SYS466: Analysis and Design using OO Models

Lecture 1: Introduction and System Sequence Diagrams

Key Reference: Applying UML and Patterns, 3rd edition, Craig Larman, Chapter 9

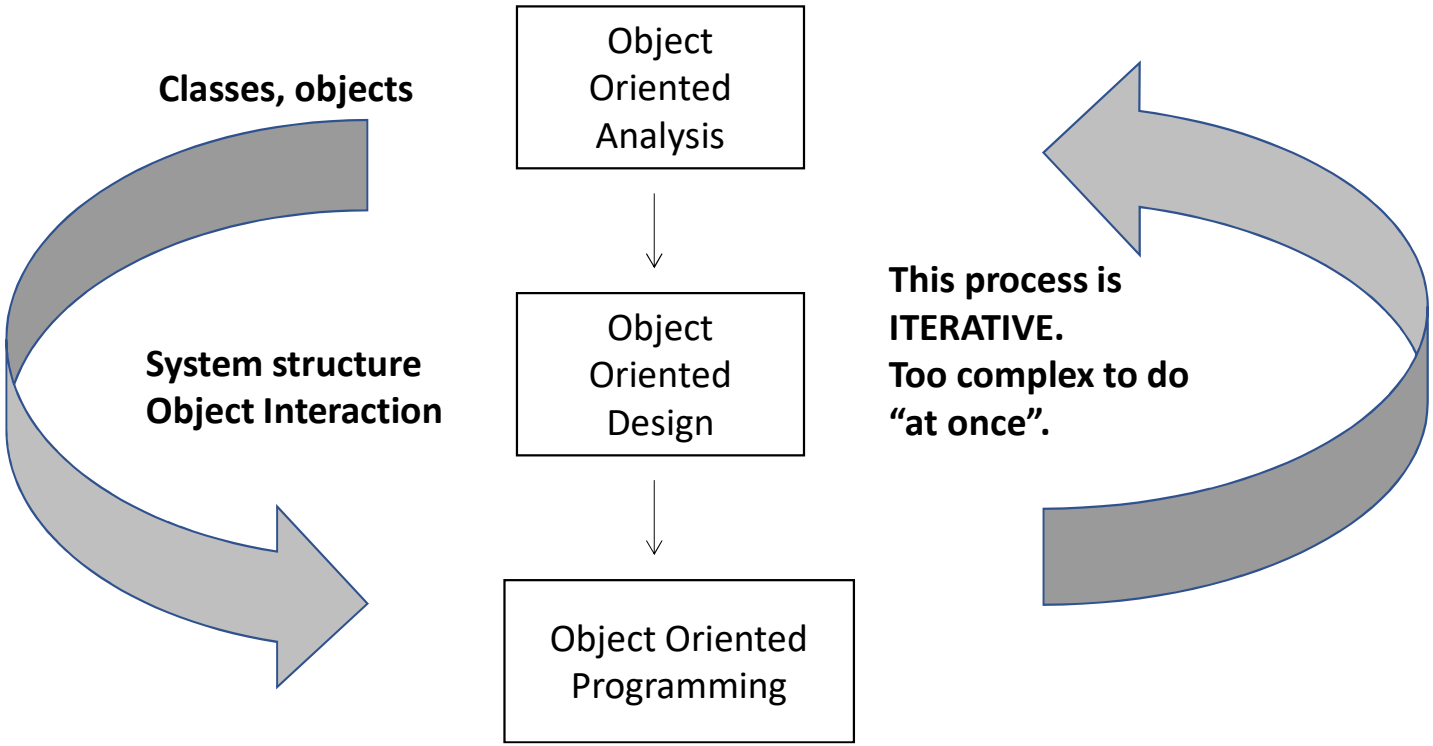
OO Models

- Simulate the logic, structure, object interactions and architecture of a "significant" OO system.
- Facilitate development of complex systems:
  - Classes, or packages of classes can be designed to be **reusable**
  - Functionality can be decomposed into pieces that are delivered on an **iterative and incremental basis**—a piece is developed, tested, deployed and, when it is stable, the next piece is added.
- *Trying to build a complex system in its entirety, from scratch, without iterating through stable system increments is simply not possible.*

Models in SYS466:

- Use case model
  - Use cases, descriptions and diagrams (Business and System)
  - System level sequence diagrams
- Domain model
  - Domain class diagrams: initial class definitions, attributes, relationships and multiplicity.
- Design model
  - Object level sequence diagrams
  - Design class diagrams—design level classes (reference associations, interfaces, etc.)

OO Analysis & Design: a high level view



Use Case Model

System Sequence Diagrams

Two Types of Sequence Diagrams

- **System Level Sequence Diagram**
  - Shows how actor interacts with the System as a black box
  - Shows the messages an actor sends to the system
  - Looks at ONE scenario in an Actor – Use case interaction
- Object level Sequence Diagram
  - Opens up the black box. Shows much more detail—how the objects interact with each other
  - We will look at this later in the course

SSDs and Use Cases

- Use cases describe how external actors interact with the software system…
  - **An actor generates system events/messages to a system, requesting some system operation to handle the event.**
  - **The use case text implies the event/message…the SSD makes it concrete and explicit.**

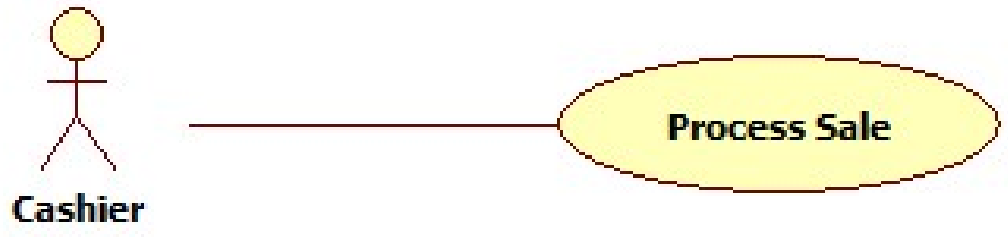
- Larman, APPLYING UML AND PATTERNS, p. 176

System Sequence Diagrams

- A system sequence diagram is a picture that shows, *for one particular scenario of a use case*, the events/messages that external actors generate, their order and the inter-system events.
- All systems are treated as a black box.

- Larman, APPLYING UML AND PATTERNS, p. 176

Use Case Diagram



Let’s take one scenario from process sale and look at it more closely.

Fig. 10.3

Actor (cashier)	System
Cashier starts a new sale	Displays a transaction entry area.
Cashier enters item identifier	System creates a line item record and retrieves and presents item description (including price) and running total for the transaction.
The above step is repeated until the cashier signals that the transaction is complete.	The system calculates total with taxes and presents the results.

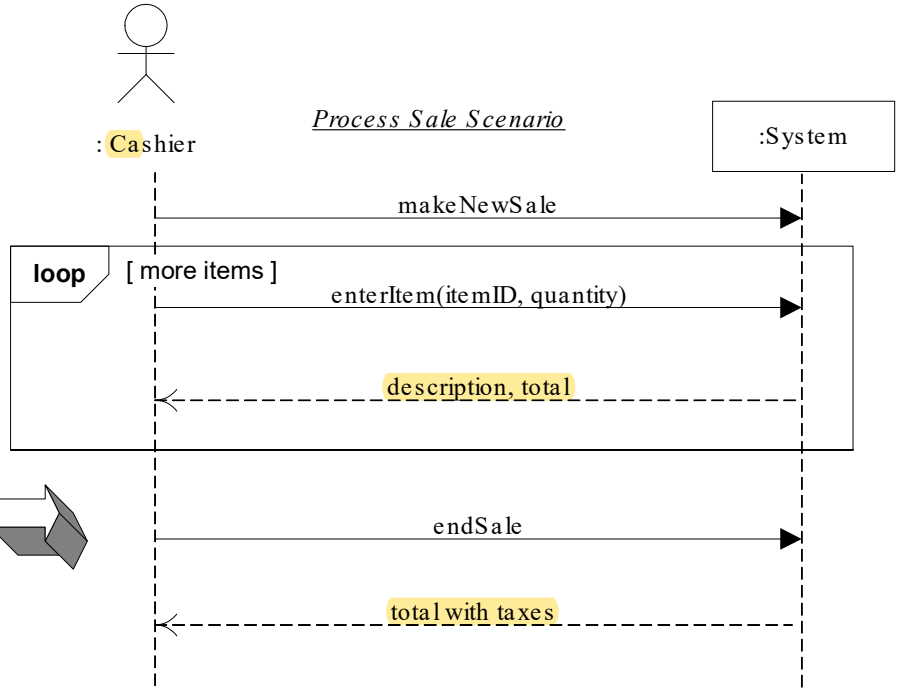
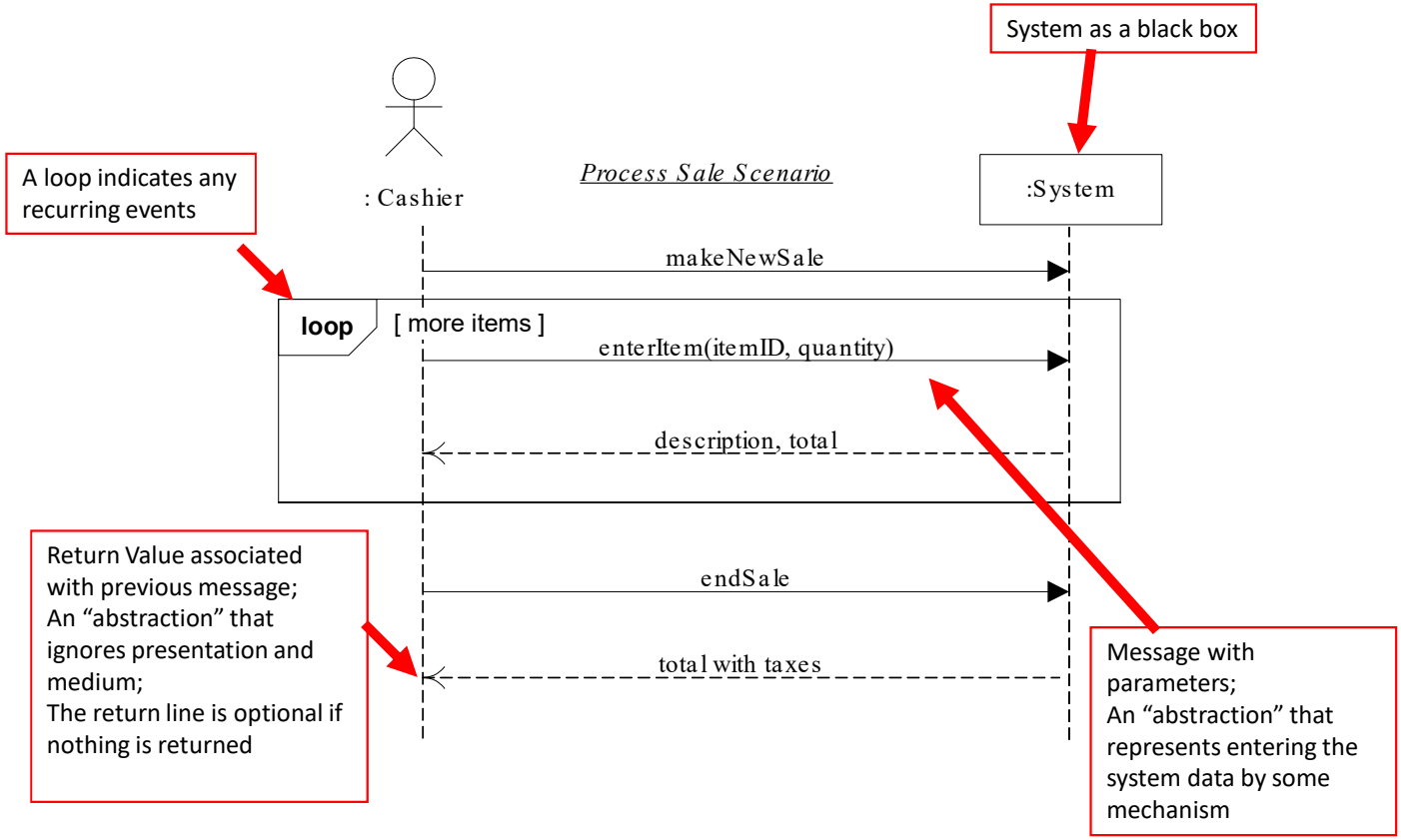


Fig. 10.3

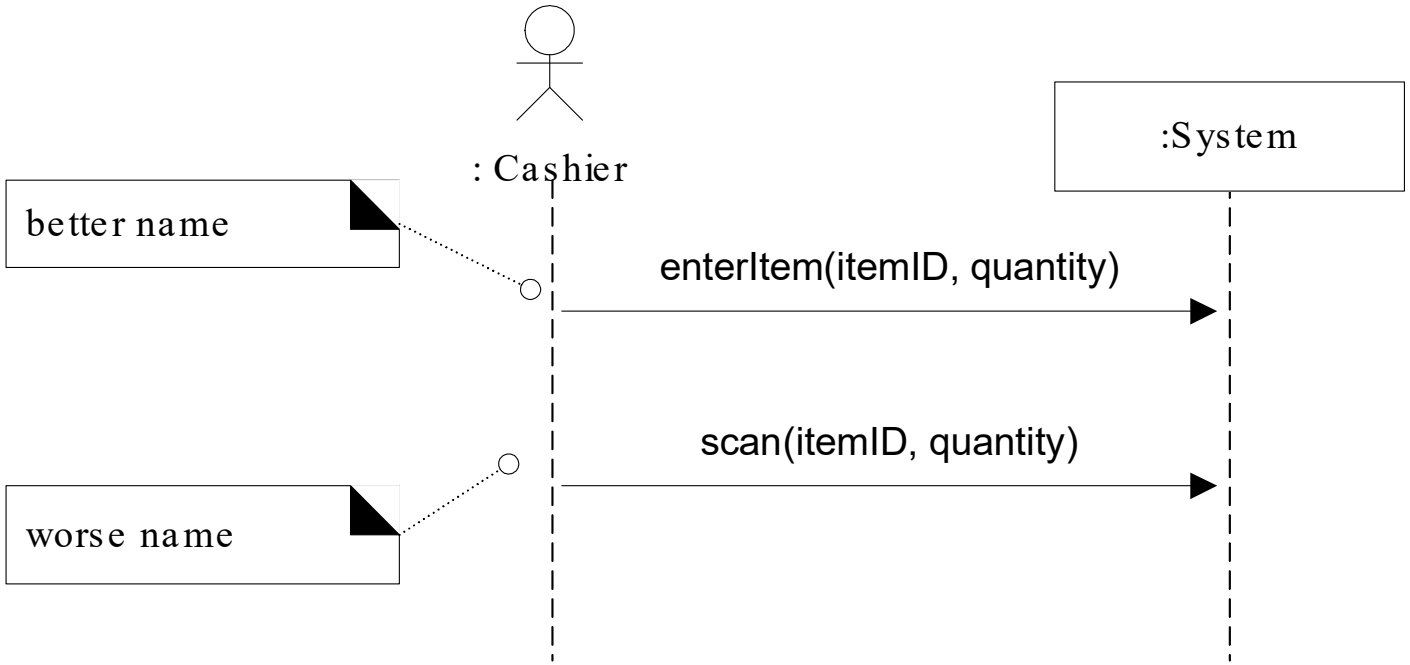

System as a black box
A loop indicates any recurring events
: Cashier
Process Sale Scenario
:System
makeNewSale
loop
[more items]
enterItem(itemID, quantity)
description, total
Return Value associated with previous message;
An “abstraction” that ignores presentation and medium;
The return line is optional if nothing is returned
endSale
total with taxes
Message with parameters;
An “abstraction” that represents entering the system data by some mechanism

System Sequence Diagrams

- System events should be expressed at the abstract level of intention rather than in terms of the physical input device.
- In other words—avoid UI

- Larman, APPLYING UML AND PATTERNS, p. 178

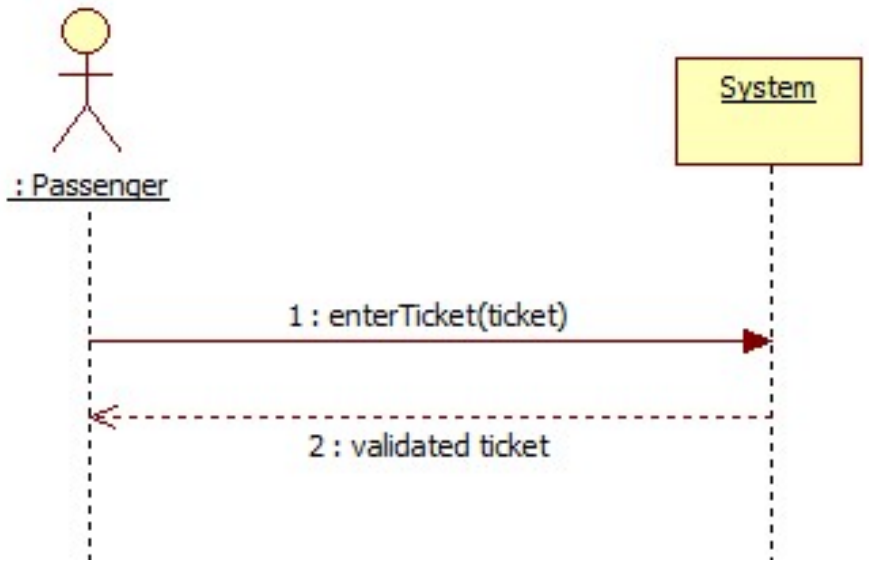
Fig. 10.4


: Cashier
:System
better name
worse name
enterItem(itemID, quantity)
scan(itemID, quantity)

Examples

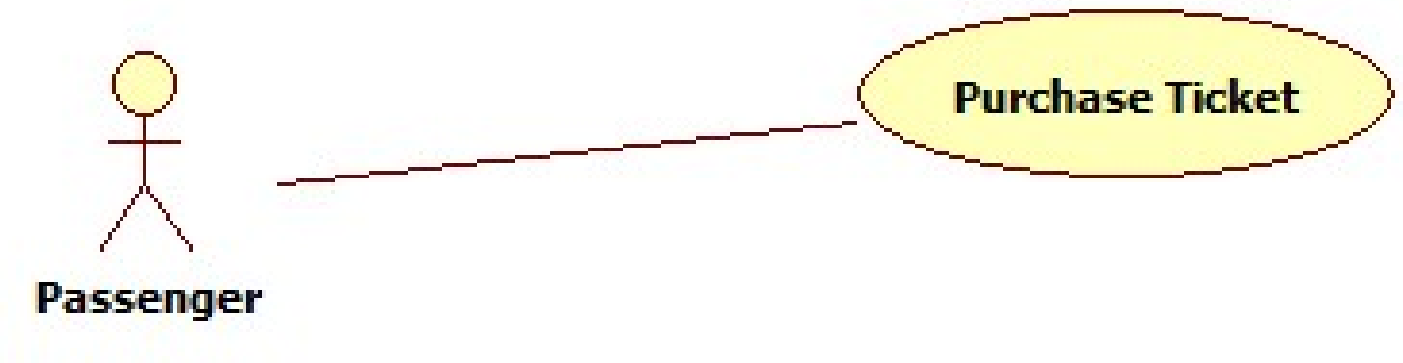
Validate Bus Ticket Use Case Diagram

Validate Bus Ticket SSD

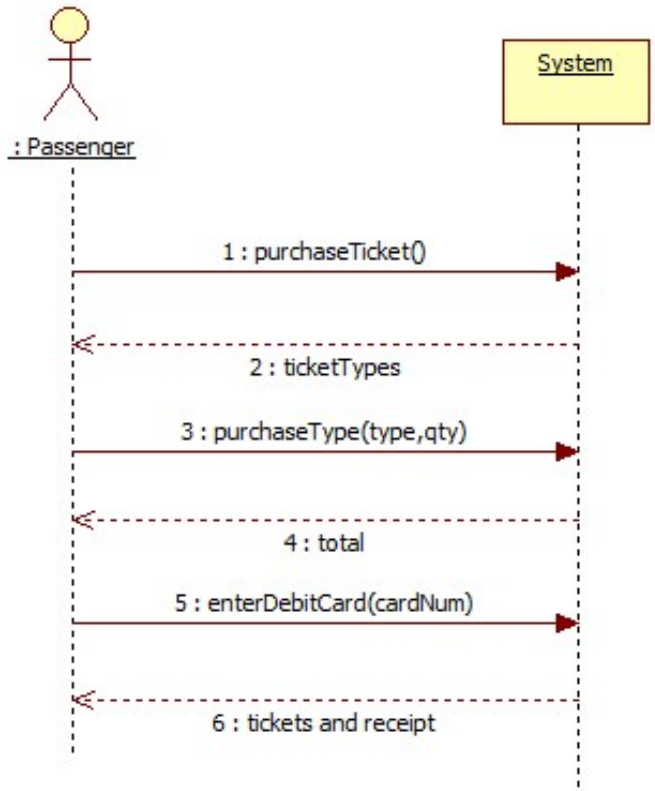


- Passenger inserts ticket into machine
- System validates ticket, cuts off a corner and returns ticket

Purchase Ticket Use Case Diagram

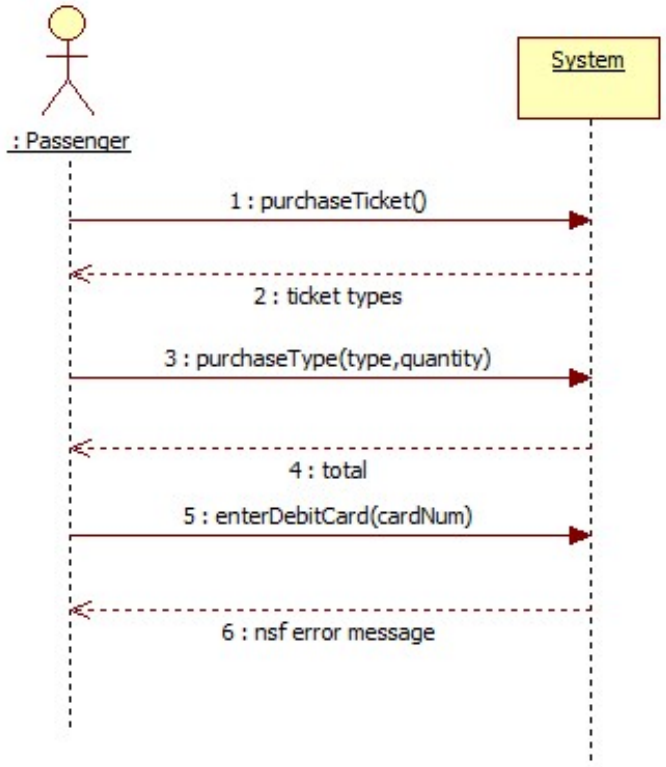


SSD: Purchase Ticket – Enough Funds



- Passenger requests to purchase ticket
- Machine displays ticket types
- Passenger selects ticket type and enters quantity
- System displays total and asks for debit card
- Passenger inserts debit card
- System validates card, creates debit transaction and sends it to the bank, and prints the tickets and receipts

SSD: Purchase Tickets – Insufficient Funds



- Passenger requests to purchase ticket
- Machine displays ticket types
- Passenger selects ticket type and enters quantity
- System displays total and asks for debit card
- Passenger inserts debit card
- System validates card: there are insufficient funds.
- Displays message and returns card. No ticket is issued.

Common SSD Errors

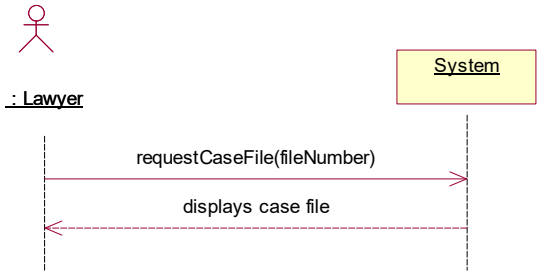
Return Errors

Most Common:

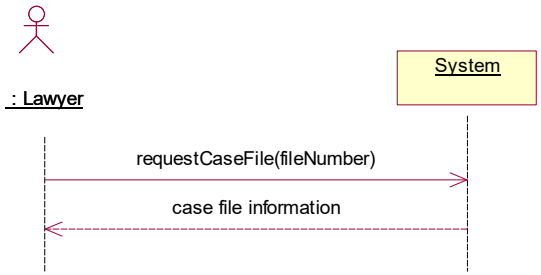
- Error:
  - return indicates some kind of action
- Correction:
  - return should only contain a list of data that is returned.

Examples:

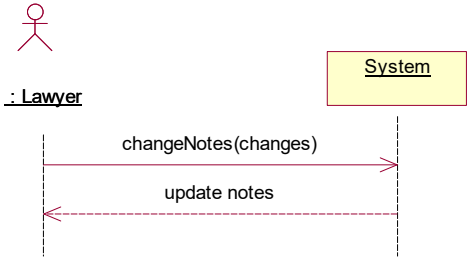
Error:



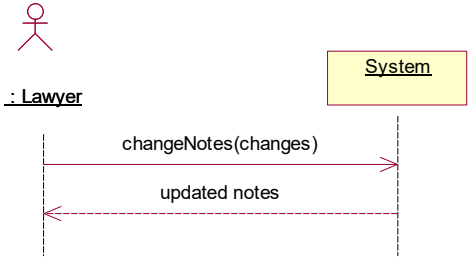
Correction:



Error:



Correction:



Message Errors

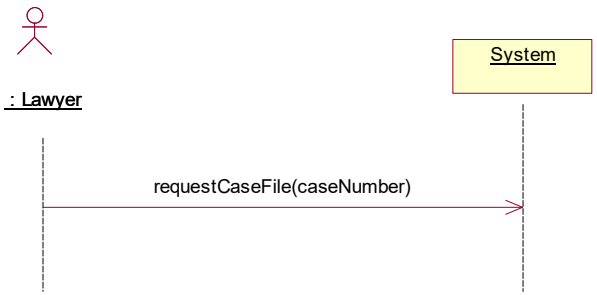
Most Common

- Error:
  - Not specifying arguments when input data is required
- Correction:
  - use arguments

Example

Error:

Correction:



Coverage Errors

Most Common

- Error:
  - Trying to fit many scenarios into one SSD
- Correction:
  - One SSD per scenario

Example: Three scenarios

Scenario: Change Case Notes:

Actor	System
Enters case file number	Returns case file information including notes
Changes a line of notes	Updates notes and shows changed notes

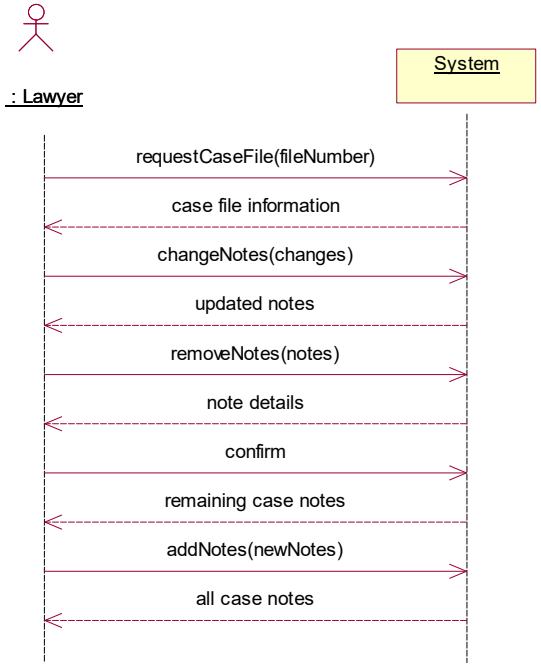
Scenario: Add Case Notes:

Actor	System
Enters case file number	Returns case file information including notes
Adds a new line of notes.	Adds notes to case file and shows all case file notes

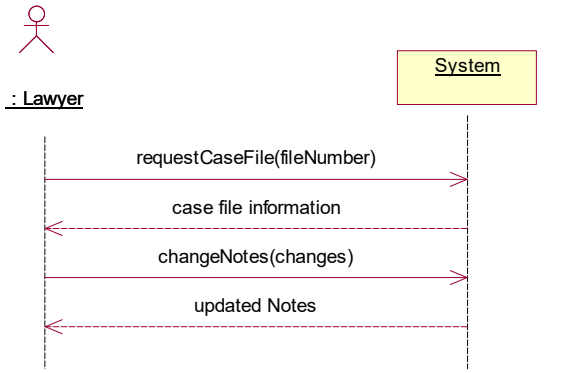
Scenario: Remove Case Notes:

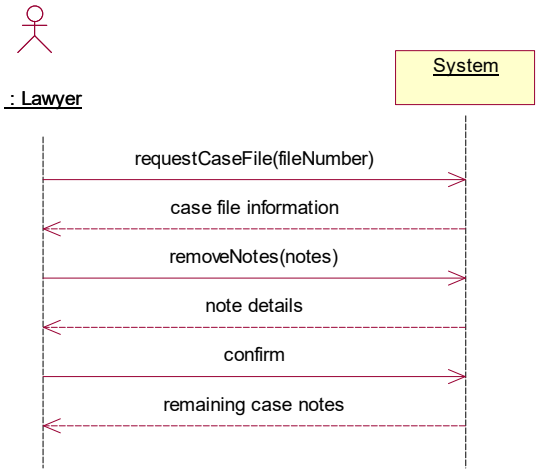
Actor	System
Enters case file number	Returns case file information including notes
Requests to remove a line of notes.	Requests confirmation.
Confirms	Removes notes from the case file and shows all case file notes.

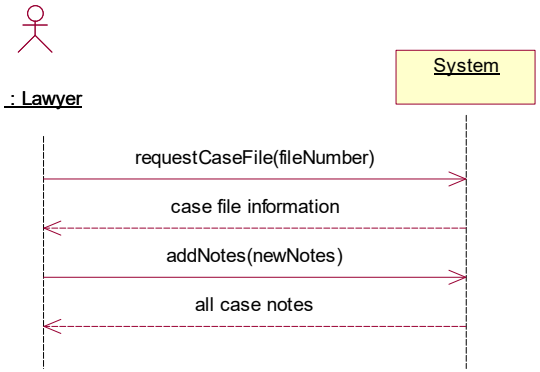
Error: Putting them all into one SSD



Correction: Three SSDs







In-class SSD Exercises

Tasty Cakes

- Tasty Cakes sells all kinds of baked goods—cakes, cookies, pies, pastries, muffins, donuts, and so on. Its customers are hotels, businesses, wedding planners and so on.
- For each of the given scenarios, create a system sequence diagram

Exercise 1:
Add New Business Customer Account

Actor (Manager)	System
Requests to add a business customer and enters name, address, phone number.	Validates the entered information to make sure the customer is not a duplicate customer (it is not) and displays an entry form for contact information.
Enters a contact name, contact email address, contact phone.	Makes sure the entered contact is not a duplicate of any contacts that might already exist for this customer (it is not). Adds the contact information to the customer. Displays entry form for next contact information.
Repeats the above row until requests to finish.	Saves the new customer and contact information to the database.

Exercise 2:
Make an Address Correction

Actor (Admin)	System
Requests to see customer “Ace Planners”	Retrieves and displays business name, address, phone and contact name for Ace Planners.
Changes “355B Dundas E.” to “355C Dundas E.” and requests to save the information	Valildates the entered address (for illegal characters) and saves the new address to the database.

Exercise 3:
View Bakery Products

Actor (Manager)	System
Requests to see a list of Tasty Cakes product categories.	Displays a list of product categories showing the name of each.
Selects the category cookie and requests to see a list of all cookies that Tasty Cakes makes and sells.	Retrieves and displays a list of cookies—for each cookie the ID and name are displayed.
Selects the "Champion Chocolate Chip" cookie.	Displays the cookie description and a list of ingredients.