Shipping Store UML Design

Zachary Golla, Kentessa Fanfair

Use Cases

(Use Case Free-Text)

Use Case 1: “Employee Managing Packages in Shipping Store Inventory”

1. Employee Log in **include** (Authenticate Employee)
2. System displays menu options
3. Employee searches system for package via tracking **include** (Identify Package)
4. Employee enters current location of package **include** (Update Package Information)
5. System removes delivered packages from inventory **include** (Identify Delivery Status)
6. System displays all remaining packages in inventory **extend** (Display Manage Inventory Menu Options)

Use Case 2: “Handling Customer Transaction”

Basic Flow:

1. User records package details **include** (Identify User)
2. System validates package details given by user **include** (Return Invalid Fields)
3. System determines cost of shipping
4. System determines date of shipping transaction
5. System determines date of delivery
6. System determines package deliverer
7. System updates transaction status
8. System displays package details
9. System saves package order to inventory **include** (Identify Delivery Status)
10. **extend** User prints shipping label

Alternative Flows:

[Invalid Order Entry]

After step 1, when system determines package details are invalid,

Repeat steps 1 to 2 until system determines details are valid.

Resume at step 3, to determine cost of shipping

[Multiple Packages to Same Location]

At step 3, System calculates shipping cost based on each individual of items’, volume, and weight

[Order Created by User On Website]

At step 1, when Actor is recording packages details,

if Actor recording details is a Customer,

1. System defaults package status to 'Pending' until package is dropped off
2. System defaults cost of shipping, employee id, and date of delivery to 'Null' until package is dropped off

Resume at step 2 when package is dropped off

[Drop off Order]

At step 1, if order exists in the inventory already, update relevant fields,

if package is for drop off,

1. Update order transaction status
2. Record employee who collected the package

Resume at step 3

Use Case 3: "Administrator Managing Shipping Store Accounts"

1. Administrator logs in **include** (Authenticate Employee)
2. System displays menu options
3. System displays net income from completed shipping transaction sales **include** (Calculate Total Shipping Transaction Sales)
4. System displays all expenses accrued from employee salaries and space rental **include** (Calculate Total Expenses, Identify Expenses Paid)
5. System displays statements of cash flow and invoices and balance sheets **include** (Identify Expenses Paid, Identify Total Shipping Transaction Sales, Net Profit)
6. **extend** (Display Menu Options, Administrator prints financial documents available)

Use Case 4: "Employee Using User Management System"

1. Employee logs in **include** (Authenticate Employee)
2. System determines user-access-privileges based on employee level **include** (Identify Employee Authorization Level)
3. System displays menu options **include** (Identify Menu Options Per Employee Authorization Level)
4. System displays list of existing users **include** (Identify Employee Authorization Level)
5. Employee searches for user via user id **include** (Identify Employee Authorization Level)
6. Employee creates new user account **include** (Identify User Type)
7. Employee updates user account information (Identify Employee Authorization Level)
8. System saves changes to user account information **extend** (Display Menu Options)

Alternative Flow:

[Collecting User Specific Information]

When new user is being created, Employee specify the type of user the account is being created for, if the new user is a Customer,

1. In addition to recording basic information, Employee records address

if the new user is an Employee,

1. In addition to recording basic information, Employee records SSN, Monthly Salary, Bank Account Number, Member Status, and PIN Number
2. Administrator sets basic employee system access rights i.e. create-shipping-transactions-access-rights, manage-inventory-system-access-rights, create-update-customer-access-rights, reply-to-customers-messages-access-rights

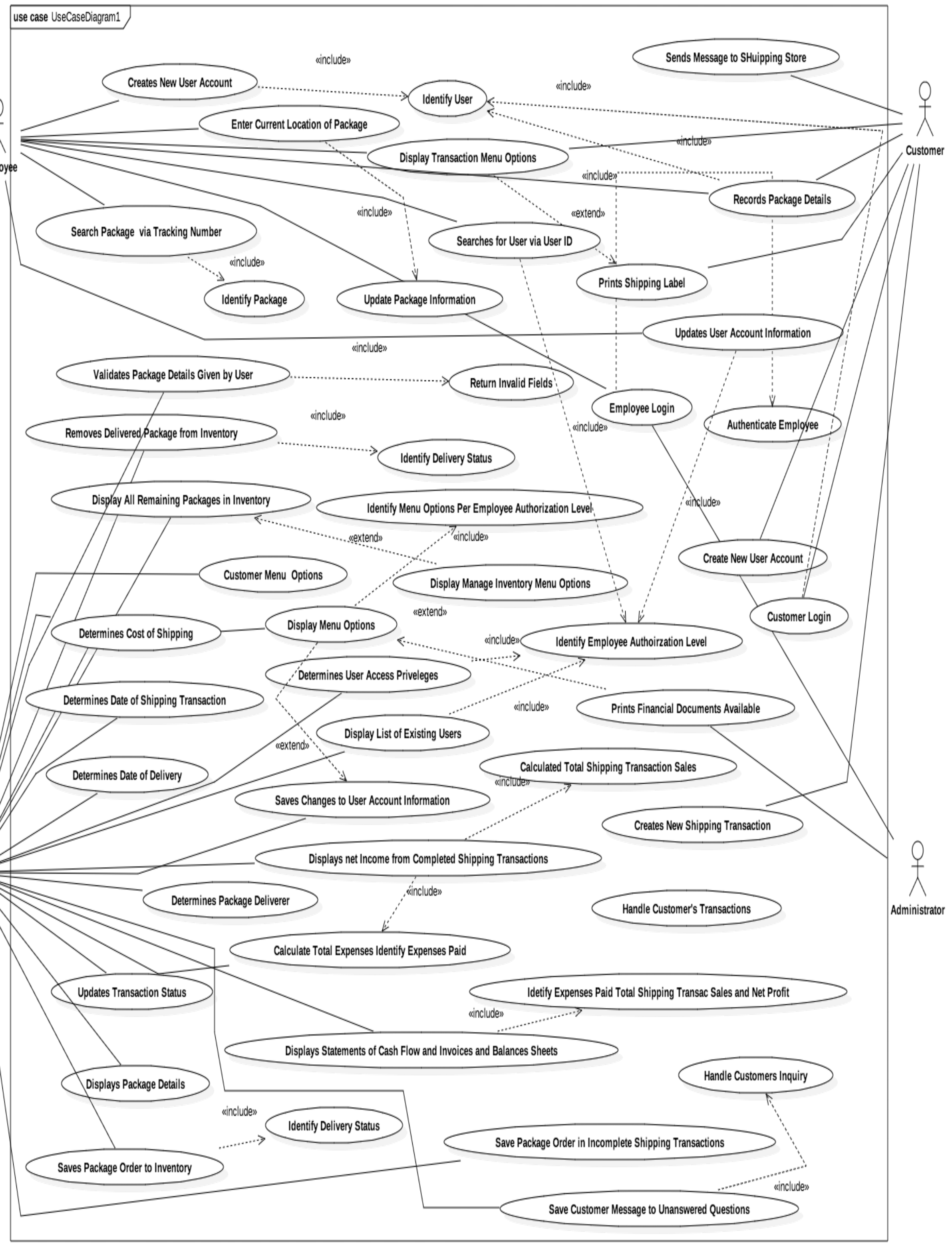
if the new user is an Administrator,

1. In addition to recording basic user information and employee information, Administrator sets additional system access rights. i.e. employee-management-system-access-rights, accounting-management-system-access-rights

Use Case 5: "Customer Using Shipping Store Website"

1. Non-Existing customer create new account
   Existing customer logs in with established credentials **include** (Identify User)
2. System displays customer menu options
3. Customer creates new shipping transactions **include** (Handle Customer's Transactions
4. System saves package order in 'Incomplete Shipping Transactions'
5. System displays package details **include** (Identify Delivery Status)
6. **extend** Customer prints receipt/shipping label
7. **extend** Customer sends message to shipping store via web form
8. **extend** System saves customer message to pool of unanswered questions **include** (Handle Customer's Inquiry)
9. **extend** Return to Main Portal

(Use Case Diagrams)


use case UseCaseDiagram1
Employee
Customer
System
Administrator
Creates New User Account
Identify User
Sends Message to SHuipping Store
Enter Current Location of Package
Display Transaction Menu Options
Records Package Details
Search Package via Tracking Number
Searches for User via User ID
Prints Shipping Label
Identify Package
Update Package Information
Updates User Account Information
Validates Package Details Given by User
Return Invalid Fields
Employee Login
Removes Delivered Package from Inventory
Identify Delivery Status
Authenticate Employee
Display All Remaining Packages in Inventory
Identify Menu Options Per Employee Authorization Level
Create New User Account
Customer Menu Options
Display Manage Inventory Menu Options
Customer Login
Determines Cost of Shipping
Display Menu Options
Identify Employee Authoirzation Level
Determines User Access Priveleges
Determines Date of Shipping Transaction
Prints Financial Documents Available
Display List of Existing Users
Determines Date of Delivery
Calculated Total Shipping Transaction Sales
Saves Changes to User Account Information
Creates New Shipping Transaction
Displays net Income from Completed Shipping Transactions
Determines Package Deliverer
Handle Customer's Transactions
Calculate Total Expenses Identify Expenses Paid
Updates Transaction Status
Idetify Expenses Paid Total Shipping Transac Sales and Net Profit
Displays Statements of Cash Flow and Invoices and Balances Sheets
Handle Customers Inquiry
Displays Package Details
Identify Delivery Status
Save Package Order in Incomplete Shipping Transactions
Saves Package Order to Inventory
Save Customer Message to Unanswered Questions
«include»
«extend»

CRC Cards

(Classes, Responsibilities, and Collaborators)

Class User

The basic responsibility of a User object is to maintain basic information about a single individual.

Responsibilities	*Collaborators*
Create a new object, given an individual’s user id, name, phone, and email	
Furnish the individual’s user id	
Furnish the individual’s first name	
Furnish the individual’s last name	
Furnish the individual’s phone number	
Furnish the individual’s email	
Update the stored information (except user id) about the individual	

Class Customer

The basic responsibility of a Customer object is to maintain information about a single customer.

Responsibilities	*Collaborators*
Create a new object, given an individual’s user id, name, phone, email, and address	
Furnish the individual’s user id	User
Furnish the individual’s first name	User
Furnish the individual’s last name	User
Furnish the individual’s phone number	User
Furnish the individual’s email	User
Furnish the individual’s address	
Update the stored information (except the user id)	User

Class Employee

The basic responsibility of an Employee object is to maintain information about a single employee.

Responsibilities	*Collaborators*
Create a new object, given an individual’s name, phone, email, ssn, monthly salary, bank account number, employment status, system access rights, and pin number	
Furnish the individual’s user id	User
Furnish the individual’s first name	User
Furnish the individual’s last name	User

Furnish the individual’s phone number	User
Furnish the individual’s email	User
Furnish the individual’s ssn	
Furnish the individual’s monthly salary	
Furnish the individual’s bank account number	
Furnish the individual’s employment status	
Furnish the individual’s system access rights	
Furnish the individual’s pin number	
Update the stored information (except the user id) about the individual	

Class Package

The basic responsibility of a Package object is to maintain information about a single package

Responsibilities	*Collaborators*
Create a new object, given the tracking number, type, specification, mailing class, weight, volume, order specific details, location	
Furnish the package’s tracking number	
Furnish the package’s type	
Furnish the package’s specification	
Furnish the package’s mailing class	
Furnish the package’s weight	
Furnish the package’s volume	
Furnish the package’s other specific details	
Furnish the package’s location	
Update the stored information about the package	

Class PackageTransactionsSystem

The basics responsibility of a PackageTransactionsSystem object is to maintain information about package transactions

Responsibilities	*Collaborators*
Create a new object, given the customer id, package tracking number, cost of shipping, employee id, date transaction was created, date package was delivered, package status	Package, Customer
Furnish the package transaction’s customer id	Package
Furnish the package transaction’s package tracking number	Package
Furnish the package transaction’s cost of shipping	

Furnish the package transaction's employee id for deliverer	
Furnish the package transaction's date transaction was created	
Furnish the package transaction's date package was delivered	
Furnish the package transaction's package status	
Update stored information about package transactions	
Allow the system to perform calculations showing total sale transactions	
Allow the system to authenticate User	Login, User, Employee, Customer

Class InventoryManagementSystem

The basic responsibility of an InventorySystem object is to carryout various use cases.

Responsibilities	*Collaborators*
Allow the system to authenticate employee	Login, User, Employee
Allow the employee to perform Display Menu Options	
Allow the employee to perform Find Package by Tracking Number	Package
Allow the employee to perform the Update Package Information	Package
Allow the system to perform the Identify Delivery Status	Package
Allow the system to remove the Delivered Packages from inventory list	Package
Allow the system to save all Undelivered Packages to the inventory list	Package
Allow the system to perform the Display All Packages	

Class AccountingManagementSystem

The basic responsibility of an AccountingManagement object is to carryout various use cases.

Responsibilities	*Collaborators*
Furnish account's total sales	PackageTransactionsSystem
Furnish account's total expenses	Employee
Furnish account's net profit	
Furnish account's net loss	
Furnish account's net income	
Update stored information about account	
Allow the system to authenticate administrator	Login, User, Employee
Allow the administrator to perform Display Menu Options	
Allow the system to perform Calculate Total Shipping Transaction Sales	
Allow the system to perform Calculate Net Income	
Allow the system to perform Calculate Total Expenses	
Allow the system to perform Identify Expenses Paid	

Allow the system to Display Invoices and Balance Sheets

Allow the administrator to print Financial Documents Available

Allow the system to save the date to a secure file

Class LogIn

The basic responsibilities of a LogIn object is to carryout various use cases.

Responsibilities	*Collaborators*
Furnish where or not an employee is logged in	Employee
Allow the system to Find Employee by employee id	Employee
Allow the system to Grant the User Qualifying System Access Rights	Login, Employee

Class UserManagementSystem

The basic responsibility of UserManagementSystem object is to carryout various use cases.

Responsibilities	*Collaborators*
Allow the system to authenticate Employee	Login, User, Employee
Allow the system to determine user-access-privileges granted based on the type of user	
Allow the System to grant limited view of system function based on user-access-privileges granted	
Allow the Employee to perform Display Menu Options	Login, User, Employee
Allow the Employee to perform Display List of Users	Login, User, Employee
Allow the Employee to perform Find User by User Id	Login, User, Employee
Allow the Employee to perform create new customer account	User, Customer
Allow the Employee to perform Update Customer Account Information	User, Customer, Login, Employee
Allow system to perform Save Changes to User Account Information	User, Customer

Class ShippingStoreWebsite

The basic responsibility of ShippingStoreWebsite object is to carryout various use cases.

Responsibilities	*Collaborators*
Allow new customer to perform Create New Account	User, Customer
Allow existing customer to perform Login with Established Credentials	User, Customer, Login
Allow customer to perform Create New Shipping Transactions	Customer, Package, PackageTransactionsSystem
Allow customer to perform Print Receipt/Shipping Label	Customer, Package, PackageTransactionsSystem
Allow customer to perform Send Message to Shipping Store	Customer, Message, MessageBox
Allow system to save customer messages to the Shipping Store's Message Box	

Allow the system to perform Display Customer Menu Options	
Allow the system to perform Save Package to Incomplete	Customer, Package, PackageTransactionsSystem
Allow the system to display package details	Customer, Package, PackageTransactionsSystem

Class Message

The basic responsibility of Message is to maintain information about a single message

Responsibilities	*Collaborators*
Create a new object, given a customer id, message title, message body	Customer
Furnish the message's customer id,	
Furnish the message's title,	
Furnish the message's body	
Update the stored information about the message	

Class MessageBox

The basic responsibility of a MessageBox is to manage messages and carryout various use cases

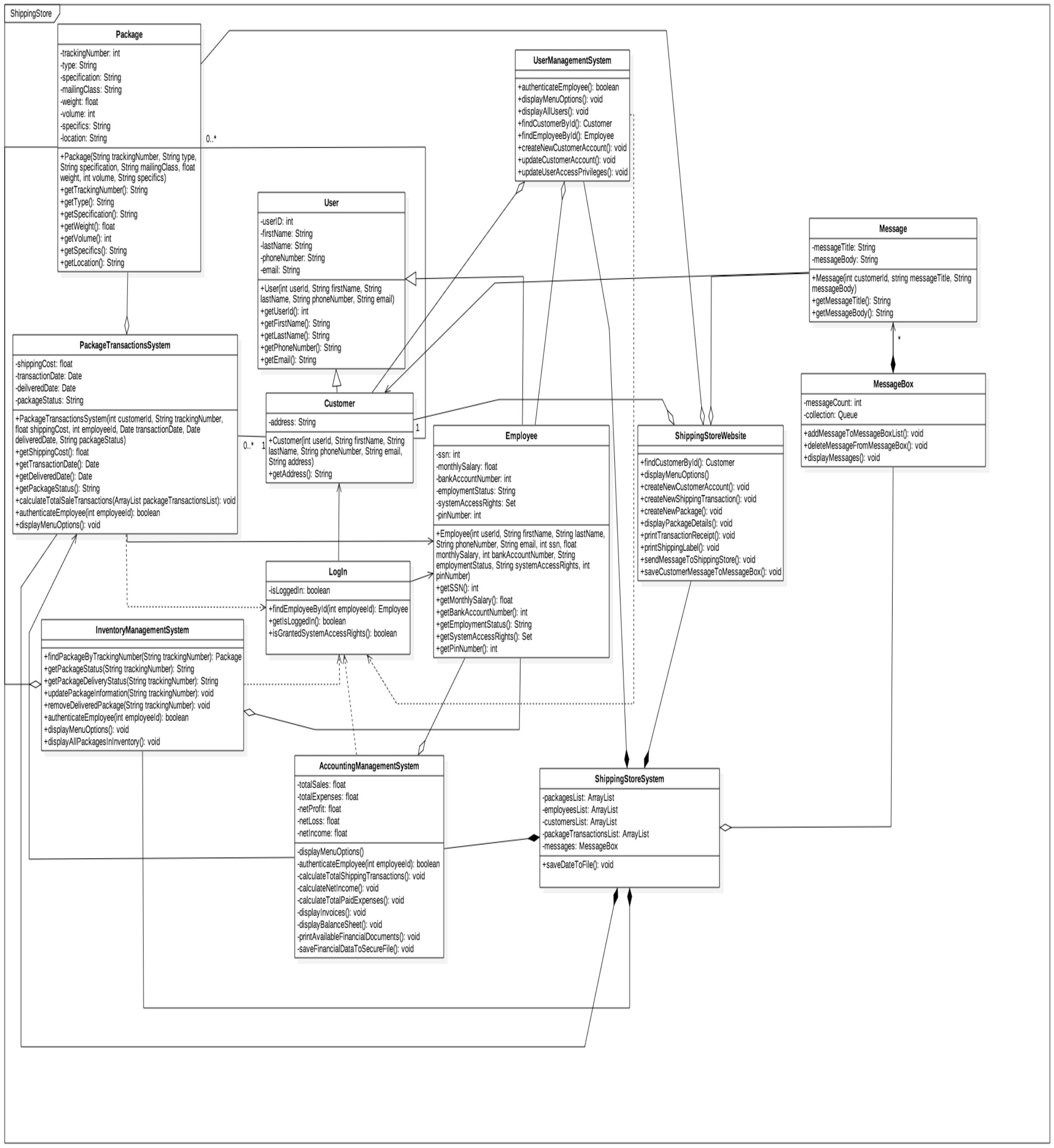
Responsibilities	*Collaborators*
Create a list to hold all messages	Message
Allow employees perform Respond to Customer Inquiries	
Allow employees to delete messages	Message

Class ShippingStoreSystem

The basic responsibility of a Shipping is to manage interaction between the main program and the classes, and carryout various use cases

Responsibilities	*Collaborators*
Create a list to hold all Packages	Package
Create a list to hold all Employees	Employee
Create a list to hold all Customers	Customer
Create a list to hold all PackageTransactions	PackageTransactionsSystem
Create a MessageBox object to hold all Messages	Message, MessageBox
Create a PackageTransactionsSystem object to access methods of the class	PackageTransactionsSystem
Create an InventoryManagementSystem object to access methods of the class	InventoryManagementSystem
Create an AccountingManagementSystem object to access methods of the class	AccountingManagementSystem
Create ShippingStoreWebsite object to access methods of the class	ShippingStoreWebsite
Persist all lists, queues	

(Class Diagram)


ShippingStore
Package
-trackingNumber: int
-type: String
-specification: String
-mailingClass: String
-weight: float
-volume: int
-specifics: String
-location: String
+Package(String trackingNumber, String type, String specification, String mailingClass, float weight, int volume, String specifics)
+getTrackingNumber(): String
+getType(): String
+getSpecification(): String
+getWeight(): float
+getVolume(): int
+getSpecifics(): String
+getLocation(): String
0..*
UserManagementSystem
+authenticateEmployee(): boolean
+displayMenuOptions(): void
+displayAllUsers(): void
+findCustomerById(): Customer
+findEmployeeById(): Employee
+createNewCustomerAccount(): void
+updateCustomerAccount(): void
+updateUserAccessPrivileges(): void
User
-userID: int
-firstName: String
-lastName: String
-phoneNumber: String
-email: String
+User(int userId, String firstName, String lastName, String phoneNumber, String email)
+getUserId(): int
+getFirstName(): String
+getLastName(): String
+getPhoneNumber(): String
+getEmail(): String
Message
-messageTitle: String
-messageBody: String
+Message(int customerId, string messageTitle, String messageBody)
+getMessageTitle(): String
+getMessageBody(): String
*
PackageTransactionsSystem
-shippingCost: float
-transactionDate: Date
-deilveredDate: Date
-packageStatus: String
+PackageTransactionsSystem(int customerId, String trackingNumber, float shippingCost, int employeeId, Date transactionDate, Date deliveredDate, String packageStatus)
+getShippingCost(): float
+getTransactionDate(): Date
+getDeliveredDate(): Date
+getPackageStatus(): String
+calculateTotalSaleTransactions(ArrayList packageTransactionsList): void
+authenticateEmployee(int employeeId): boolean
+displayMenuOptions(): void
0..*
1
Customer
-address: String
+Customer(int userId, String firstName, String lastName, String phoneNumber, String email, String address)
+getAddress(): String
1
MessageBox
-messageCount: int
-collection: Queue
+addMessageToMessageBoxList(): void
+deleteMessageFromMessageBox(): void
+displayMessages(): void
Employee
-ssn: int
-monthlySalary: float
-bankAccountNumber: int
-employmentStatus: String
-systemAccessRights: Set
-pinNumber: int
+Employee(int userId, String firstName, String lastName, String phoneNumber, String email, int ssn, float monthlySalary, int bankAccountNumber, String employmentStatus, String systemAccessRights, int pinNumber)
+getSSN(): int
+getMonthlySalary(): float
+getBankAccountNumber(): int
+getEmploymentStatus(): String
+getSystemAccessRights(): Set
+getPinNumber(): int
ShippingStoreWebsite
+findCustomerById(): Customer
+displayMenuOptions()
+createNewCustomerAccount(): void
+createNewShippingTransaction(): void
+createNewPackage(): void
+displayPackageDetails(): void
+printTransactionReceipt(): void
+printShippingLabel(): void
+sendMessageToShippingStore(): void
+saveCustomerMessageToMessageBox(): void
LogIn
-isLoggedIn: boolean
+findEmployeeById(int employeeId): Employee
+getIsLoggedIn(): boolean
+isGrantedSystemAccessRights(): boolean
InventoryManagementSystem
+findPackageByTrackingNumber(String trackingNumber): Package
+getPackageStatus(String trackingNumber): String
+getPackageDeliveryStatus(String trackingNumber): String
+updatePackageInformation(String trackingNumber): void
+removeDeliveredPackage(String trackingNumber): void
+authenticateEmployee(int employeeId): boolean
+displayMenuOptions(): void
+displayAllPackagesInInventory(): void
AccountingManagementSystem
-totalSales: float
-totalExpenses: float
-netProfit: float
-netLoss: float
-netIncome: float
-displayMenuOptions()
-authenticateEmployee(int employeeId): boolean
-calculateTotalShippingTransactions(): void
-calculateNetIncome(): void
-calculateTotalPaidExpenses(): void
-displayInvoices(): void
-displayBalanceSheet(): void
-printAvailableFinancialDocuments(): void
-saveFinancialDataToSecureFile(): void
ShippingStoreSystem
-packagesList: ArrayList
-employeesList: ArrayList
-customersList: ArrayList
-packageTransactionsList: ArrayList
-messages: MessageBox
+saveDateToFile(): void

** Please assume that there are setters for every getters

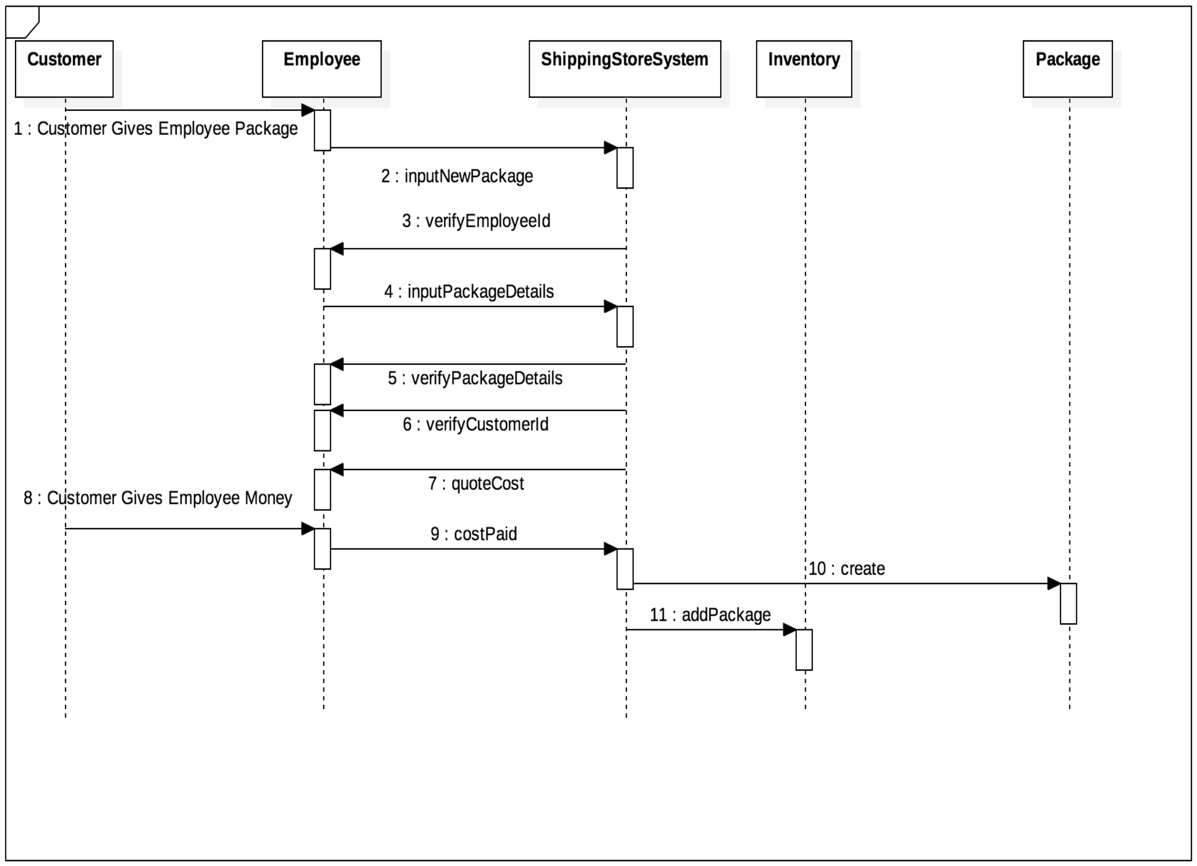
** Please assume that there are extended versions of the Package class for storing package specific details

** Please assume that each class has the appropriate overridden methods (e.g. toString ())

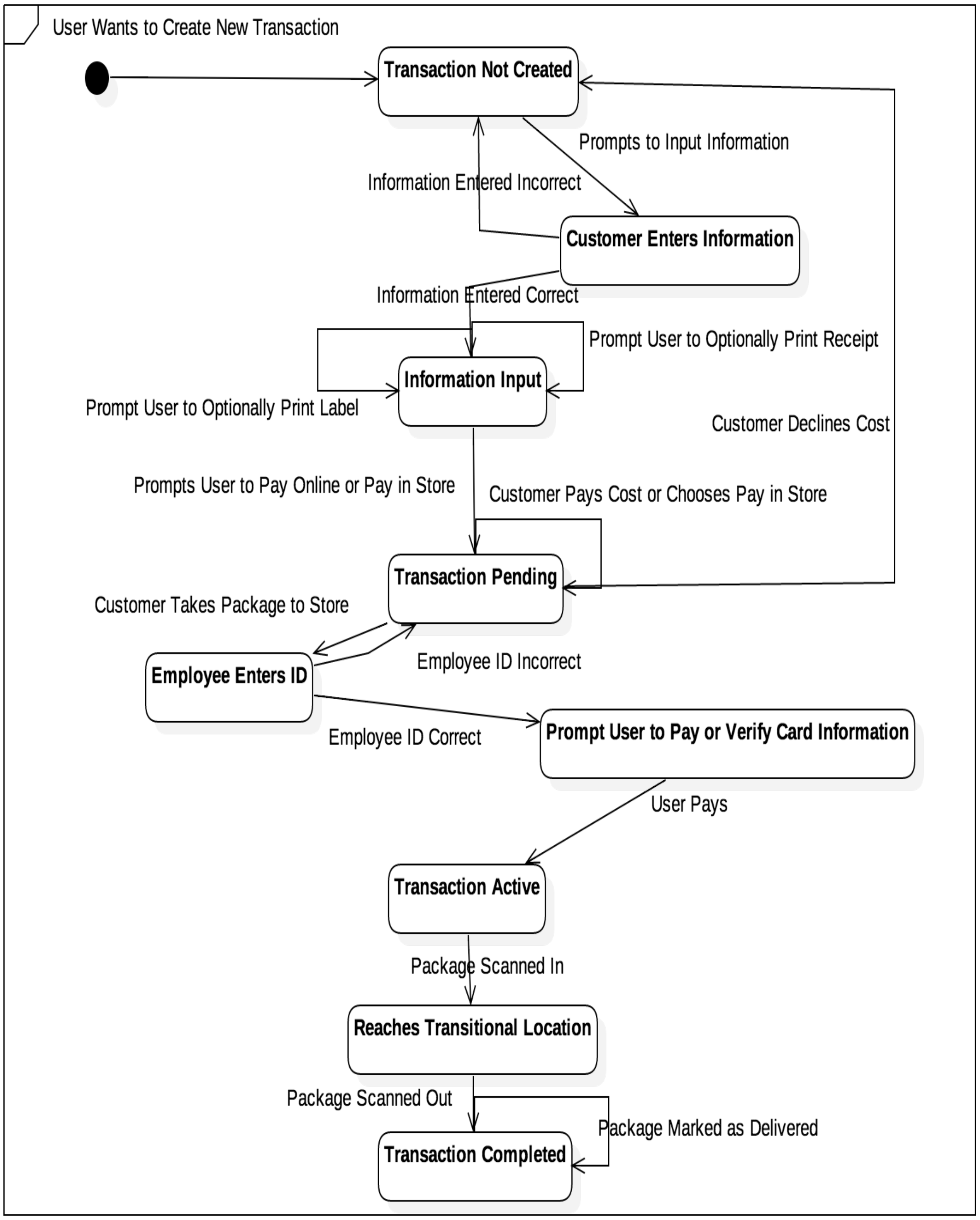
** Please assume that saving data is handled by arbitrary method in ShippingStoreSystem

** Creation of various class objects not included in diagram

SEQUENCE DIAGRAM



State Machine Diagram


User Wants to Create New Transaction
Transaction Not Created
Prompts to Input Information
Information Entered Incorrect
Customer Enters Information
Information Entered Correct
Prompt User to Optionally Print Receipt
Information Input
Prompt User to Optionally Print Label
Customer Declines Cost
Prompts User to Pay Online or Pay in Store
Customer Pays Cost or Chooses Pay in Store
Transaction Pending
Customer Takes Package to Store
Employee Enters ID
Employee ID Incorrect
Employee ID Correct
Prompt User to Pay or Verify Card Information
User Pays
Transaction Active
Package Scanned In
Reaches Transitional Location
Package Scanned Out
Package Marked as Delivered
Transaction Completed