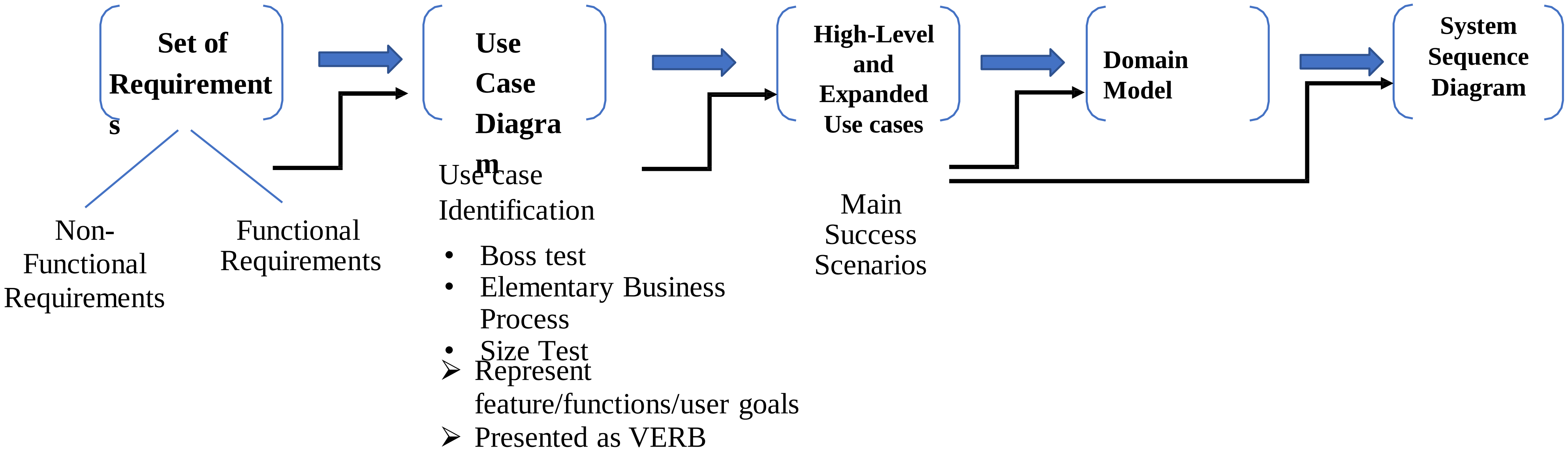
Software Design and Analysis
CS-3004
Lecture#07

Dr. Javaria Imtiaz,
Mr. Basharat Hussain,
Mr. Majid Hussain

quest lab

Revision up till now



Set of Requirements

- Non-Functional Requirements
- Functional Requirements

Use Case Diagram

Use case Identification

- Boss test
- Elementary Business Process
- Size Test

➢ Represent feature/functions/user goals

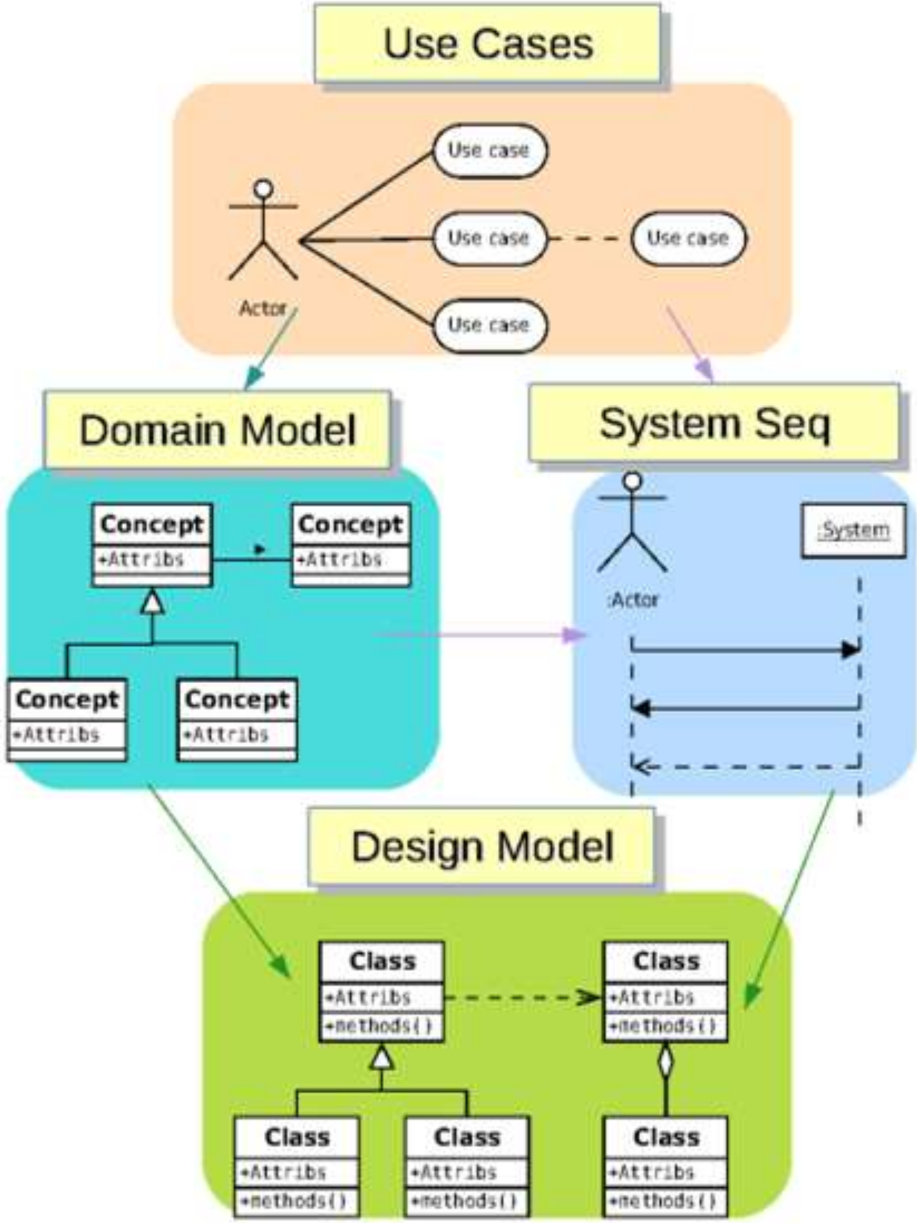
➢ Presented as VERB

High-Level and Expanded Use cases

Main Success Scenarios

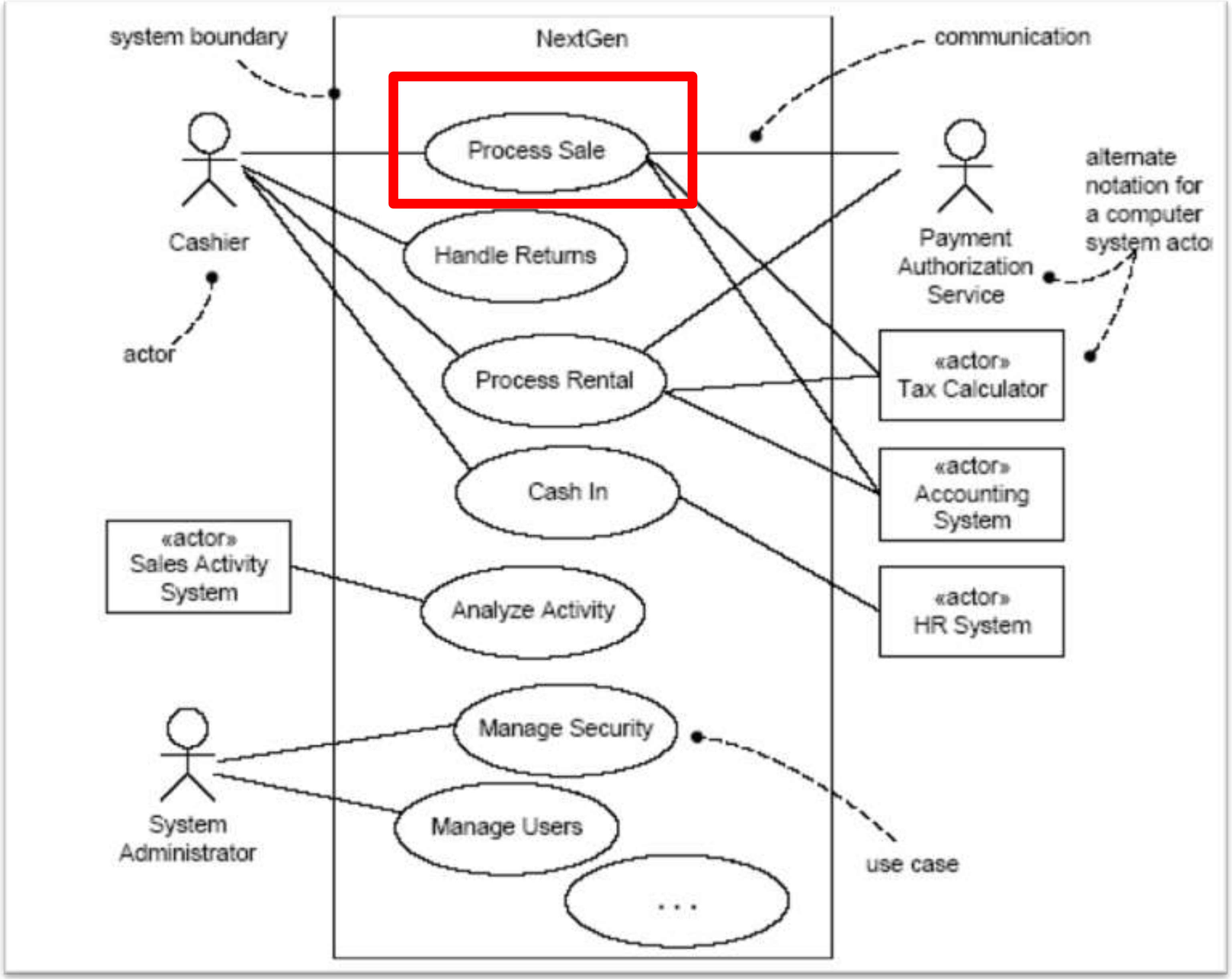
Domain Model

System Sequence Diagram


Use Cases
Actor
Use case
Use case
Use case
Use case
Domain Model
Concept
+Attribs
Concept
+Attribs
Concept
+Attribs
Concept
+Attribs
System Seq
:Actor
:System
Design Model
Class
+Attribs
+methods()
Class
+Attribs
+methods()
Class
+Attribs
+methods()
Class
+Attribs
+methods()
Class
+Attribs
+methods()

1. Use Case Diagram

Point of Sale



2. Brief Use Case

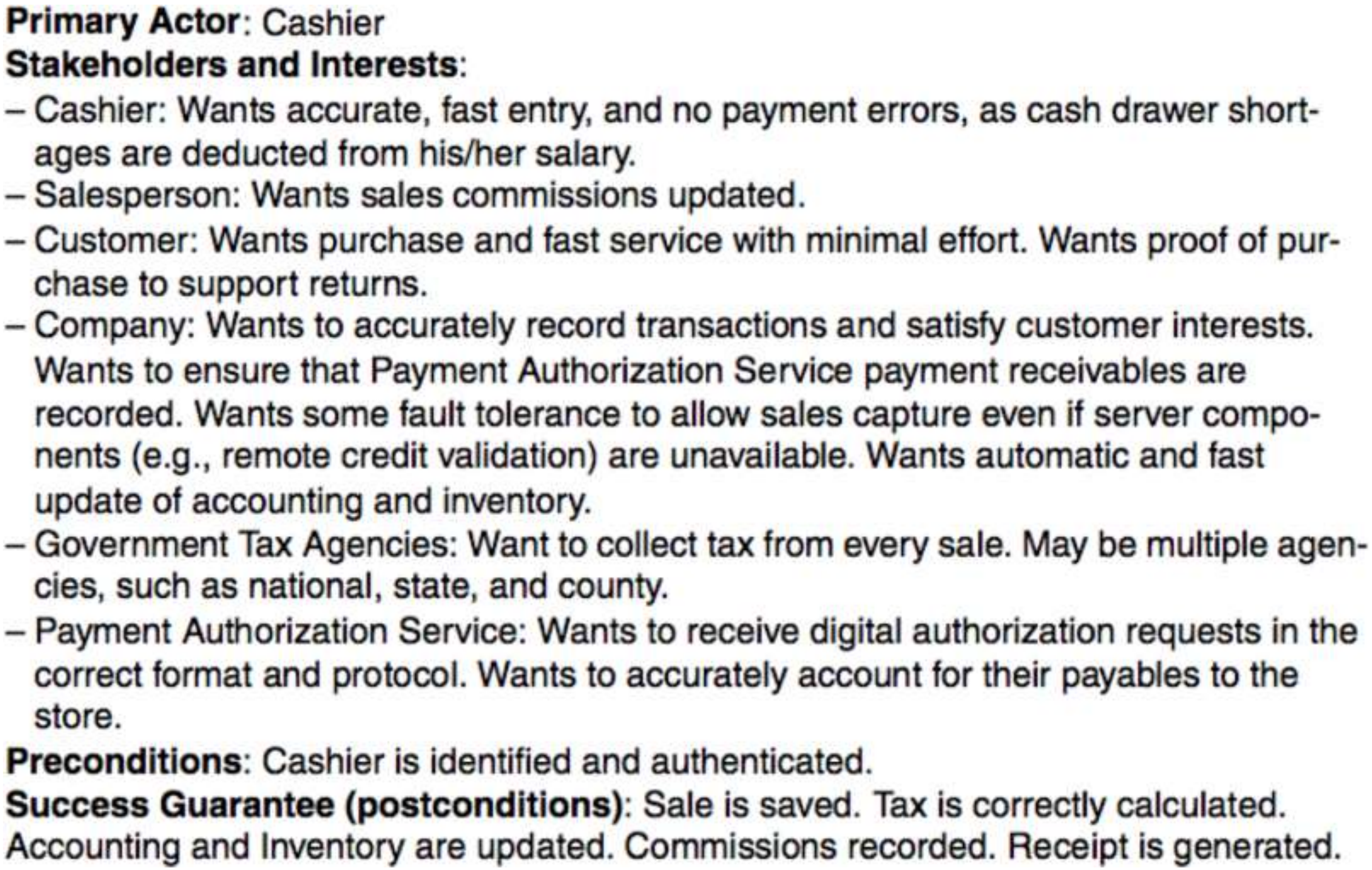
Point of Sale

- **UC01: Use Case:** Process Sale
- **Actor:** customer, cashier
- **Type:** Primary
- **Description:** A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

3. Fully dressed Use Case

Use Case UC1: (Process Sale)

Point of Sale



- **Primary Actor**: Cashier
- **Stakeholders and Interests**:
  - Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
  - Salesperson: Wants sales commissions updated.
  - Customer: Wants purchase and fast service with minimal effort. Wants proof of purchase to support returns.
  - Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
  - Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
  - Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
- **Preconditions**: Cashier is identified and authenticated.
- **Success Guarantee (postconditions)**: Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.

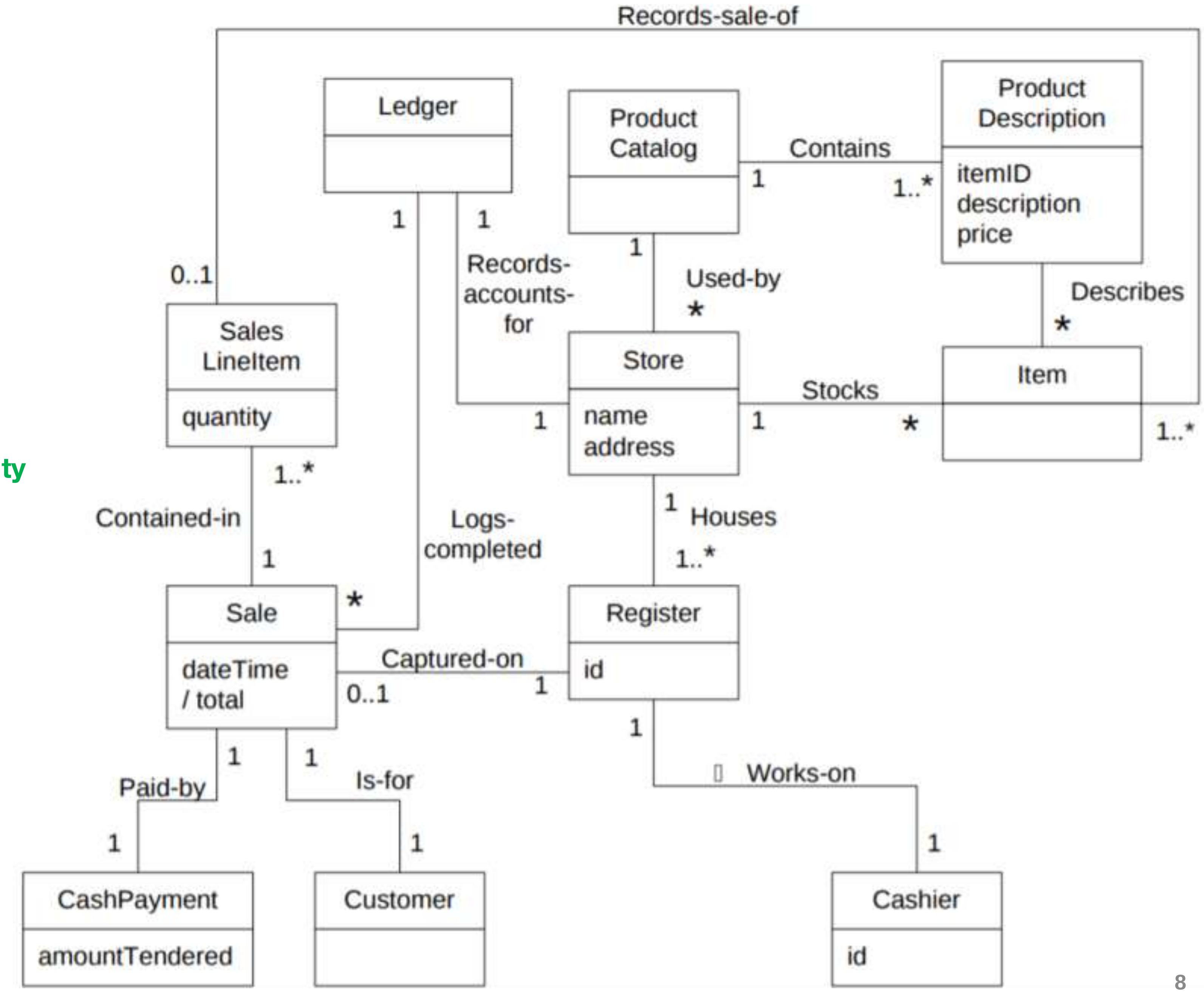
Main Success Scenario

1. Cashier starts new sale
2. Cashier enters item identifier
3. System records sale line item and presents item description, price and running total
   - Steps 2 and 3 are repeated until all items are processed.
4. System presents total with taxes calculated
5. Cashier tells Customer the total and asks for payment
6. Customer pays and System handles payment

4. Domain Model

Point of Sale

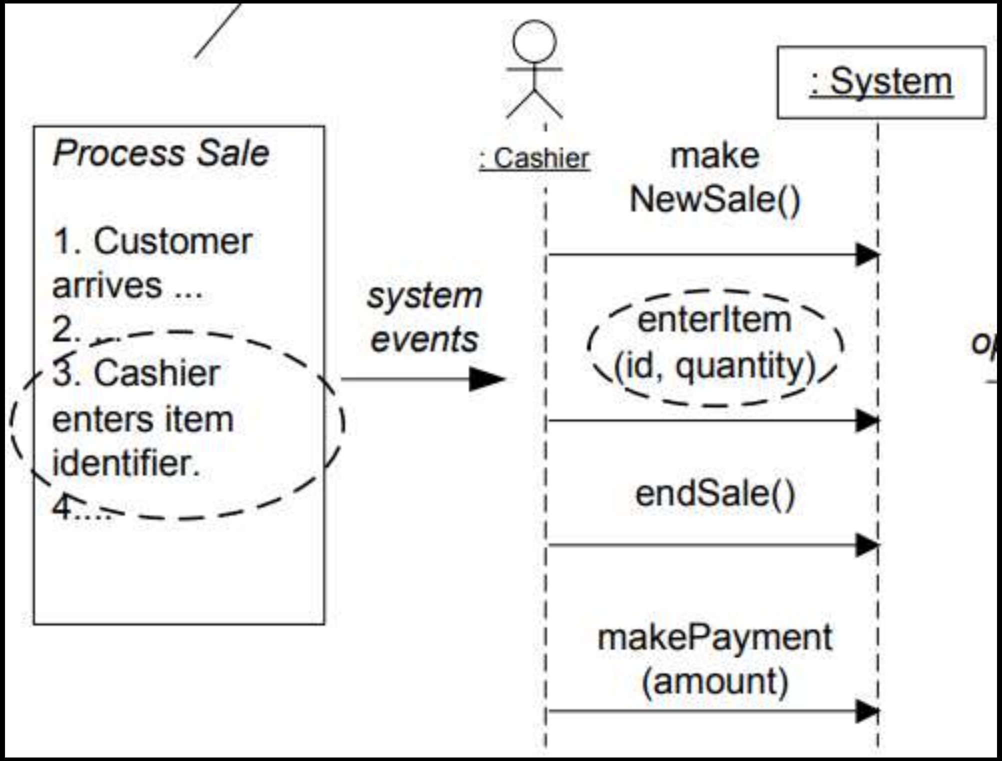
1. Is there any register?
2. Is there any actor?
3. Is there any service or product?
4. Is there any physical place where activity has been performed?
5. Is there any report?
6. Is there any payment mechanism?
7. Is there any description class?



A time for systems analysis

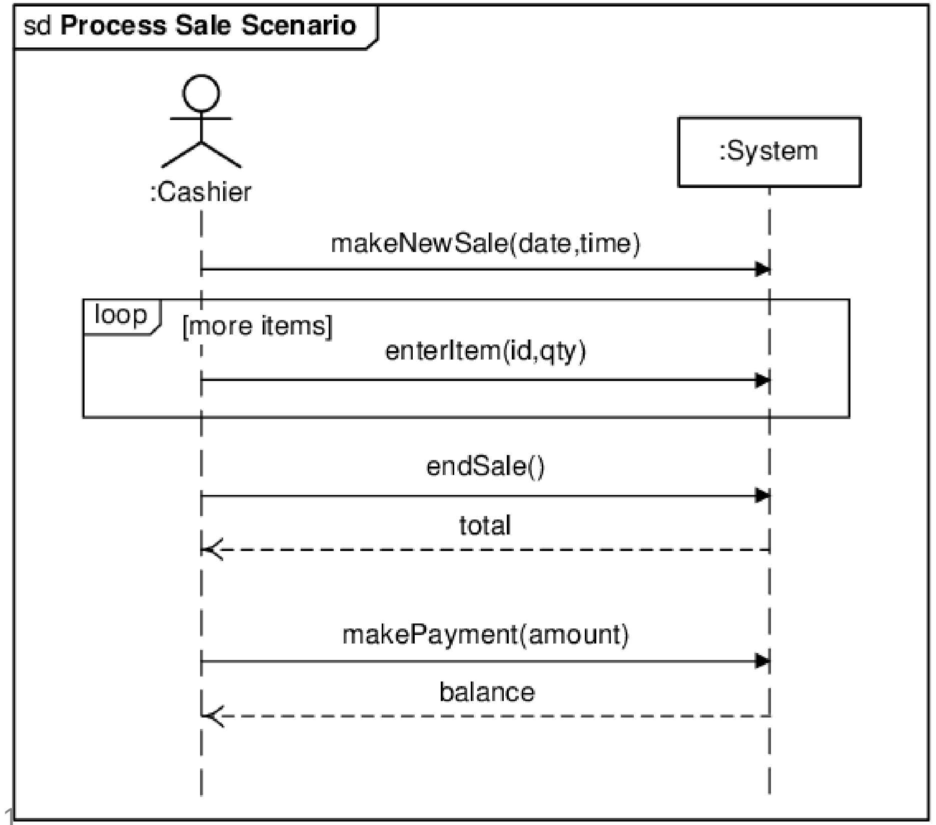
- Next comes systems analysis:
  - Clarifies input and output System events
  - Generates System Sequence Diagrams from Use Cases

System Sequence Diagram



Main Success Scenario

1. Cashier starts newsale
2. Cashier enters itemidentifier
3. System records sale line item and presents item description, price and runningtotal
   - Steps 2 and 3 are repeated until all items are processed.
4. System presents total with taxescalculated
5. Cashier tells Customer the total and asks for payment
6. Customer pays and System handlespayment



sd **Process Sale Scenario**

:Cashier

:System

makeNewSale(date,time)

loop [more items]

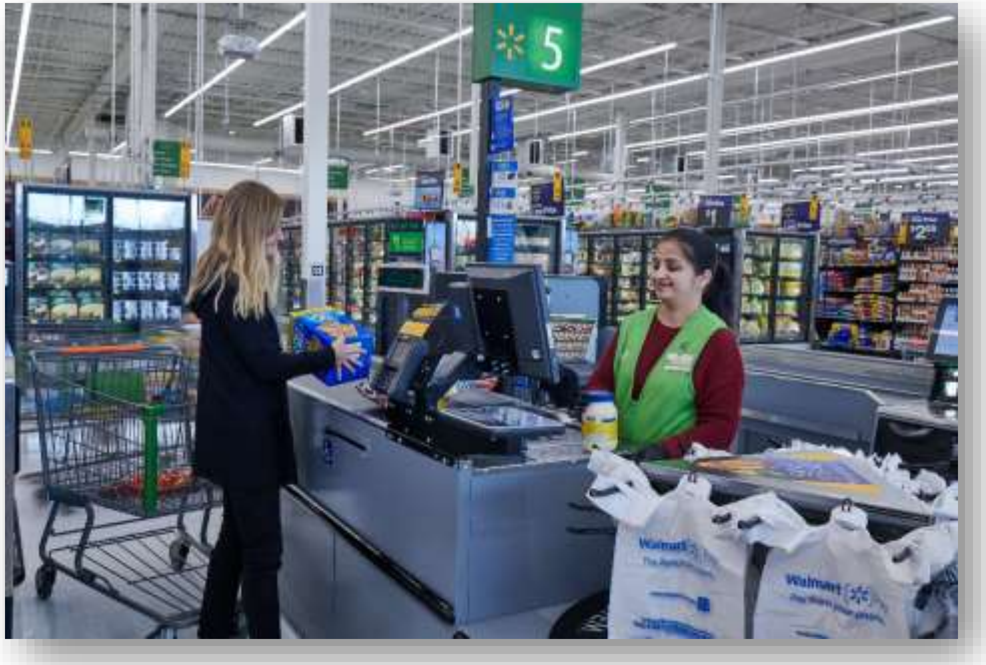
enterItem(id,qty)

endSale()

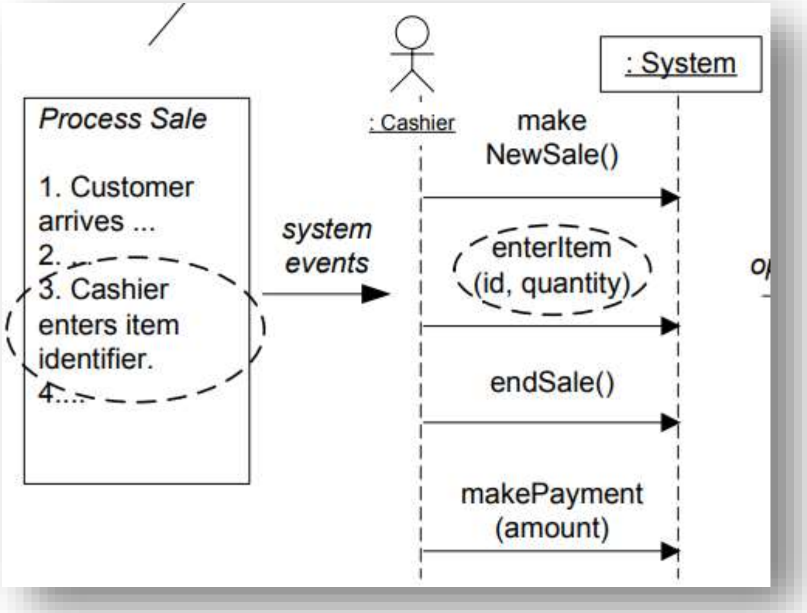
total

makePayment(amount)

balance



Interaction between cashier and system to perform process Sale



Process Sale

1. Customer arrives ...
2. ...
3. Cashier enters item identifier.
4....

system events

: Cashier

: System

make NewSale()

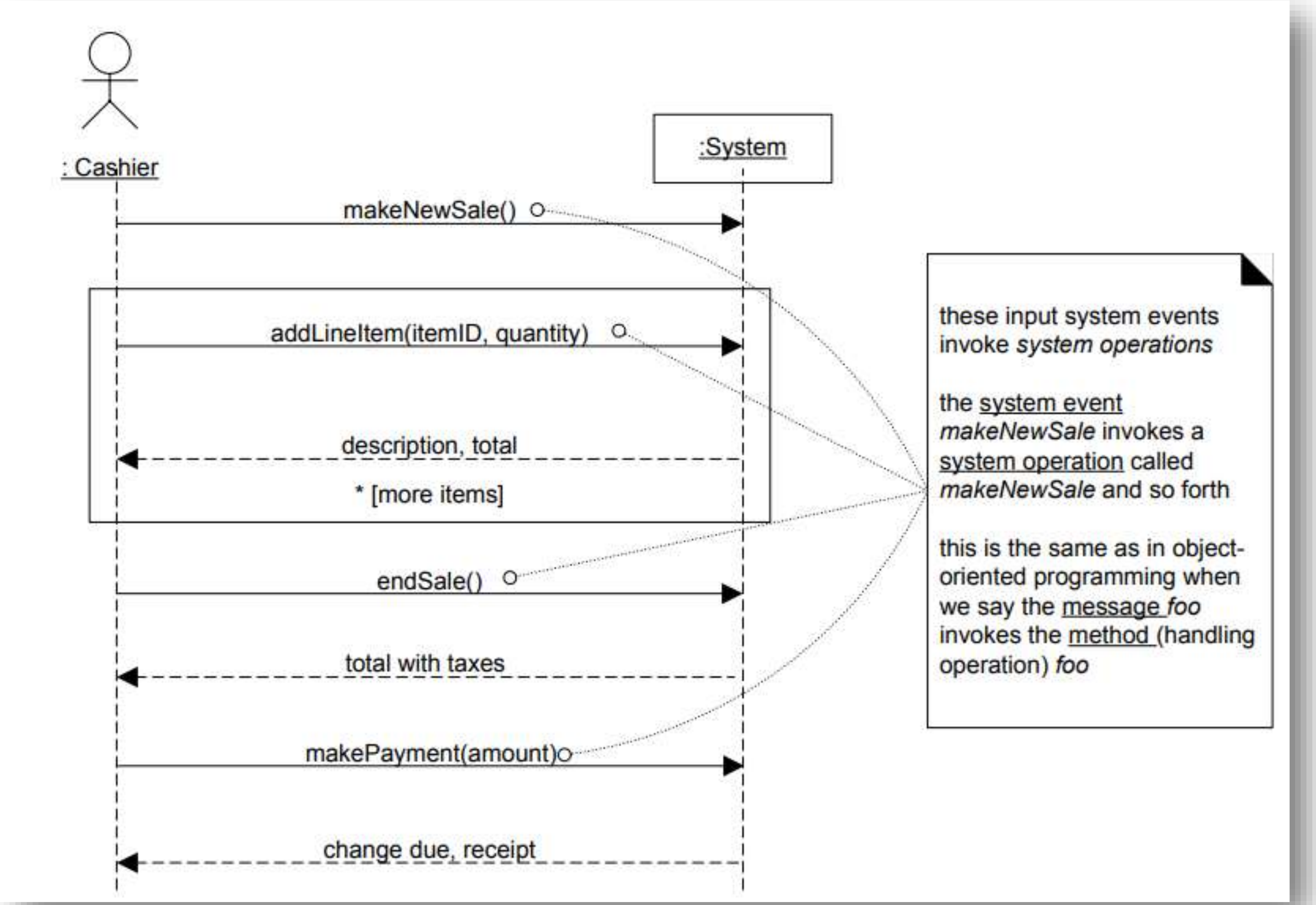
enterItem (id, quantity)

endSale()

makePayment (amount)

System Sequence Diagram

- An SSD shows – for one particular scenario of a use case –
  - the events that external actors generate,
  - their order, and
  - inter-system events
- The system is treated as a black-box (no implementation details).
- A description of “What” system does with some time aspects.
- SSDs are derived from use cases; SSDs are often drawn for the main success scenarios of each use case and frequent or complex alternative scenarios.
- SSDs are used as input for object design



these input system events invoke *system operations*

the system event *makeNewSale* invokes a system operation called *makeNewSale* and so forth

this is the same as in object-oriented programming when we say the message *foo* invokes the method (handling operation) *foo*

System Events and System Operations

- System operations are the operations that the system as a black box component offers in its public interface. These are high-level operations triggered by an external input event / system event generated by an external actor

- During system behavior analysis, system operations are assigned to a conceptual class **System**

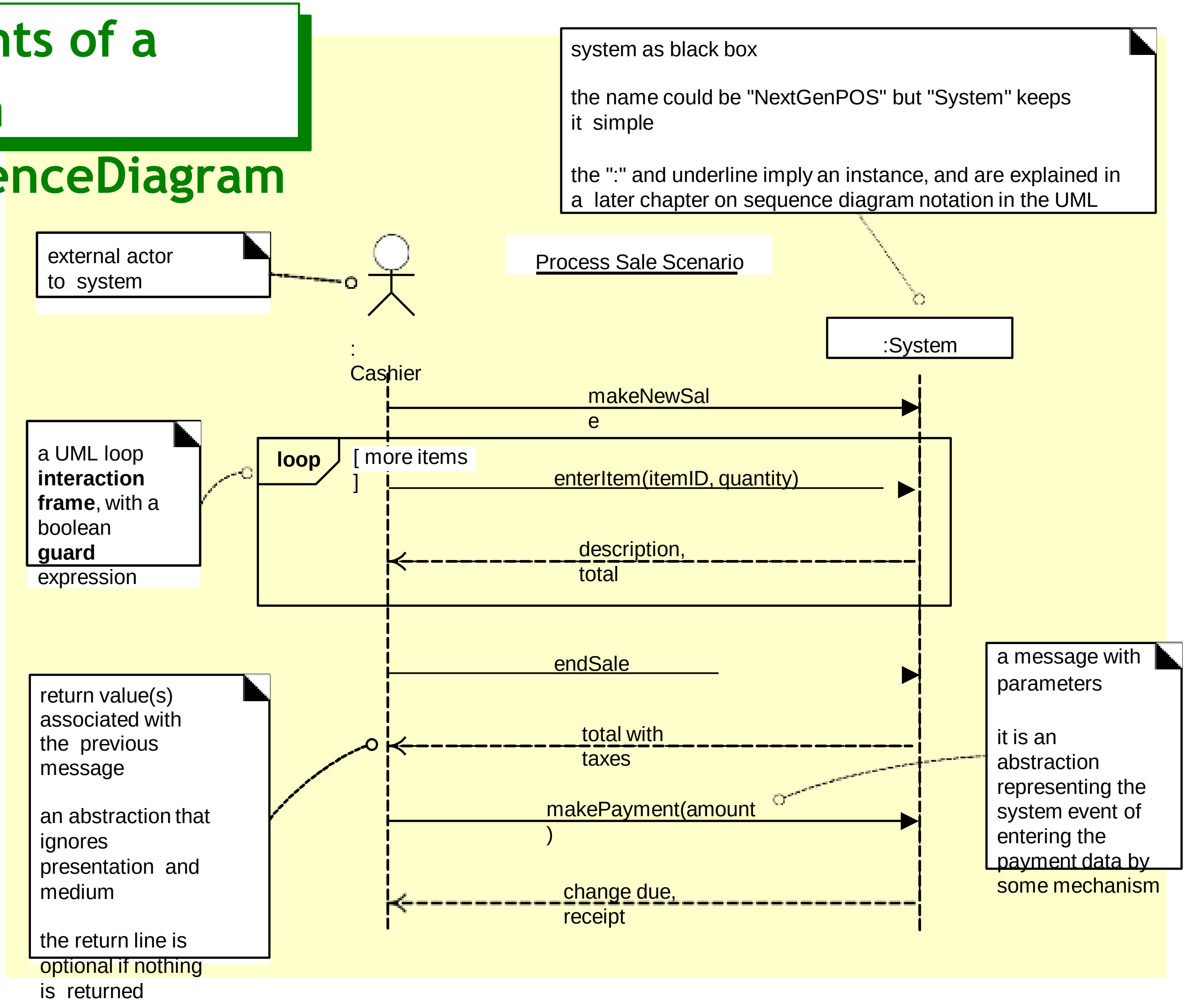
System Sequence Diagram

- A system sequence diagram shows the interaction between an actor and the system for one use case scenario.

- It shows:
  - The system (as a black box)
  - The initiating actor
  - Each external system which sends messages to the system
  - The messages into and out of the system
  - The sequence in which the messages occur

System Sequence Diagram

Elements of a System SequenceDiagram



Process Sale Scenario

external actor to system

system as black box

the name could be "NextGenPOS" but "System" keeps it simple

the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML

a UML loop **interaction frame**, with a boolean **guard** expression

return value(s) associated with the previous message

an abstraction that ignores presentation and medium

the return line is optional if nothing is returned

a message with parameters

it is an abstraction representing the system event of entering the payment data by some mechanism

1. Actor
2. Lifeline
3. System
4. Messages
5. Parameters
6. Actions
7. Responses
8. Loop

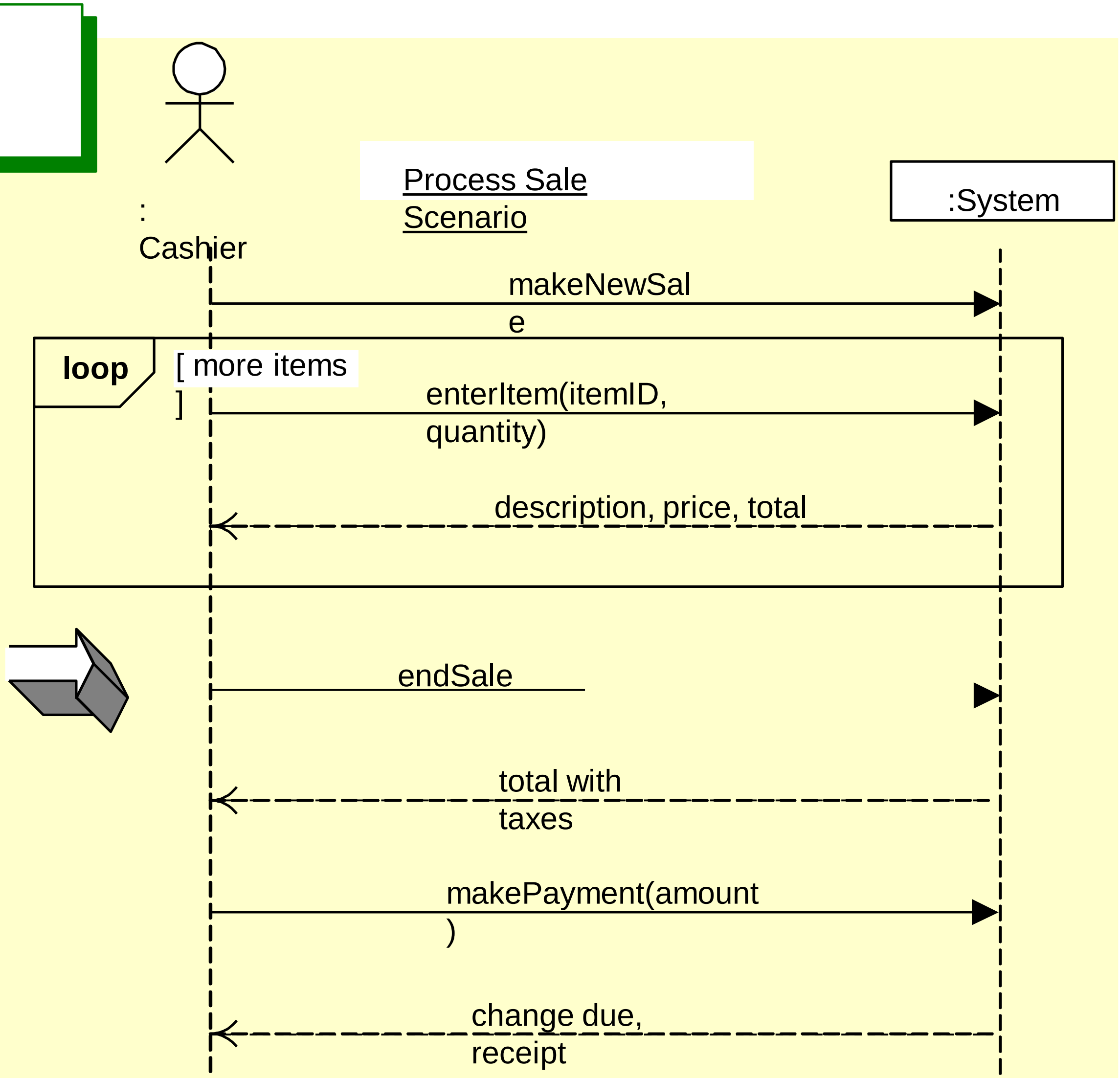
System Sequence Diagram

Mapping from Use Case to System Sequence Diagram

Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier and quantity.
4. System records sale line item and presents item description, price, and running total.
5. Cashier repeats steps 3-4 until indicates done.
6. System presents total with taxes calculated.
7. Cashier tells Customer the total, and asks for payment.
8. Customer pays and System handles payment.

...

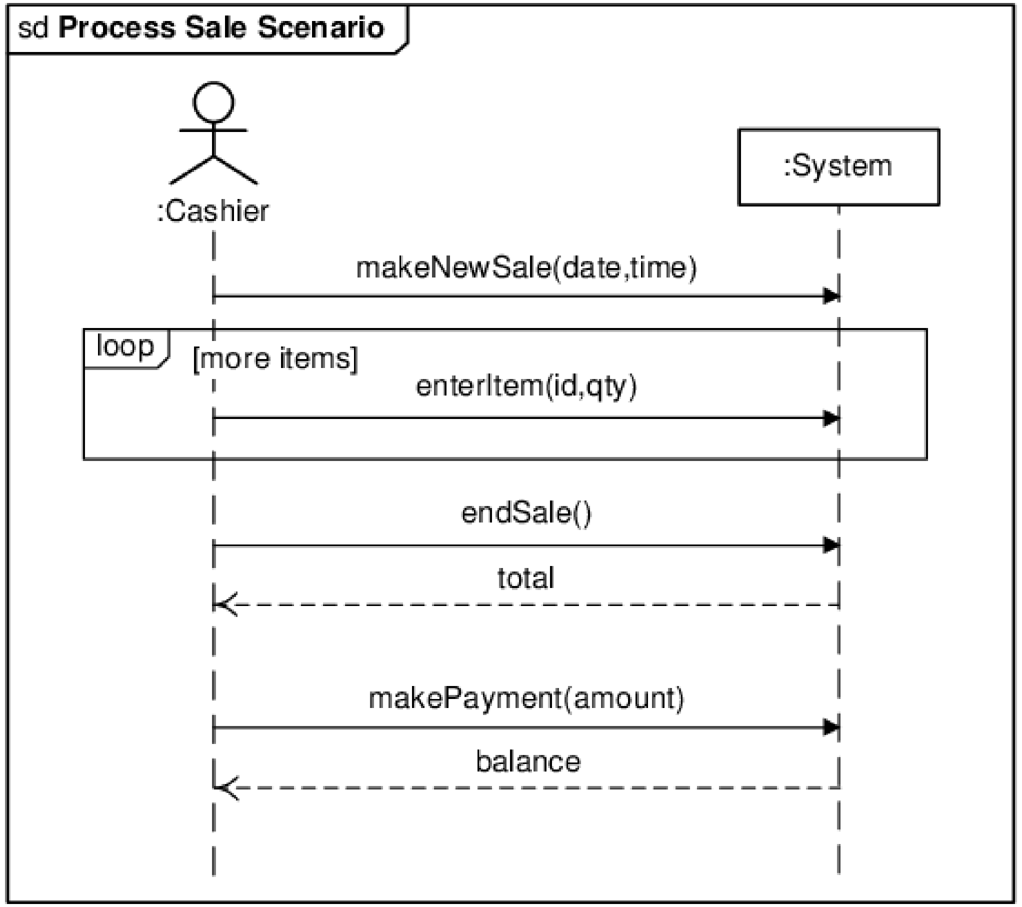


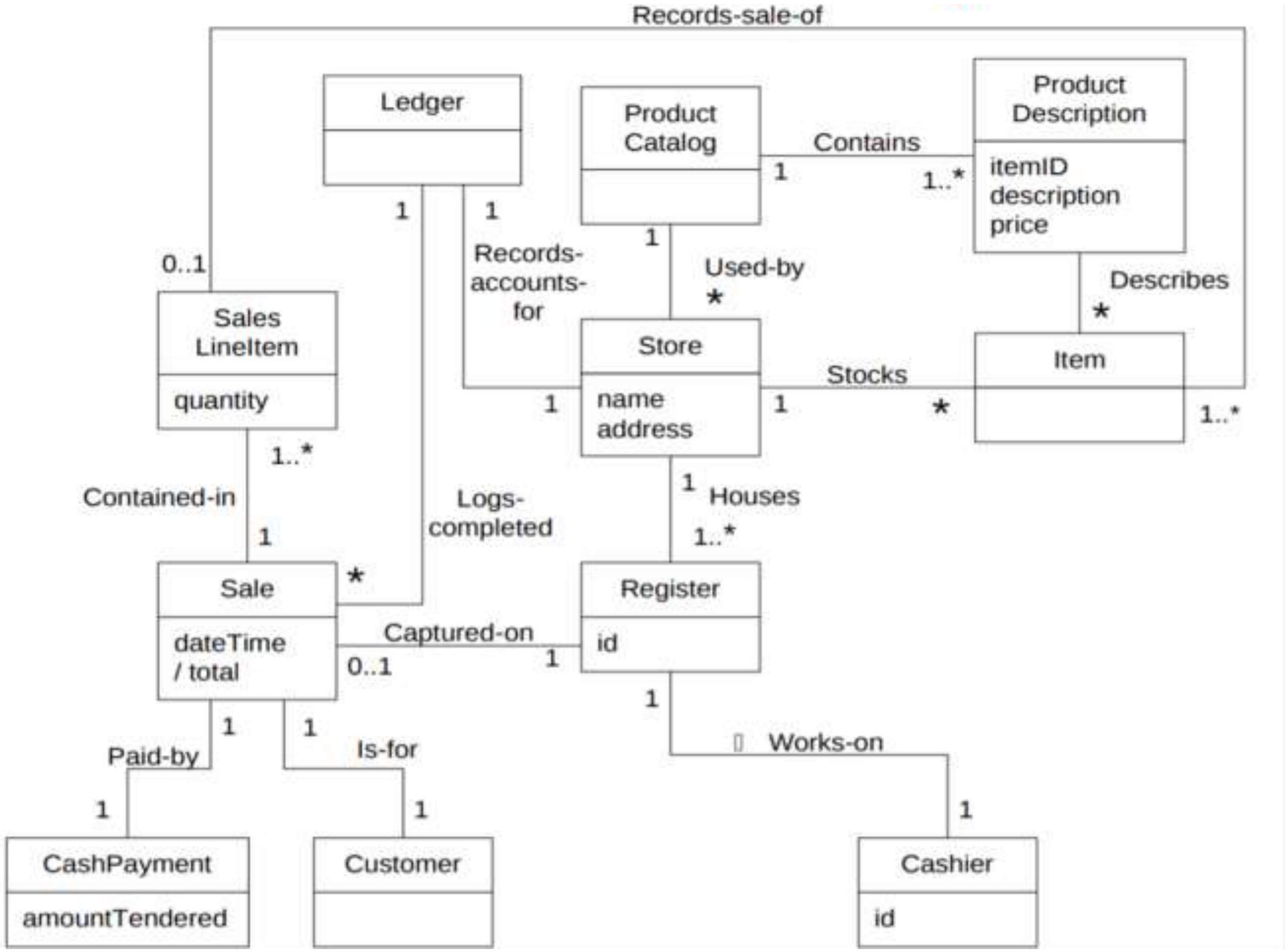
System Events and System Operations

- System operations are the operations that the system as a black box component offers in its public interface.

- These are high-level operations triggered by an external input event / system event generated by an external actor

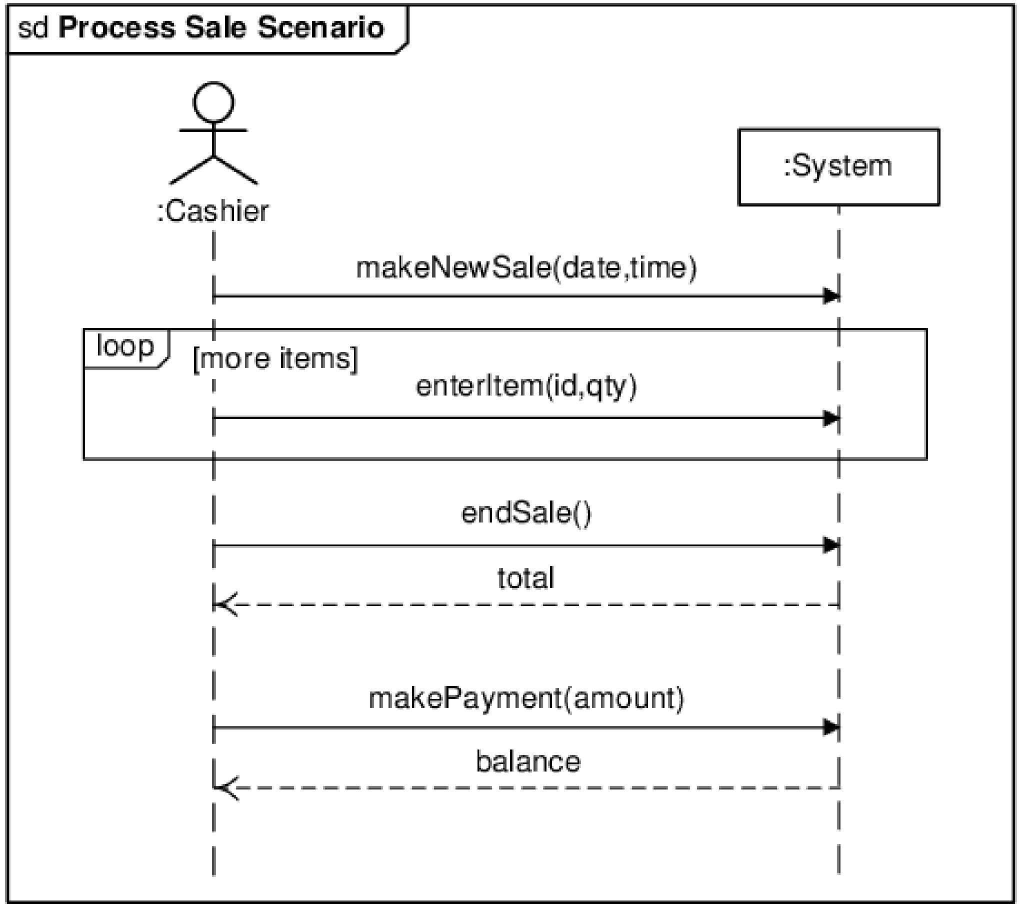
- During system behavior analysis, system operations are assigned to a conceptual class System

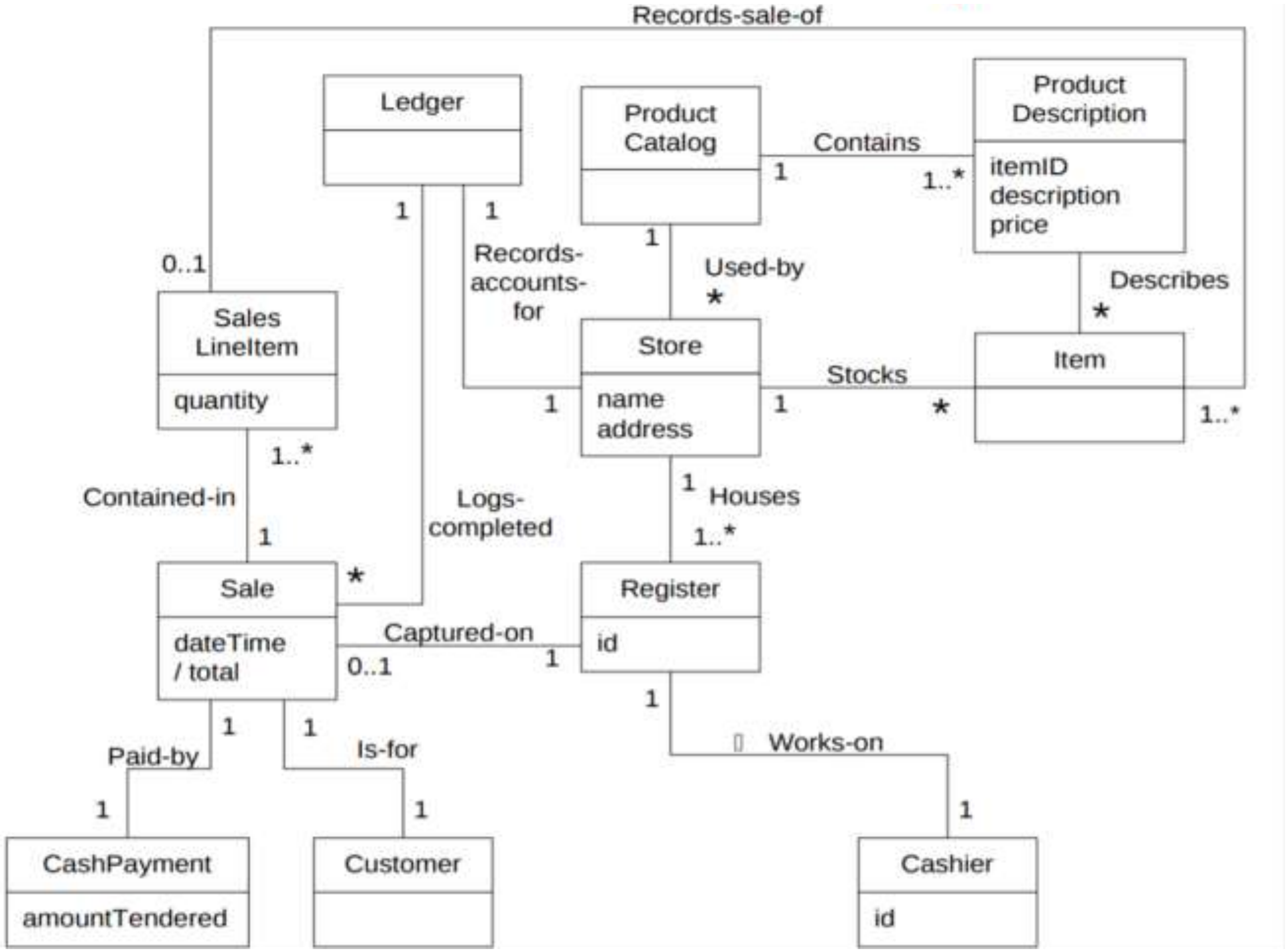




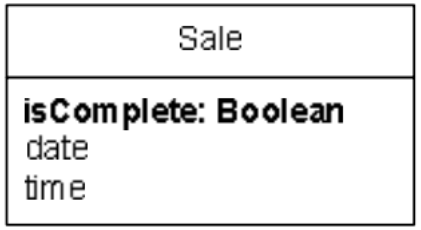
- A Sale instance s was created (instance creation).
- s was associated with the Register (association formed).
- Attributes of s were initialized.

- A SalesLineItem instance sli was created (instance creation).
- sli was associated with the current Sale (associationformed).
- sli.quantity became quantity (attribute modification).
- sli was associated with a ProductSpecification, based on itemID match (association formed).





Sale.isComplete became true (attribute modification)

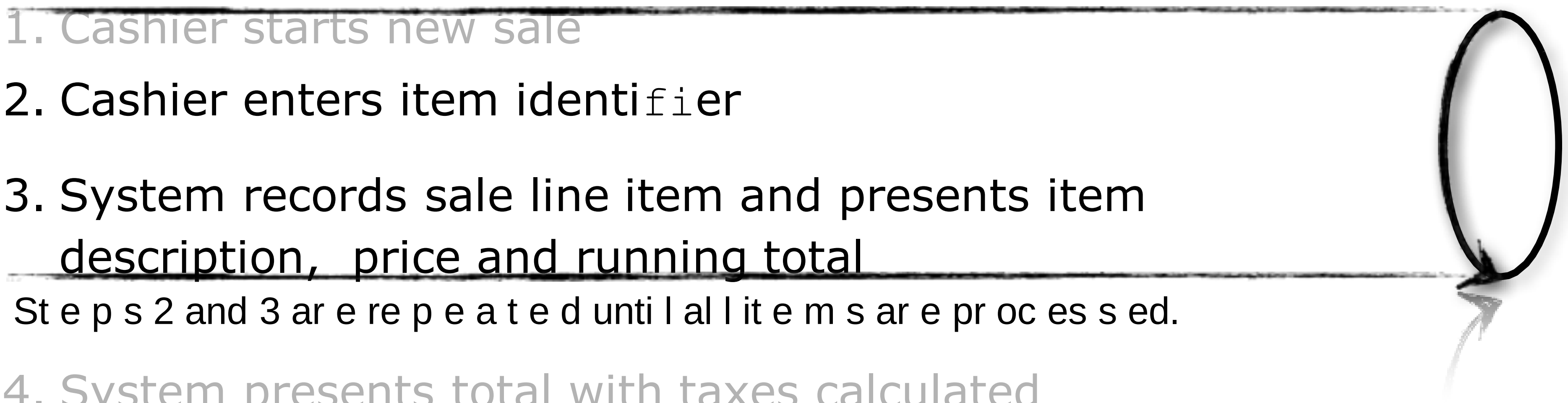


- A Payment instance p was created (instance creation).
- p.amountTendered became amount (attribute modification).
- p was associated with the current Sale (association formed).
- The current Sale was associated with the Register (association formed)

Example of an SSD for the Process Sale Scenario

Use Case: Process Sale Scenario - Main Success Story

1. Cashier starts new sale



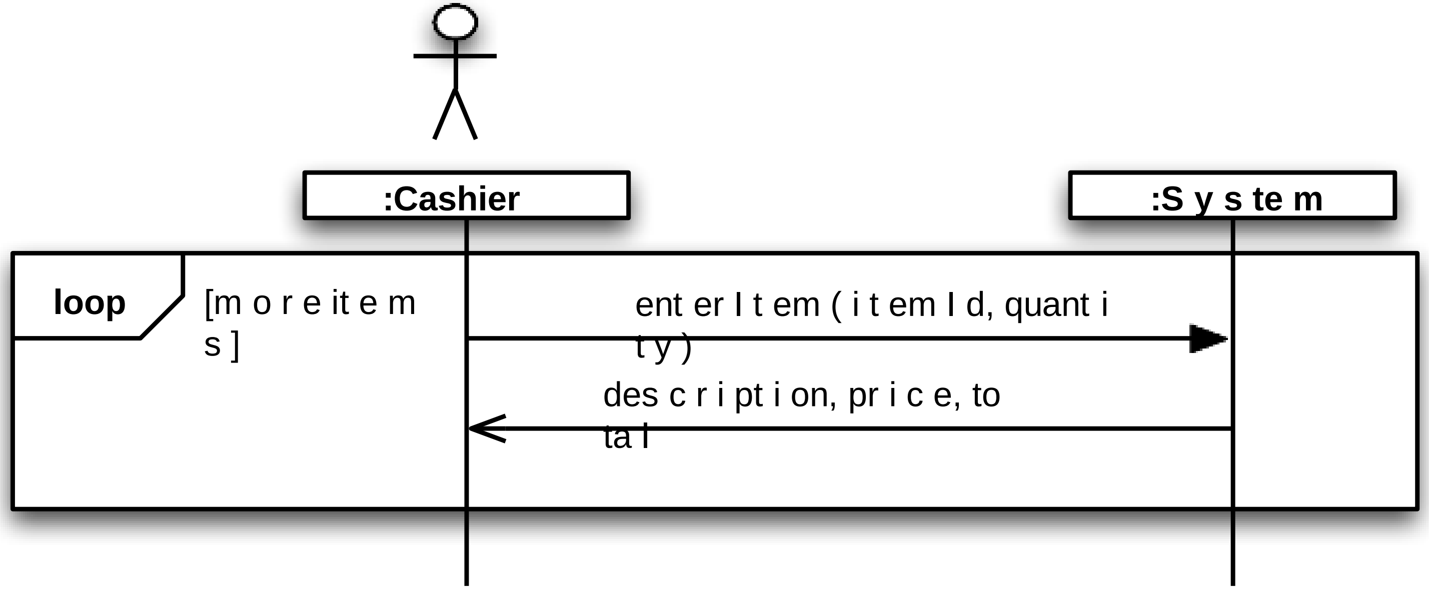
2. Cashier enters item identifier
3. System records sale line item and presents item description, price and running total

Steps 2 and 3 are repeated until all items are processed.

4. System presents total with taxes calculated
5. Cashier tells Customer the total and asks for payment
6. Customer pays and System handles payment

Visualizing SSDs - Excerpt From the POS Domain

Process Sale Scenario



Message Formats in System Sequence Diagrams

- The UML format for a message consists of a message name followed (in parentheses) by a parameter list.
- All names begin with a lower-case letter.
- There are no spaces in a name.
- Upper-case letters separate the words within a name.
- Names in the parameter list are separated by commas.