System Sequence Diagrams

Overview

- What is System Sequence Diagram?
- UML Sequence Diagram
- Case Study: Simplified "Process Sale"

System Sequence Diagram

- Definition
  - A picture that shows, *for* a use case, the events that external actors generate, their order, and inter-system events
    - Happy path + frequent/complex alternatives
- All systems are treated as a black box, focusing on WHAT instead of HOW

Compared with Class Diagram

- Class Diagram describes the static structure of software
- Sequence Diagram describes the dynamic interactions between actors and the system

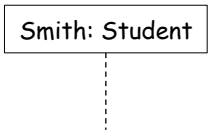
Roles of SSDs

- Generated from inspection of a use case
  - Illustrate input and output events related to the system
  - Emphasize events cross the boundary between actors and systems
- Input to OOD

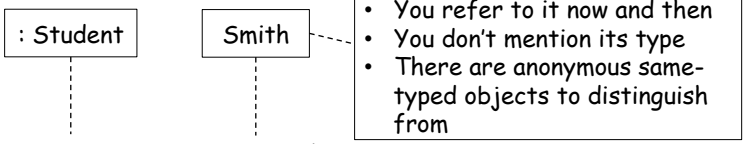
UML Sequence Diagram

- A notation to illustrate actor interactions and operations initiated by them
- Only the interaction between users and the system is modeled in system sequence diagram

Legends: Lifeline

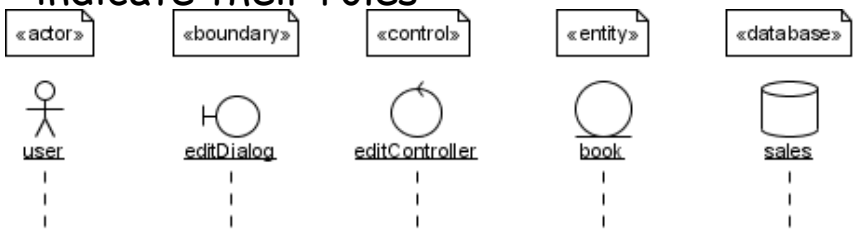


- Definition
  - Represents either actors or systems that participate by either sending or receiving messages (events)
- Naming convention
  - Instance Name: Class Name
  - Other variants

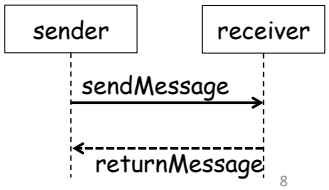


Legends: Note, Stereotype, Messages

- Stereotypes can be added to objects to indicate their roles



- Messages represent events



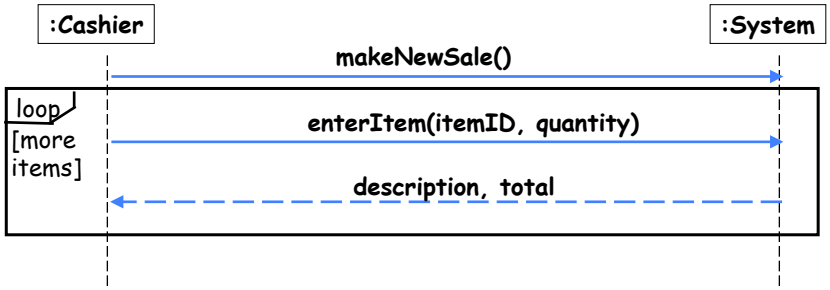
Legends: Combined Fragment

- Definition
  - An interaction fragment which defines a combination of messages between objects
  - Interaction operator(relation) + interaction operands (messages) + interaction constraints (guards)
  - Operators
    - loop – iteration
    - alt – alternatives
    - opt - option

Example: Simplified "Process Sale"

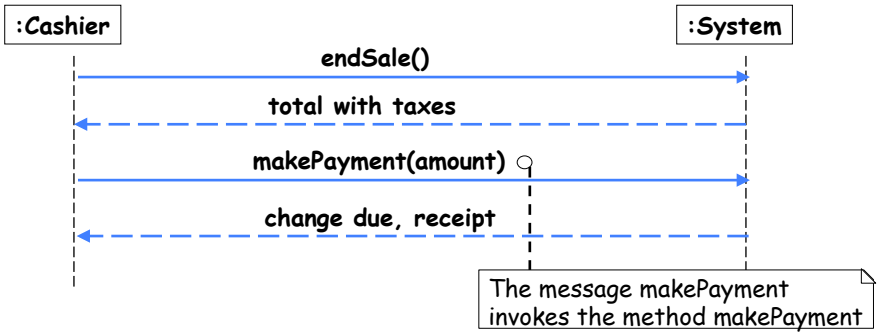
1. Cashier starts a new sale
2. Cashier enters item id
3. System records sale line item and presents description and running total

Repeat Steps 2-3 until done



Example cont.

4. System presents total with taxes calculated.
5. Customer pays and System handles payment



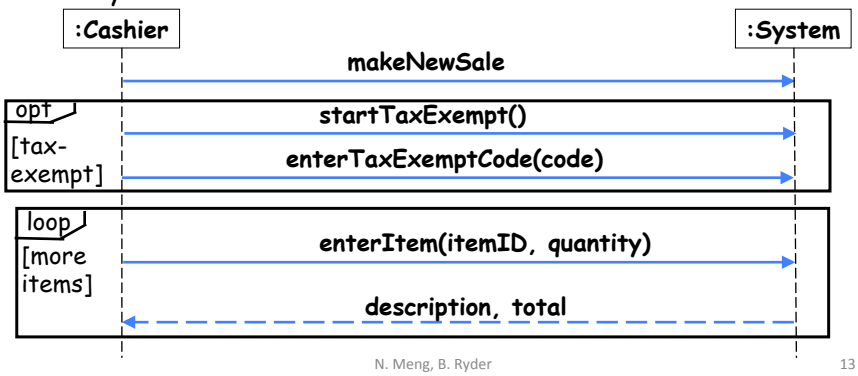
Abstractions in SSDs

- Events and return values are abstractions
  - Independent of mechanism & representation
- *makePayment(amount)*
  - Shows input info
  - Looks like a method call, but is really an abstraction of an event
- Name: should capture the intent
  - Avoid specifying implementation choices
    - *enterItem(itemID)* is better than *scan(itemID)*

Alternative Scenario

1a. Customer tells Cashier they have a tax-exempt status (e.g., seniors, native people)

1. Cashier verifies, and then enters tax-exempt status code
2. System records status



Homework: Withdraw Money from ATM

- Due: 10/07/2015 11:59pm