# LECTURE 5

## SEQUENCE DIAGRAM

# INTRODUCTION – SYSTEM SEQUENCE DIAGRAM

- A system sequence diagram is a fast and easily created artifact that illustrates input and output events related to the systems under discussion

- Before proceeding to a logical design of how a software application will work, we should investigate and define the system behavior as a "black box".

# SYSTEM SEQUENCE DIAGRAM

- A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events.

- An SSD is generated from inspection of a use case
- Suggestion:
  - One SSD – one Use Case

# SYSTEM SEQUENCE DIAGRAM VS SEQUENCE DIAGRAM

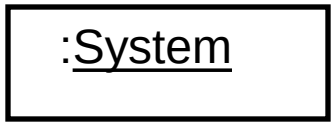
- A System Sequence Diagram is an artifact that illustrates input and output events related to the system under discussion.
- System Sequence Diagrams are typically associated with use-case realization in the logical view of system development.
- Sequence Diagrams (Not *System* Sequence Diagrams) display object interactions arranged in time sequence.

# SEQUENCE DIAGRAMS

- Sequence Diagrams depict the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the system.

# SSD—SYSTEM BEHAVIOR

- System behaves as “Black Box”.
- Interior objects are not shown, as they would be on a Sequence Diagram.



# SYSTEM SEQUENCE DIAGRAMS

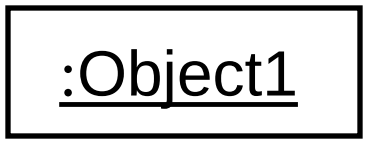
- For a particular scenario of use-case an SSD shows-
- The external actors that interact directly with the system.
- The System (as a black box).
- The system events that the actors generate.

# SYSTEM SEQUENCE DIAGRAMS

- The operations of the system in response to the events generated.
- System Sequence Diagrams depict the sequential order of the events.
- System Sequence Diagrams should be done for the main success scenario of the use-case, and frequent and alternative scenarios.

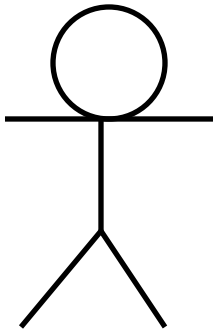
# NOTATION

- Object: Objects are instances of classes. Object is represented as a rectangle which contains the name of the object underlined.
- Because the system is instantiated, it is shown as an object.



# NOTATION (2)

Actor: An Actor is modeled using the usual symbol, the stick figure.



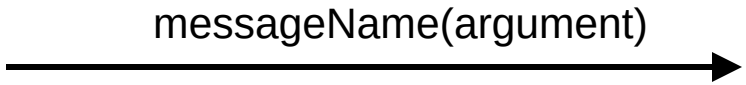
actor1

# NOTATION (3)

- Lifeline: The LifeLine identifies the existence of the object over time. The notation for a Lifeline is a vertical dotted line extending from an object.

# NOTATION (4)

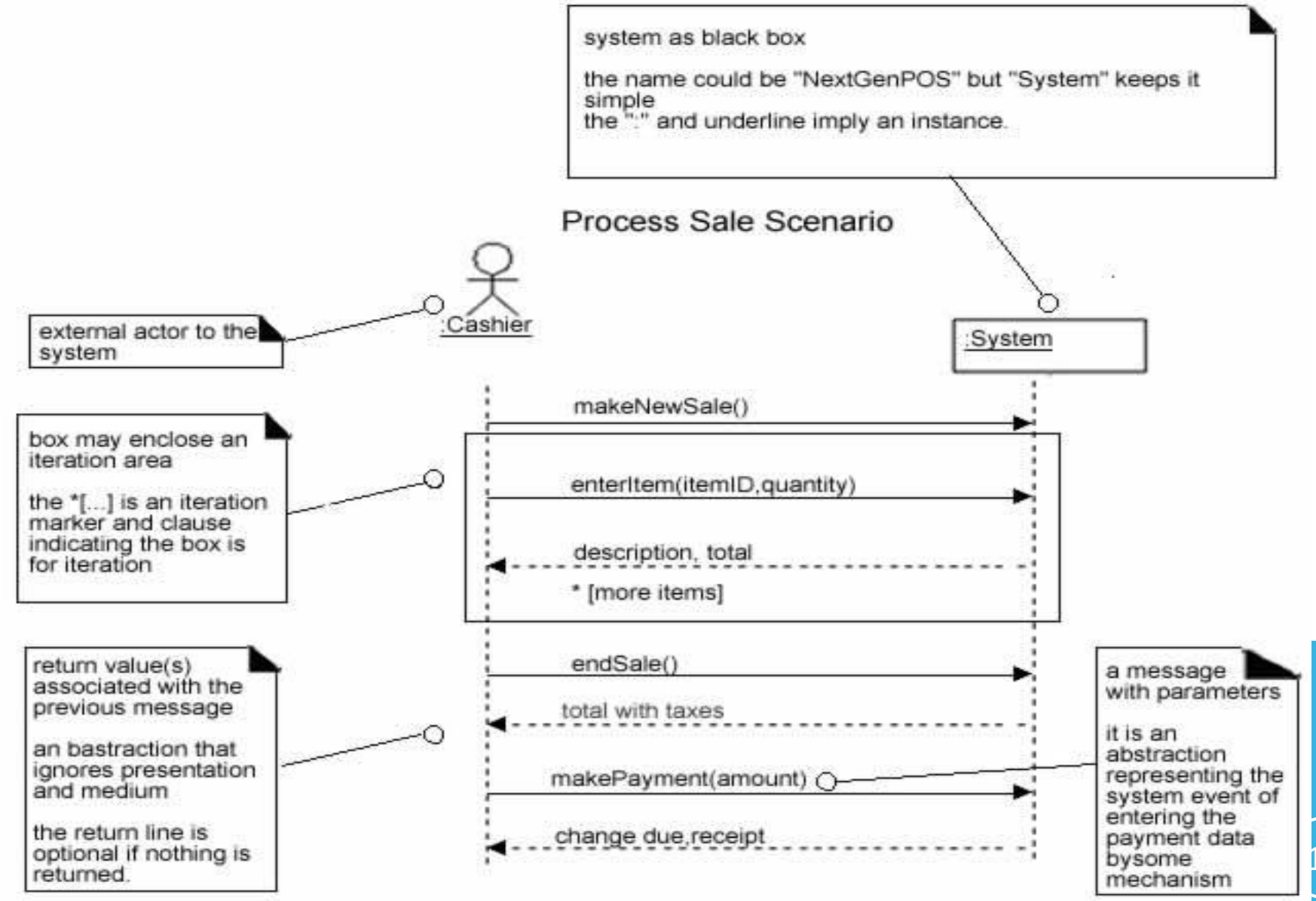
Message: Messages, modeled as horizontal arrows between Activations, indicate the communications between objects.



# EXAMPLE OF AN SSD

- Following example shows the success scenario of the Process Sale use case.
- Events generated by cashier (actor)-
- makeNewSale
  - enterItem
  - endSale and
  - makePayment.

# SSD FOR PROCESS SALE SCENARIO



# SYSTEM SEQUENCE DIAGRAMS AND USE CASES

- System Sequence Diagram is generated from inspection of a use case.
- Constructing a systems sequence diagram from a use case-

1. Draw a line representing the system as a black box.
2. Identify each actor that directly operates on the system. Draw a line for each such actor.
3. From the use case, typical course of events text, identify the system (external) events that each actor generates. They will correspond to an entry in the right hand side of the typical use case. Illustrate them on the diagram.

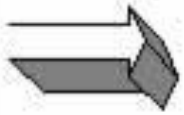
# SSDS ARE DERIVED FROM USE CASES.

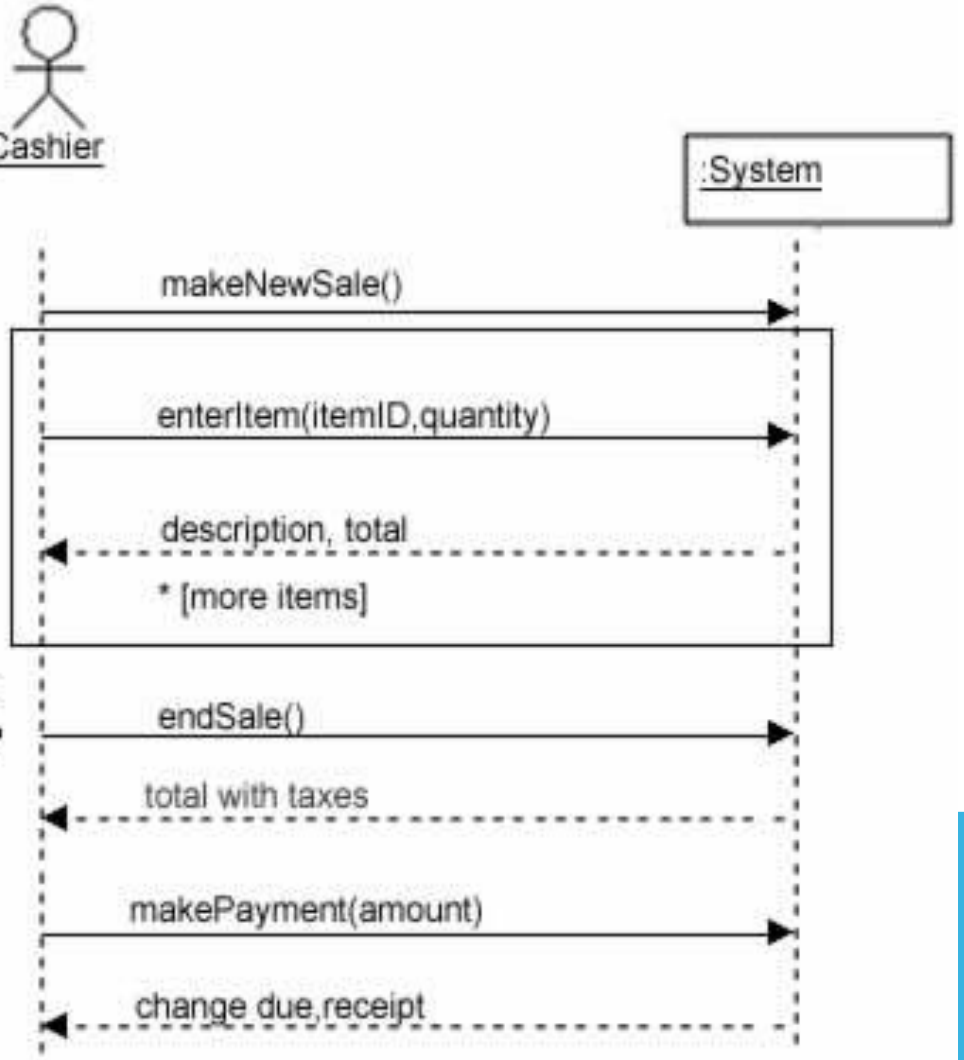
Simple Cash-only Process Sale scenario:

1.customer arrives at aPOS check out with goods and/or services to purchase.
2.Cashier starts a new sale.
3.Cashier enters a new item identifier.
4.System records new sale line item and presents item description,price and running total.

Cashier repeats steps steps 3-4 until indicates done.

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





# SYSTEM EVENTS AND SYSTEM BOUNDARY

Identifying the System events-

1. Determine the actors that directly interact with the system.
2. In the process Sale example, the customer does not directly interact with the POS system. Cashier interacts with the system directly. Therefore cashier is the generator of the system events.

# NAMING SYSTEM EVENTS AND OPERATIONS

- **System event**
- External input event generated by an actor.
- Initiates a responding operation by system.
- **System operation**
- Operation invoked in response to system event.

# NAMING SYSTEM EVENTS AND OPERATIONS

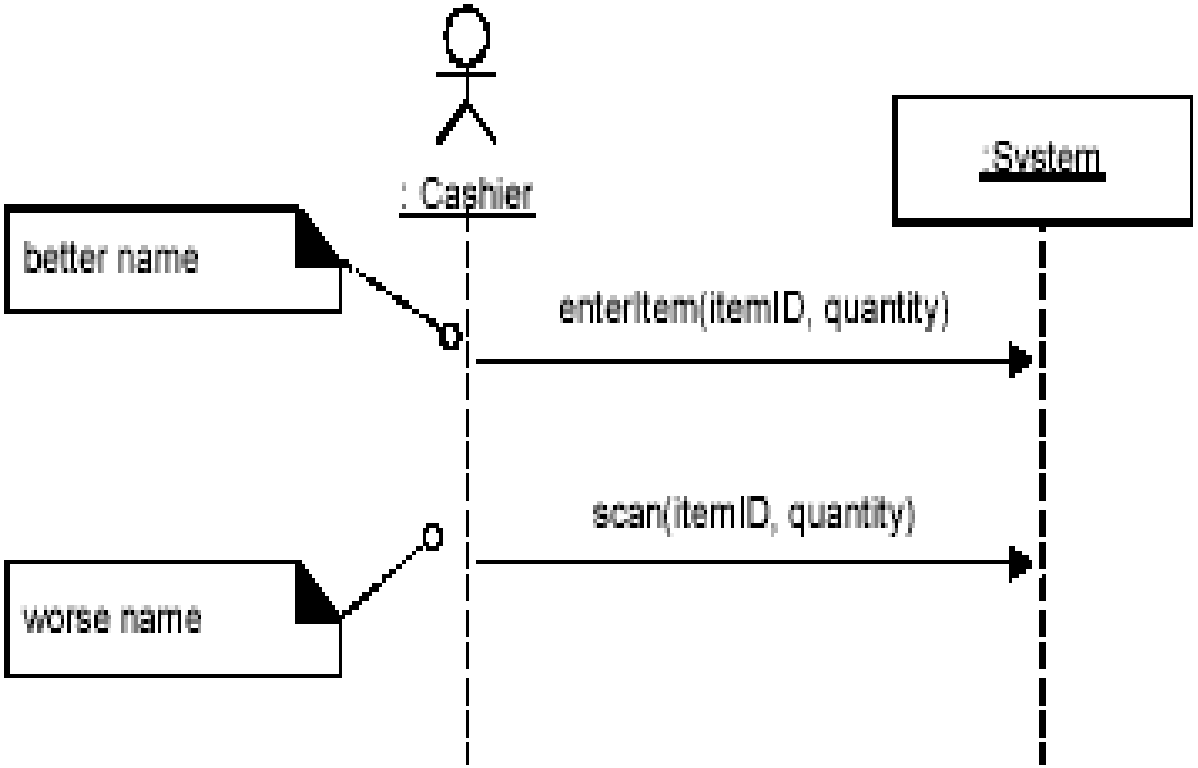
- System events and their associated system operations should be expressed at the level of intent rather than in terms of the physical input medium or widget.

- In order to improve the clarity, it is appropriate to start the name of the system event with a verb (for example- add….,enter….,end….,make…. etc.,). It also emphasizes the command origination of these events.

# NAMING SYSTEM EVENTS AND OPERATIONS

- For example “enterItem” is better than “scan” as it captures the intent of operation rather than what interface is used to capture the system event (design choice).

# CHOOSE EVENT AND OPERATION NAMES AT AN ABSTRACT LEVEL



# SHOWING USE CASE TEXT

- It is desirable to show at least fragments of use case text for the scenario.
- The text provides the details and context, while the diagram visually summarizes the interaction.

# SSD WITH USE CASE TEXT

Simple Cash-only Process Sale scenario:

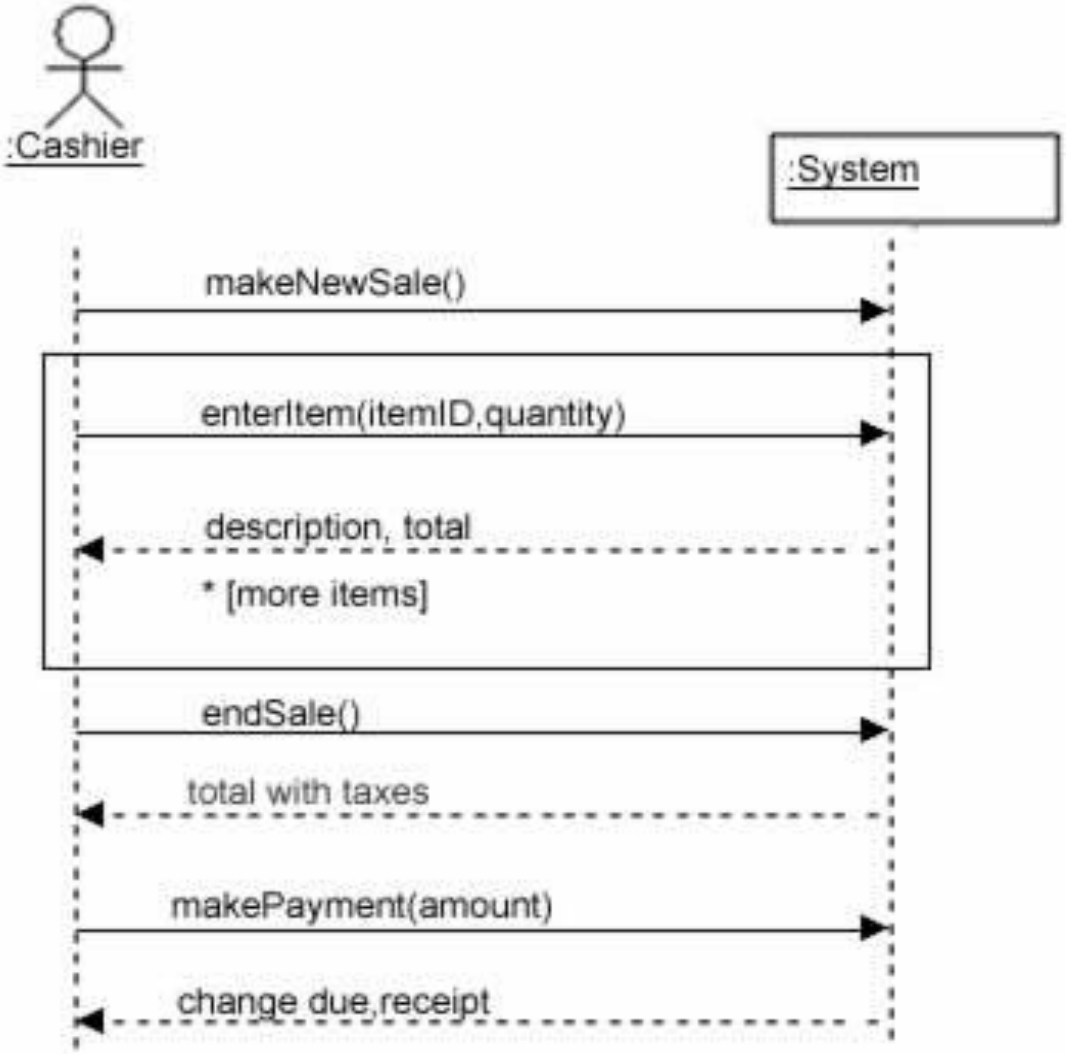
1.customer arrives at aPOS check out with goods and/or services to purchase.
2.Cashier starts a new sale.

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

Cashier repeats steps steps 3-4 until indicates done.

5.System presents total with taxes calculated.

6.Cashier tells Customer the total, and asks for payment.
7.Customer pays and System handles payment.
....



# CONCLUSION

- System Sequence Diagrams provide a way for us to visually step through invocation of the operations defined by Use-Cases.
- It is not necessary to create SSDs for all scenarios of all use-cases,at least not at the same time.