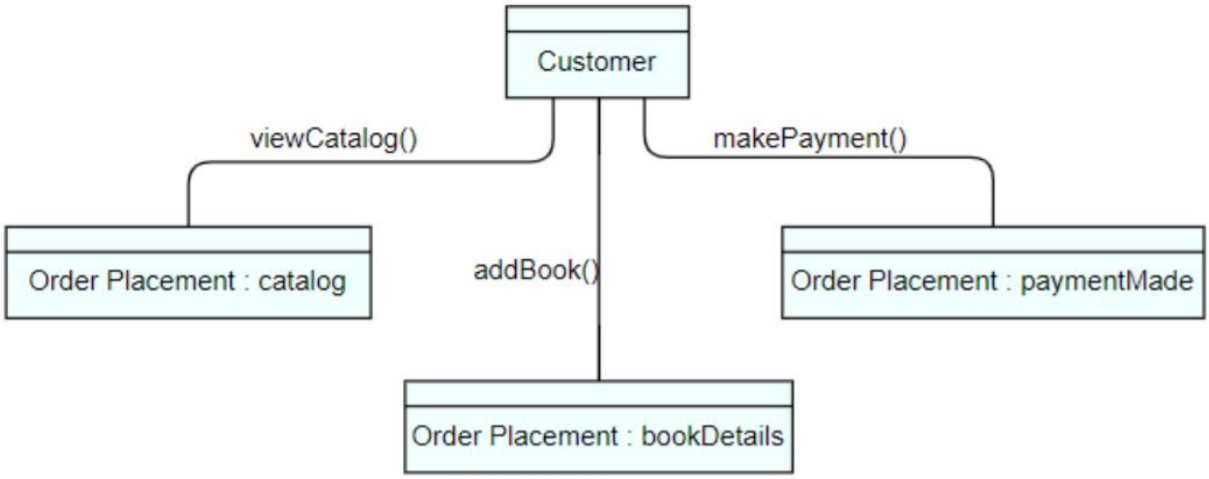
Homework 4 solutions:

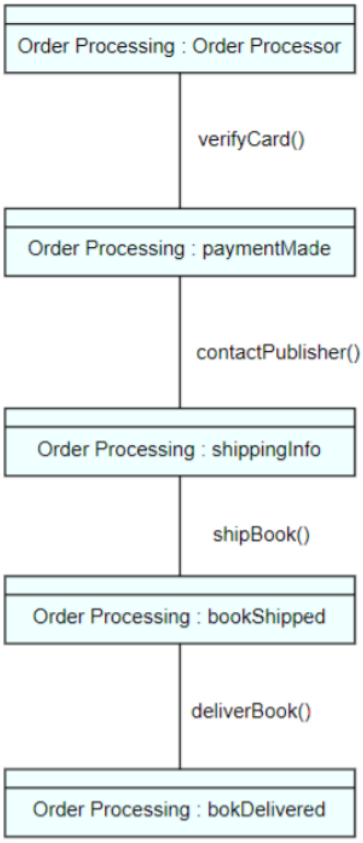
NOTE: These are examples of answers that would be enough for the questions. There could be many variations of the answer that is right.

Question 1a (4*10=40 Points): For each use case, identify system operations and develop one or more interaction diagrams, as applicable. Apply GRASP/GoF design patterns, as appropriate.

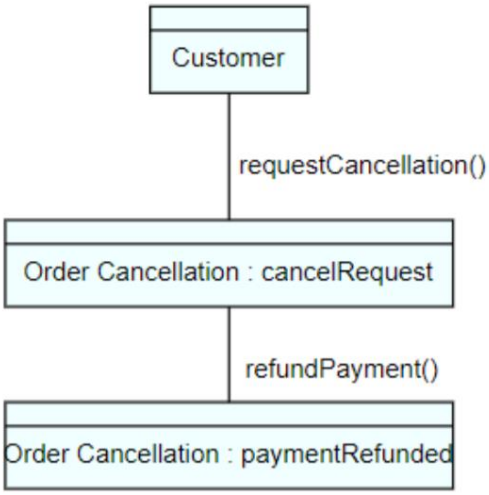
1. Order placement
    a. System Operations
        i. viewCatalog()
        ii. addBook()
        iii. makePayment()
    b. Interaction Diagrams



2. Order processing
    a. System Operations
        i. verifyCard()
        ii. contactPublisher()
        iii. shipBook()
        iv. deliverBook()
    b. Interaction Diagrams

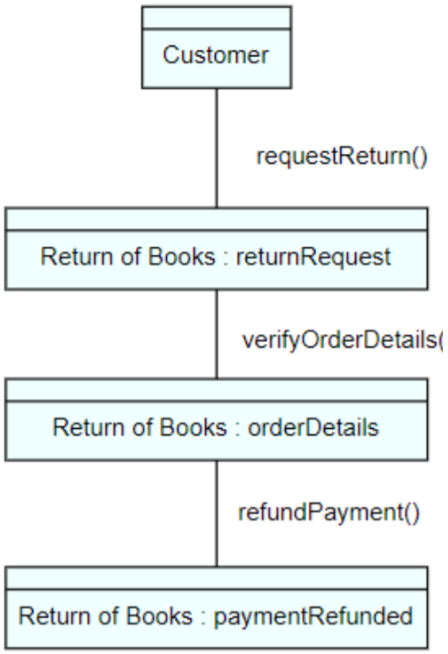


3. Order cancellation
    a. System Operations
        i. requestCancellation()
        ii. refundPayment()
    b. Interaction Diagrams



4. Return of books

a. System Operations
    i. requestReturn()
    ii. verifyOrderDetails()
    iii. refundPayment()
b. Interaction Diagrams



Question 1b (4*10=40 Points): Use at least ten different GRASP / GoF design patterns, to solve Question 1a. Provide a brief reasoning for applicability of each design pattern.

GRASP / GoF design patterns:

1. **High Cohesion**: Design elements to have strongly related and focused responsibilities
    a. Order Placement and Order Processing have this in that they both are strongly related and require similar attributes to work.
2. **Low Coupling**: Assign a responsibilities so that class depends less on other classes using a "need to know basis".
    a. Return of books and Order Cancellation are both smaller use cases and require data only if it is needed (need to know basis)
3. **Polymorphism**: Give the same name to services in different objects when the services are similar or related
    a. Both Return of books and Order Cancellation have the method refundPayment()
4. **Bridge**: A class of objects that acts as an interface between clients and an implementation.
    a. Order Placement is the bridge between the client and Ebook's System which is represented by Order Processing
5. **Expert**: allocate a responsibility to a class that has the information.
    a. Order Placement has the information for the other three use cases (starts the process)
6. **Builder**: How to constuct complex objects with varying parts? A director calls various builders as needed

a. Order Processor builds the order for the publisher

7. **Controller**: Assign the responsibility for handling system event messages to a class representing either the whole system, device, or subsystem, or representing the use case /scenario within which the system event occurs.
   a. Return of Book is able to "control" the data from Order Placement/Order Processing to complete a secondary task
8. **Chain of Responsibility**: Attach responsibilities to a linked list of abstract handlers each is a special concrete handler.
   a. Return of Book and Order Cancellation each handle secondary tasks not related to the main line of orders
9. **Composite**: Make complex and simple kinds of an object share behavior. And often the complex objects keep a list of their parts: each of which is a similar kind of object.
   a. Order Placement is a simpler "bridge" to the more complex Order Processing use case which is able to truly handle the creation of the book order
10. **Command**: Design objects that know how to do (and undo) a family of tasks and pass them to other objects to be used as needed.
    a. Order Cancellation is able to "command" the data from Order Placement/Order Processing to complete a secondary task

Question 2 (20 Points): Develop the design class diagram