**Project 3**

|  |  |
| --- | --- |
| Specifications | Test Case |
| The starting Inventory shall be read from the supplied CSV file, processed, and stored(product ID, and total units). | Test Case 1: The starting inventory will be output with the inventory output, and can be verified against the CSV file data proving that the data was read, and processed. |
| The Inventory System shall read Sales Data (Customer ID, Product ID, Units Sold) from the supplied CSV file. | Test Case 1: The sales totals will be processed and output with the inventory output, and can be verified against the CSV file data, proving that the sales data was accurately read and processed. |
| The Products sold shall be deducted from the current Inventory. | Test Case 1: The inventory output will output a calculated total of units remaining, which will demonstrate that the units that were sold have been deducted from the starting total. This will be visually verified at the final output. |
| The Inventory System shall output the  **Sales Data:**  A listing of all Products sold, the number of units sold for each Product, the total revenue received for each Product and the total revenue from the sales of all Products. | Test Case 1: The program will output the products sold, the number of units sold of each, the total for each product, and a total for all sales. This will be visually verified at the final output. |
| The Inventory System shall output the **Inventory:** A listing of the change in Inventory: starting number of units for each Product, units sold and the final count of units in the Inventory. | Test Case 1: The program will output the starting number of units, the units sold, and the units remaining in the program output. This will be visually verified at the final output. |

|  |
| --- |
| Main |
| -inventoryCSV: String  -productData: String  -salesCSV: String  -inventory: LinkedList<Product>  -sales: LinkedList<salesReceipt>  -df: DecimalFormat |
| +inventory(inventoryFileName: String, productFileName: String): LinkedList<Product>  +receipts(filename: String, currentInventory: LinkedList<Product>):LinkedList<salesRecipet> |

**Project 3**

**UML DIAGRAM**

|  |
| --- |
| salesReceipt |
| -saleID: Int  -customerID: Int  -saledateTime: Date  -items: Linkedlist <saleItem>  -total: Double  -df: DecimalFormat |
| +salesReceipt(cID: Int, sID: Int)  +setSaleID(sID: Int)  +getSaleID(): Int  +setCustomerID(cID: Int)  +getCustomerID(): Int  +setSaleDateTime()  +getDate():Date  +addItem(id: int, u: int, currentInventory: LinkedList<product>)  +getItem(a: int): saleItem  +calculate()  +printSalesReceipt() |

|  |
| --- |
| saleItem |
| -Productid: Int  -units: Int  -productDescription: String  -unitPrice: Double  -lineTotal: Double  -df: DecimalFormat |
| +saleItem(id: int, u: int, currentInventory: LinkedList<product>)  +setProductID(id: int)  +getProductID(): Int  +setUnits(units: Int)  +getUnits() Int  +setUnitPrice(p: Double)  +getUnitPrice(): Double  +getLineTotal(): Double  +printSaleItem() |

|  |
| --- |
| Product |
| -productID: Int  -description: String  -price: Int  -df: DecimalFormat  -unitsInInventory: int |
| +product(id: int, d: String, p: Double)  +setDescription(d: String)  +getDescription(): String  +setProductID(id: Int)  +getProductID(): Int  +setPrice(p: Double)  +getPrice(): Int  +setUnitsInInventory(u: int)  +getUnitsInInventory(): int  +printdetails() |

**Project 3 PSUEDO CODE:**

//populate a linked list of products with the inventory method processing the inventoryCSV and productData. These are fed into the method by the file name reference variables.

//Read the product data with a buffered reader to store the variable information it contains, into a linked list of product objects. The reader reads one line, stores the product ID int, the Item description string, and the price double, and repeats this until there are no further lines.

//Read the inventory data into a buffered reader. Use the inventory data from the reader’s product ID data to match the input to the product in the linked list product list with the corresponding product ID, and then update the unitsInInventory int with the readers units data.

//Create a copy of the linked list of products, and use this to track the current inventory.

//populate a linked list of salesReceipt with the receipts method, by giving the method the file reference variable, and the current product inventory linked list.

//Use a buffered reader to process the data to the salesReceipts linked list, and matching data from the product inventory, to fill the fields for the salesReceipts(sale ID, Customer ID, Date, units, price, product description, line total, and total).

//Create a copy of the current inventory, by storing the inventory as a receipt at the list head.

//Prime the salesReceipts LinkedList by creating a blank total sales receipt in registry 1.

//Set the date

//Use a buffered reader to read and add the data line by line, as a salesItem until the customer ID changes.

//Add each new line storing the sales data productid, customer ID and units to the linked list of salesItems.

//Use the productid to scan the productinventory linked list to find the products price, and description, then store it in the saleItem variables.

//If the product already exists, add the units to the existing line item.

//Remove the number of units sold in the current line item from the instances product inventory linked list from the matching product by searching the productIDs, and reducing the stored variable unitsInInventory.

//Update the stored inventory image attached to the receipt.

//When the customer ID changes, start a new sales receipt.

//set the date

//Use a buffered reader to read and add the data line by line, as a salesItem until the customer ID changes.

//Add each new line storing the sales data productid, customer ID and units to the linked list of salesItems.

//Use the productid to scan the productinventory linked list to find the products price, and description, then store it in the saleItem variables.

//If the product already exists, add the units to the existing line item.

//Remove the number of units sold in the current line item from the current inventory linked list by matching product by searching the productIDs, and reducing the stored variable unitsInInventory from the inventory linked list.

//Store the ending inventory data at the end of the linked list.

//Create a loop to add another sales receipt for total sales at the end of the linked list, that uses the value from the start inventory sales receipt and deducts the remaining sales inventory, to calculate the difference, providing the total sales.

//Return the saleReceipt linked list, and the list head will be the total sales, with each receipt stored at the registry matching the sale ID.

//Print the inventory data, referencing the start inventory, the end inventory, the total sales for each product by referencing the start inventory receipt, end inventory receipt, and total sales receipt.