Object Oriented Methods: CS377

A Pet Nose Best: Use of Object-Oriented and Recap of Design Changes

Antoine Gaton

October 6, 2024

Table of Contents

**Project Description (Week1)3**

A Pet Nose Best Ordering System**3**

**Use Case Diagram (Week1)4**

Diagram**6**

**Class Diagram (Week2)7**

Classes/Properties**7**

Diagram**9**

**Sequence and Collaboration Diagram (Week3)11**

Use Case #1: Create Order and Payment Entry**11**

Use Case #2: Update Payment Entry**13**

Use Case #3: View Order and Payment Entry**15**

Use Case #4: Delete Order and Payment Entry**17**

Use Case#5: Read Existing Orders *(new)* **19**

**State Transition and Activity Diagram (Week4)21**

Create Order and Payment Entry**21**

Update Payment Entry**22**

View Order and Payment Entry**23**

Delete Order and Payment Entry**24**

Delete Order and Payment Entry**25**

**Follow-Up Discussion on Use of Object-Oriented and Recap of Design Changes (Week5)27**

**Change Control Document (Week5)29**

# Project description (Week 1)

**A Pet Nose Best Ordering System**

* Enterprise
  + A Pet Nose Best
* Business Description
  + *A Pet Nose Best* is a full-service pet care company dedicated to making life easier for pet owners by offering a range of services that ensure pets are happy, healthy, and well cared for. The services include dog walking, cat and dog sitting, appointment pickups/drop-offs, pet parties, and daycare. The business is focused on creating a family-like environment where pets feel loved and cared for while their owners are away.
* System Scope
  + The goal of this project is to design an ordering and payment system for existing customers to book pet care services. The system will allow users to select services like dog walking or daycare, make a booking, and process payments via credit card. Since the focus is on an ordering system, customer management and scheduling features are out of scope. The system will only handle bookings and payment processing for existing customers.

# Use Case Diagram (Week 1)

* Use Case Diagram
  + We can design the system with three types of users (actors):
    - *Customer*: The pet owner who books and pays for services.
    - *Clerk*: A staff member who manages orders and payments.
    - *Manager*: A senior staff member who can manage all orders and payments, including deleting records.
* Use Cases (*Revised*)
  + *Create Order and Payment Entry*: This allows customers to book a service and make a payment.
  + *Update Payment Entry*: Clerks and managers can update payment details for an order if there was an issue with the original entry.
  + *View Order and Payment Entry*: Clerks and managers can view all orders and associated payment information.
  + *Delete Order and Payment Entry*: Managers can delete an order or payment record if a service is canceled or needs to be corrected.
  + *Read Existing Orders (New)*: Authorized external parties can access and view information about existing orders in the system for purposes such as auditing or reporting.
* Use Case Narrative
  + **#1: Create Order and Payment Entry**
    - *Actors*: Customer, Payment Processor
    - *Description*: A customer logs into the system, selects a pet care service, and proceeds to the checkout page. The system validates the booking, calculates the total, and prompts the customer to enter credit card details. Upon successful payment, the order is confirmed, and a receipt is emailed to the customer.
    - *Preconditions*:
      * The customer is already registered and logged into the system.
      * The customer has selected at least one service to book.
    - *Postconditions*:
      * A new order is created in the system.
      * The payment is successfully processed and recorded.
  + **#2: Update Payment Entry**
    - *Actors*: Clerk, Manager
    - *Description*: A clerk or manager can view and update the payment information for a booking if there was an issue with the original payment. The updated payment details are reprocessed through the payment gateway.
    - *Preconditions*:
      * The original order and payment information exist in the system.
    - *Postconditions*:
      * The payment entry is updated with the new information.
      * A receipt is sent to the customer if the payment is reprocessed.
  + **#3: View Order and Payment Entry**
    - *Actors*: Clerk, Manager
    - *Description*: Clerks and managers can view the list of all customer orders and payments. This view includes details like the order number, service booked, date, total cost, and payment status.
    - *Preconditions*:
      * The system has existing orders and payments.
    - *Postconditions*:
      * No changes are made to the system. The data is only viewed.
  + **#4: Delete Order and Payment Entry**
    - *Actors*: Manager
    - *Description*: A manager can delete an order or payment record if a service is canceled or entered in error. This functionality ensures that invalid or incorrect entries do not remain in the system.
    - *Preconditions*:
      * An order and payment entry must exist in the system.
    - *Postconditions*:
      * The order and payment entry are deleted from the system.
  + **#5: Read Existing Orders**
    - Actors: External Party
    - Description: An authorized external party can access and read existing order information from the system. This allows for auditing, reporting, or integration with other systems.
    - Preconditions:
      * The external party has proper authentication and authorization.
      * There are existing orders in the system.
    - Postconditions:
      * The external party has retrieved the requested order information.
      * No changes are made to the system; data is only read.

***Use Case Diagram***

A diagram of a diagram

Description automatically generated

# Class Diagram (Week 2)

* **Class Diagram Overview**
  + The following class diagram reflects the object-oriented design for the new ordering system based on the use case model created in Week 1. This diagram identifies the objects involved, including the key entities, their properties, and relationships. Inheritance, relationships, and data types are shown to reflect the system’s structure.
* **Objects from the Use Case Model:**

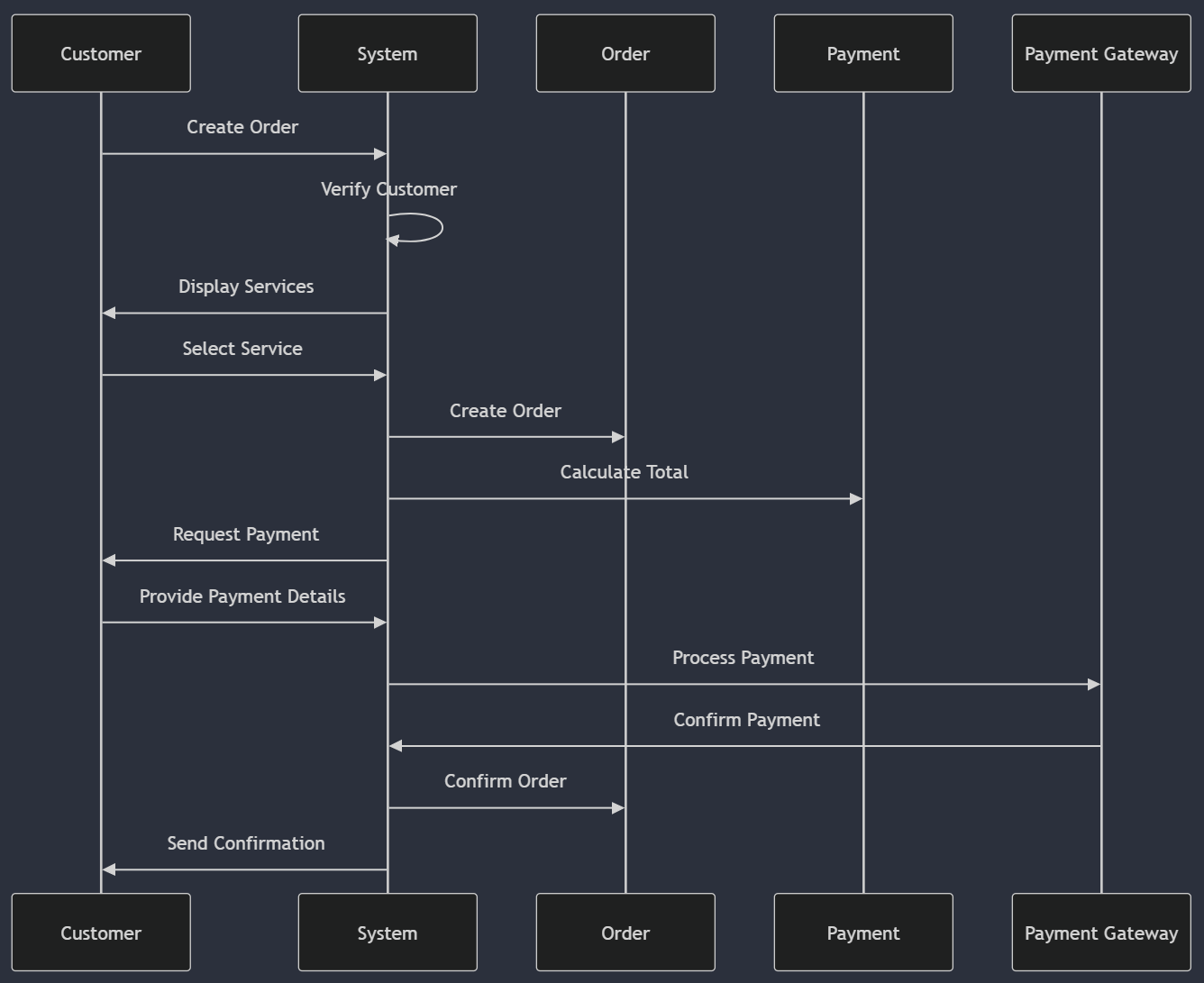
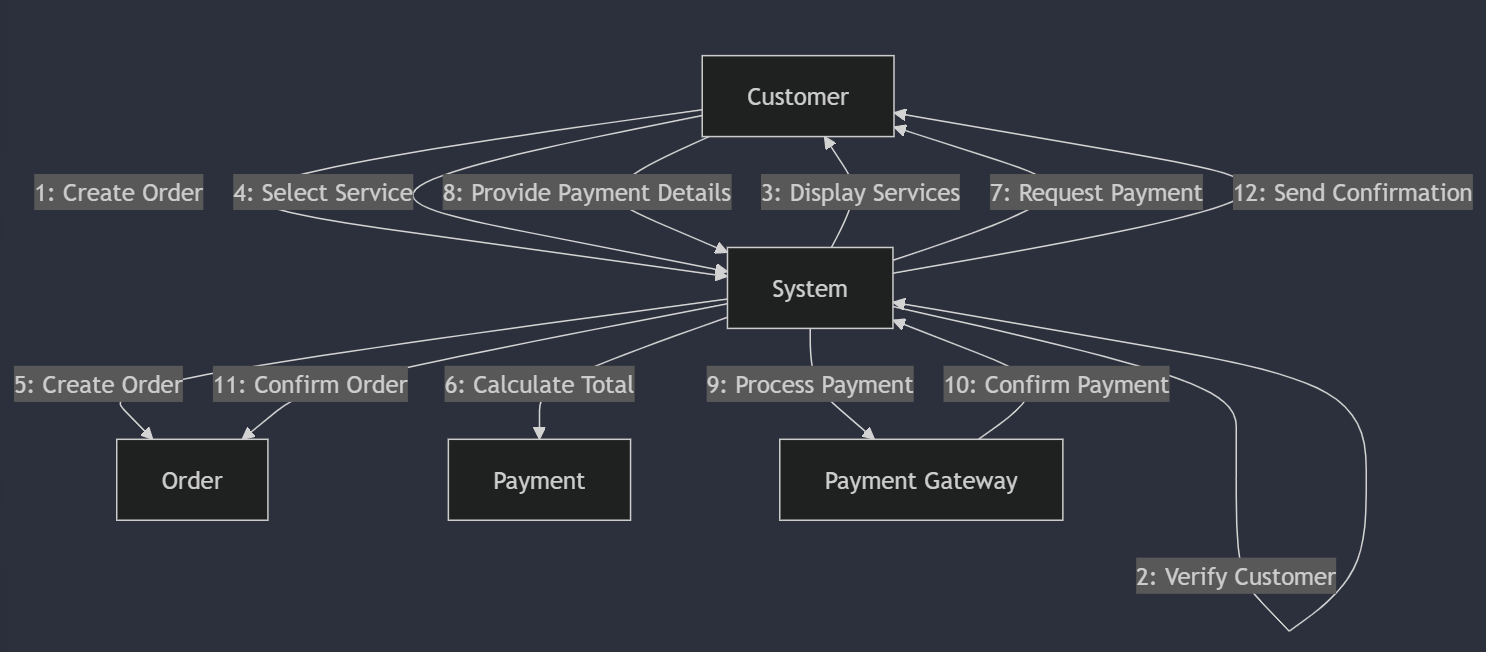
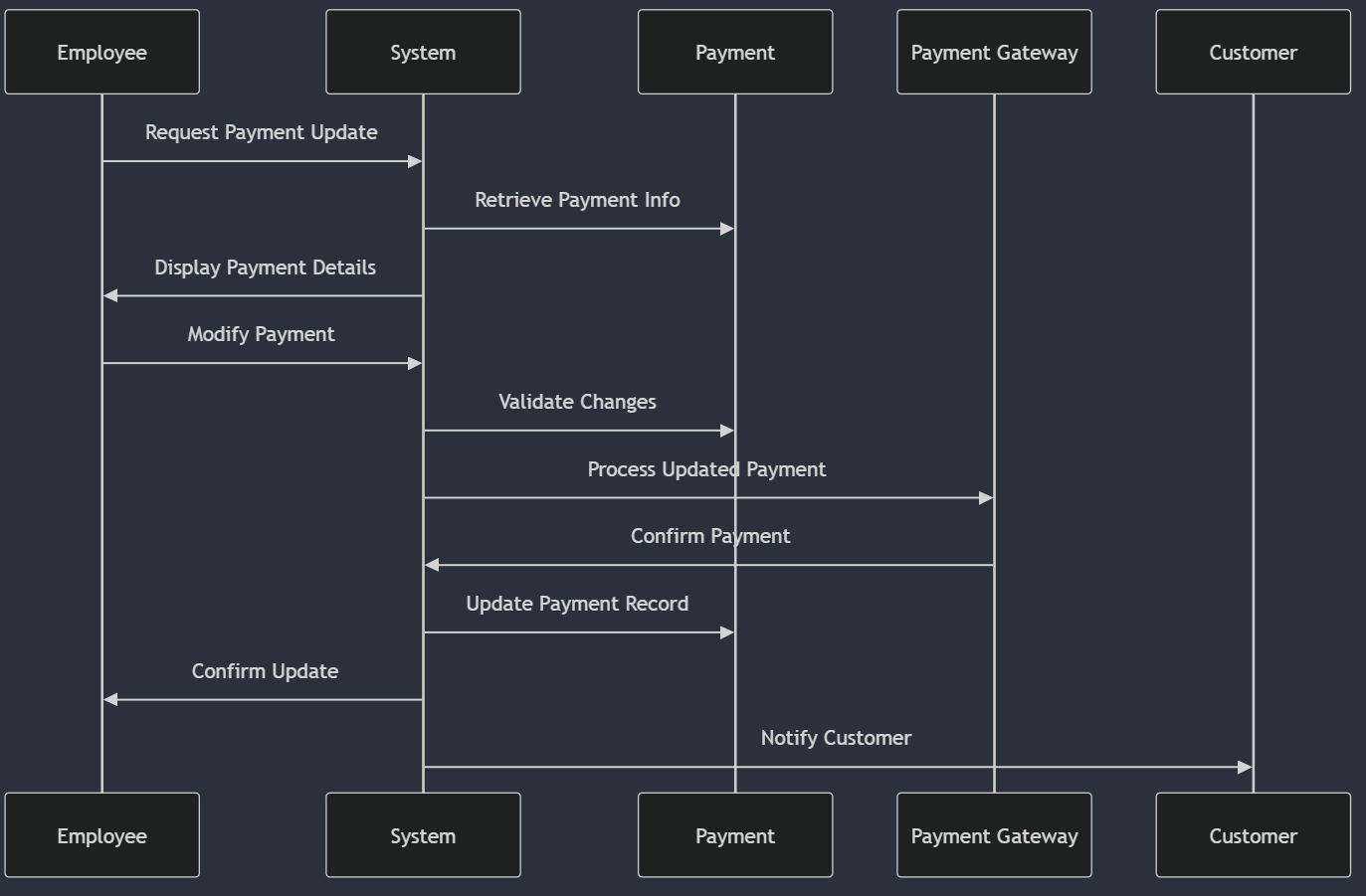
1. *Customer*
2. *Clerk*
3. *Manager*
4. *Order*
5. *Payment*

* **Class Diagram Elements**
  + Classes and Properties *(Revised)*:
    - *Customer*
      1. Attributes
         1. customerID: int
         2. name: String
         3. email: String
         4. phone: String
      2. Methods
         1. createOrder()
         2. viewOrder()
    - *Order*
      1. Attributes
         1. orderID: int
         2. service: String
         3. orderDate: Date
         4. status: String
      2. Methods *(Revised)*
         1. updateOrder()
         2. deleteOrder()
         3. getOrderDetails() *(new)*
    - *Payment*
      1. Attributes
         1. paymentID: int
         2. orderID: int
         3. amount: float
         4. paymentDate: Date
         5. status: String
      2. Methods
         1. processPayment()
         2. updatePayment()
    - *Clerk (Inherits from Staff)*
      1. Attributes
         1. clerkID: int
      2. Methods
         1. updateOrder()
         2. viewOrders()
    - *Manager (Inherits from Staff)*
      1. Attributes
         1. managerID: int
      2. Methods
         1. deleteOrder()
         2. managePayments()
    - *Staff (Base class for Clerk and Manager)*
      1. Attributes
         1. staffID: int
         2. name: String
         3. role: String
      2. Methods
         1. viewOrders()
    - *ExternalParty*
      1. Attributes:
         1. partyID: int
         2. name: String
         3. accessLevel: String
      2. Methods:
         1. readOrders()
  + Relationships
* ***Customer*** has a relationship with **Order**: A customer can create and view multiple orders.
* ***Order*** has a relationship with ***Payment***: An order is associated with a payment.
* ***Clerk*** and ***Manager*** can view and update orders; however, only the ***Manager*** has the authority to delete orders and manage payments.
  + Inheritance and Class Diagram Notations
* ***The Clerk*** and ***Manager*** inherited from the ***Staff*** base class, which defines shared attributes and behaviors.
* ***Customer***, ***Order***, and ***Payment*** are independent classes but are linked through their relationships.

***Class Diagram***



# Sequence and Collaboration Diagram (Week 3)

* **Use Case #1**: Create Order and Payment Entry
  + Sequence Diagram
    - Narrative
      * This sequence diagram illustrates the process of creating an order and processing a payment in the A Pet Nose Best system. The interaction begins when a Customer initiates the order creation process. The system then verifies the Customer's account, retrieves available services, and presents them to the Customer. After the Customer selects a service, the Order is created and associated with the Customer. The system then calculates the total cost and requests payment information. Once the payment details are provided, the system processes the payment through a Payment Gateway. Upon successful payment, the Order is confirmed, and a confirmation is sent to the Customer. This sequence ensures a smooth, step-by-step process for order creation and payment.
  + Collaboration Diagram
    - Narrative
      * The collaboration diagram for creating an order and payment entry shows the interactions between objects in a non-linear fashion. It emphasizes the relationships between the Customer, OrderSystem, Order, PaymentSystem, and PaymentGateway objects. The diagram illustrates how messages flow between these objects to accomplish the task of creating an order and processing a payment. The numbered messages indicate the sequence of operations, starting with the Customer initiating the order and ending with the confirmation being sent back to the Customer. This diagram helps visualize how different parts of the system collaborate to complete the order and payment process.
* **Use Case #2**: Update Payment Entry
  + Sequence Diagram
    - Narrative
      * This sequence diagram depicts the process of updating a payment entry, typically performed by a Clerk or Manager. The process begins when the employee requests to update a payment. The system then retrieves the existing payment information and presents it for modification. After the employee makes the necessary changes, the system validates the new information and processes the updated payment through the Payment Gateway. If successful, the payment record is updated in the system, and a confirmation is sent to both the employee and the customer. This sequence ensures accurate and traceable updates to payment information.
  + A computer screen shot of a diagram

    Description automatically generatedCollaboration Diagram
    - Narravite
      * The collaboration diagram for updating a payment entry illustrates the interactions between the Employee (Clerk or Manager), PaymentSystem, Order, and PaymentGateway objects. It shows how these objects work together to update an existing payment entry. The diagram emphasizes the central role of the PaymentSystem in coordinating the update process, from retrieving the initial payment information to processing the updated payment and confirming the changes. This representation helps in understanding the relationships and dependencies between different parts of the system during the payment update process.
* **Use Case #3**: View Order and Payment Entry
  + A diagram of a system

    Description automatically generatedSequence Diagram
    - Narrative
      * This sequence diagram shows the process of viewing an order and its associated payment entry, typically performed by a Clerk or Manager. The sequence begins with the employee requesting to view an order. The system then retrieves the order details and the associated payment information. This information is then presented to the employee for review. The diagram illustrates a straightforward process of information retrieval and display, emphasizing the system's ability to provide comprehensive order and payment information on demand.
  + A diagram of a system

    Description automatically generatedCollaboration Diagram
    - Narrative
      * The collaboration diagram for viewing an order and payment entry demonstrates the interactions between the Employee, OrderSystem, Order, and PaymentSystem objects. It shows how these objects collaborate to retrieve and display order and payment information. The diagram emphasizes the central role of the OrderSystem in coordinating the information retrieval process, working in conjunction with the Order and PaymentSystem to compile a complete view of the order and its associated payment. This representation helps in understanding how different parts of the system work together to provide a comprehensive view of order and payment information.
* **Use Case #4**: Delete Order and Payment Entry
  + A screenshot of a computer screen

    Description automatically generatedSequence Diagram
    - Narrative
      * This sequence diagram illustrates the process of deleting an order and its associated payment entry, an action typically restricted to Managers. The sequence begins with the Manager requesting to delete an order. The system then retrieves the order and payment details, presenting them for review. After the Manager confirms the deletion, the system processes the request, first deleting the payment entry and then the order itself. Finally, a confirmation of the deletion is sent to the Manager. This sequence ensures that deletions are performed systematically and with proper authorization.
  + A diagram of a system

    Description automatically generatedCollaboration Diagram
    - Narrative
      * The collaboration diagram for deleting an order and payment entry shows the interactions between the Manager, OrderSystem, Order, and PaymentSystem objects. It illustrates how these objects work together to complete the deletion process. The diagram emphasizes the OrderSystem's role in coordinating the deletion, interacting with both the Order and PaymentSystem to ensure complete removal of the order and its associated payment information. This representation helps in understanding the relationships and dependencies between different parts of the system during the deletion process, highlighting the importance of maintaining data integrity across related records.
* **Use Case#5**: Read Existing Orders *(new)*
  + A diagram of a system

    Description automatically generatedSequence Diagram
    - Narrative
      * This sequence diagram illustrates the process of an External Party accessing and reading existing order data from the A Pet Nose Best system. The interaction begins when the External Party requests to read orders. The system first authenticates the request through an Authentication service to ensure the External Party has the necessary permissions. Once authenticated, the system retrieves the relevant order data and returns it to the External Party. This sequence ensures secure and controlled access to order information for authorized external entities.
  + A diagram of a system

    Description automatically generatedCollaboration Diagram
    - Narrative
      * The collaboration diagram for reading existing orders demonstrates the interactions between the External Party, System, Order, and Authentication objects. It shows how these components work together to provide secure access to order information. The diagram emphasizes the System's central role in coordinating the process, from authenticating the External Party's request to retrieving and delivering the order data. This representation helps in understanding the security measures and data flow involved in allowing external access to order information, highlighting the importance of authentication in the process.
        + This use case is particularly important for scenarios where external auditors, regulatory bodies, or integrated systems need access to order data. It demonstrates the system's capability to provide controlled access to sensitive information while maintaining security and data integrity.

# State Transition and Activity Diagram (Week 4)

* Use Cases
  + **Create Order and Payment Entry**
    - A screenshot of a computer

      Description automatically generatedA screenshot of a computer

      Description automatically generatedActivity and State Transition Diagram
  + **Update Payment Entry**
    - A screenshot of a flowchart

      Description automatically generatedA screenshot of a computer

      Description automatically generatedActivity and State Transition Diagram
  + **View Order and Payment Entry**
    - Activity and State Transition Diagram

A screenshot of a computer

Description automatically generatedA diagram of a process

Description automatically generated

* + **Delete Order and Payment Entry**
    - A screenshot of a flowchart

      Description automatically generatedA screenshot of a computer

      Description automatically generatedActivity and State Transition Diagram
  + **Read Existing Orders**
    - A screenshot of a computer flowchart

      Description automatically generatedActivity Diagram
    - A diagram of a system

      Description automatically generatedState Transition Diagram

# Follow-Up Discussion on Use of Object-Oriented and Recap of Design Changes (Week 5)

**Recap of Each Model Used in the Design**

1. ***Use Case Model***: This model helped identify the main actors and their interactions with the system. It provided a clear overview of the system's functionality and helped in understanding the new requirement for external party access.
2. ***Class Diagram***: The class diagram showed the structure of the system, including the relationships between different entities. It was particularly useful in identifying where to add the new ExternalParty class and how it relates to existing classes.
3. ***Sequence and Collaboration Diagrams***: These diagrams illustrated the flow of messages between objects for each use case. They were instrumental in designing the new functionality for reading existing orders by an external party.
4. ***State Transition and Activity Diagrams***: These diagrams helped in understanding the different states of the system and the flow of activities. They were particularly useful in designing the process for external party authentication and order data retrieval.

**Discussion: How Object-Oriented Methodology with UML Enhanced the Process of Developing the System Design**

The object-oriented methodology with UML significantly enhanced the process of developing the system design in several ways:

1. ***Clarity and Visualization***: UML diagrams provided a clear visual representation of the system's structure and behavior. This made it easier to understand complex relationships and processes, especially when communicating ideas with team members or stakeholders.
2. ***Modularity and Reusability***: The object-oriented approach encouraged thinking in terms of modular, reusable components. This made it easier to add new functionality (like the external party access) without significantly disrupting the existing system.
3. ***Consistency***: Using UML ensured consistency across different aspects of the design. For example, the classes identified in the class diagram were consistently used in sequence and collaboration diagrams, ensuring a coherent overall design.
4. ***Scalability***: The object-oriented design made it easier to scale the system. Adding the new ExternalParty class and its associated functionalities was straightforward due to the modular nature of the design.
5. ***Traceability***: UML diagrams provided traceability from high-level use cases down to detailed class structures and interactions. This made it easier to ensure that all requirements were addressed in the design.
6. ***Iterative Development***: The UML models were easy to update and refine as new requirements emerged. This supported an iterative development process, allowing for continuous improvement of the design.
7. ***Standardization***: Using UML provided a standardized way of representing the system design. This is particularly valuable for future maintenance or when onboarding new team members.

The object-oriented methodology with UML provided a comprehensive toolkit for designing the system. It allowed for a systematic approach to incorporating new requirements while maintaining a clear overview of the entire system. This methodology not only enhanced the design process but also resulted in a more robust, flexible, and maintainable system design.

# Change Control Document (Week 5)

1. **Change**: Addition of 'status' attribute to Order class
   1. ***Original Information***: Order class did not have a status attribute
   2. ***Change Made***: Added 'status: String' to Order class attributes
   3. ***Location***: Class Diagram section
   4. ***Reason***: To track the current state of an order (e.g., pending, completed, cancelled)
2. **Change**: Modification of Payment class methods
   1. ***Original Information***: Payment class had only 'processPayment()' method
   2. ***Change Made***: Added 'updatePayment()' method to Payment class
   3. ***Location***: Class Diagram section
   4. ***Reason***: To allow for modification of payment details after initial processing
3. **Change**: Update to Manager class permissions
   1. ***Original Information***: Manager could only view and update orders
   2. ***Change Made***: Added 'deleteOrder()' and 'managePayments()' methods to Manager class
   3. ***Location***: Class Diagram and Use Case Diagram sections
   4. ***Reason***: To provide managers with more control over order and payment management
4. **Change**: Refinement of sequence diagram for 'Create Order and Payment Entry'
   1. ***Original Information***: Sequence diagram did not include payment processing step
   2. ***Change Made***: Added payment processing step in the sequence of actions
   3. ***Location***: Sequence and Collaboration Diagram section
   4. ***Reason***: To accurately represent the complete process of creating an order with payment
5. **Change**: Addition of 'View Order and Payment Entry' use case
   1. ***Original Information***: This use case was not initially included
   2. ***Change Made***: Added new use case for viewing orders and payments
   3. ***Location***: Use Case Diagram section
   4. ***Reason***: To provide a way for customers, clerks, and managers to view existing orders and payments