List<Shop> shops;
  private int phoneNumber;
  private String emailAddress;
  private Address address;


addNewShop(Shop shop):void

removeShop(Shop shop):void

addNewManufacturer(Manufacturer manufacturer):void

removeManufacturer(Manufacturer manufacturer):void
```

2- SHOP

```
public class Shop {
    private List<Employee> employees;
    private ArrayList<Item> products;
    private int phoneNumber;
    private String emailAddress;
    private Address address;
```

3- EMPLOYEE

```
public class Employee {

    private String name;
    private Address address;
    private int phoneNumber;
    private int weeklyWorkHours;
    private Position position;
    private int shopID;
    private int employeeID;
    private String password;
    private int weeklyWageBonus;
```

4- MANUFACTURER

```
public class Manufacturer {
    private String name;
    private int manufacturerID;
    private Address address;
    private String phoneNumber;
    private List<Item> soldItemList;
```

UTILITY CLASSES:

1-ADDRESS

```
public class Address {
    private String city;
    private String town;
    private String street;
```

2-DATE

```
public class Date {
    private int day;
    private int month;
    private int year;

validateDate(Date date) :boolean
```

3- POSITION

```
public class Position {
    private String positionName;
    private int baseWage;
```

4- RECEIPT

```
public class Receipt {
    private Date date;
    private double totalPrice;
    private Employee cashier;
    private List<Item> cart;
    private int receiptID;
```

5- PHONE NUMBER

```
public class PhoneNumber {
   private String countryCode;
   private int code;
   private int number;
   private String type;

validatePhoneNumber(PhoneNumber number):boolean
```

GUI CLASSES:

CONTROLLERS:

(Common Function)

```
public void handleButtonAction(ActionEvent event) {}:void
```

1– CompanyScreenController

2- HomeScreenController

```
This function logs out from current user and opens Login Screen
logout():void

This function saves all data to a file for next session
saveData():void
```

3-LoginScreenController

```
Checks username and password
loginCheck() :boolean

Loads data from previous sessions to initialize the program
loadData():void
```

4-SaleScreenController

5-ReportsScreenController

FXMLs:

1-companyScreen

2-homeScreen

3-loginScreen

4-saleScreen

5-reportsScreen

CHAPTER THREE

UML DIAGRAMS

In this chapter, the solutions given in the previous chapter must be supported by UML diagrams: Use Case Diagrams, Class Diagrams, Activity Diagrams, Sequence Diagrams and State Diagrams. All diagrams must be explained in at least one paragraph.

CHAPTER FOUR

IMPLEMENTATION

In this chapter, all interfaces and background codes should be explained in detail.

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

In this chapter, a summary and future works must be written.