**Project 3**

|  |  |
| --- | --- |
| Specifications | Test Case |

|  |
| --- |
| 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 transaction type database

//While data, read store, store as array

//populate the queue

//While customer equals the last new customer, keep adding the times for transactions

//when reader reads a null new line end loop

//add last event to the event queue

//start the time counter

//when the time counter matches the peeked first arrival time, pop customer from event Q to the bank Q, storing the wait time variable to the event data

// Create departure event, and insert into the event queue in the proper location

//When counter hits a departure event, dequeue departure, enqueue bank queue customer to the served customer queue, dequeue bank queue, enqueue new customer to bank queue from event queue, dequeue that customer from the event queue, insert queue for new departure event

//When both queues are empty

//start customer count

//start wait time sum

//while queue not empty

//add customer wait time

//increase customer count

//print customer

//dequeue customer