Sequence diagram

Sequence diagram

- Describes in more detail a scenario in a use case

  - Created from the text of the use case

  - Sequence diagram is a simplified version useful for requirement Analysis.

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

  - They are input to operation Contracts and most importantly - object design

Constructing sequence diagram

step 1: Designate actors and business system
- who is taking part?

Step 2: Designate initiators - who start interactions?

Step 3: Describe the message exchange b/w actors and business system - which messages are

being exchanged?

step 4: identify the course of interactions, what is the order?

step 5: Insert additional information - what else is important?

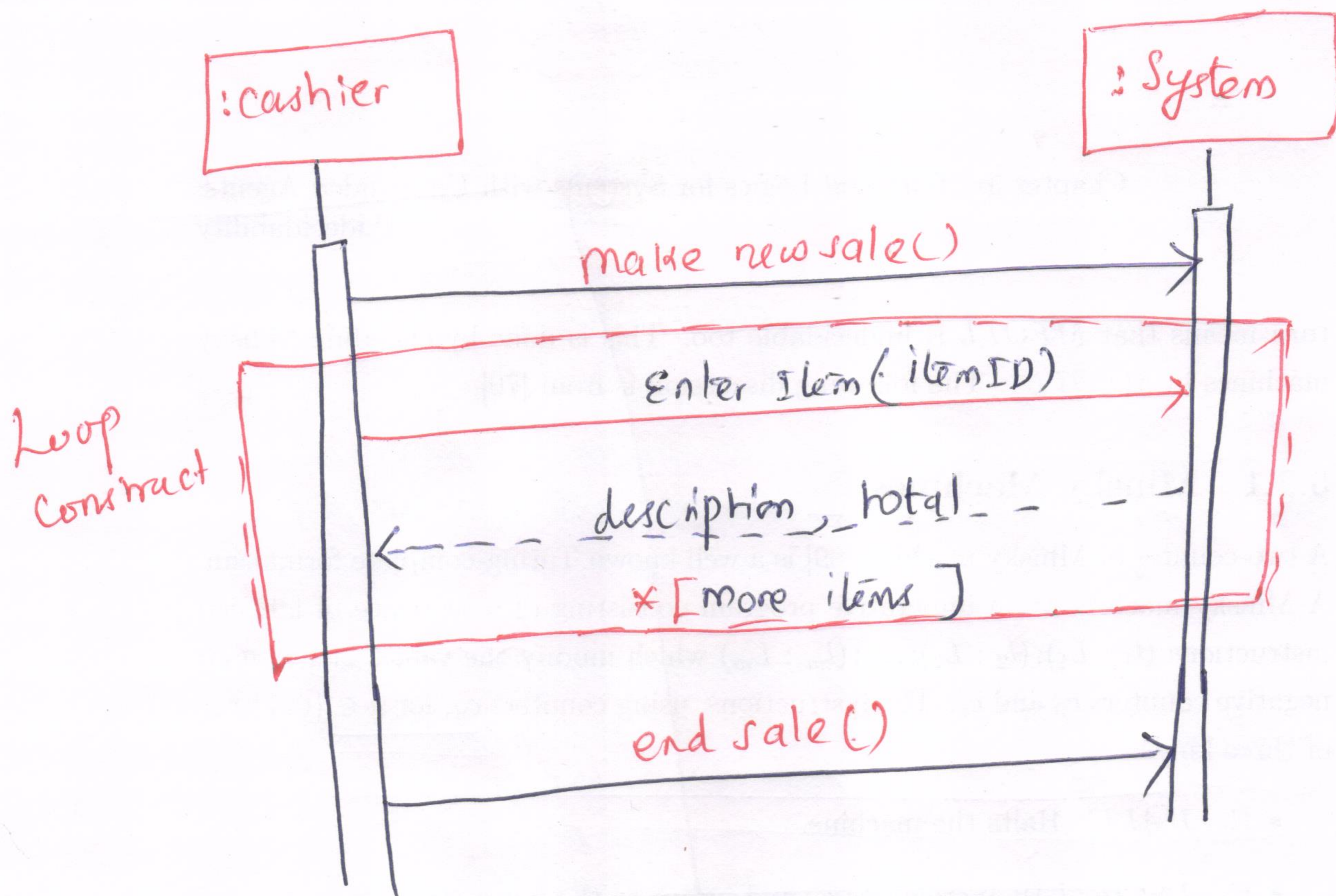step 6: Verify the view - Is everything correct?

Sequence diagram

Sequence diagram illustrate the interactions between actors and business System.

Example: process sale for pos system

Statements:

- Customer arrives with goods
- Cashier starts a new sale
- Repeat until done:
  - Cashier enter item id
  - System records sale line item and presents description and running total.

Loop
Contract

:cashier   : System

make new sale()

Enter Item (itemID)

description , total

*[more items]

end sale()

Elements of SSD

  _ Actor: :Cashier   uml notation for an object

  _ System Events:

    _ cashier generates make new sale, enter item,
  and end sale system Events.
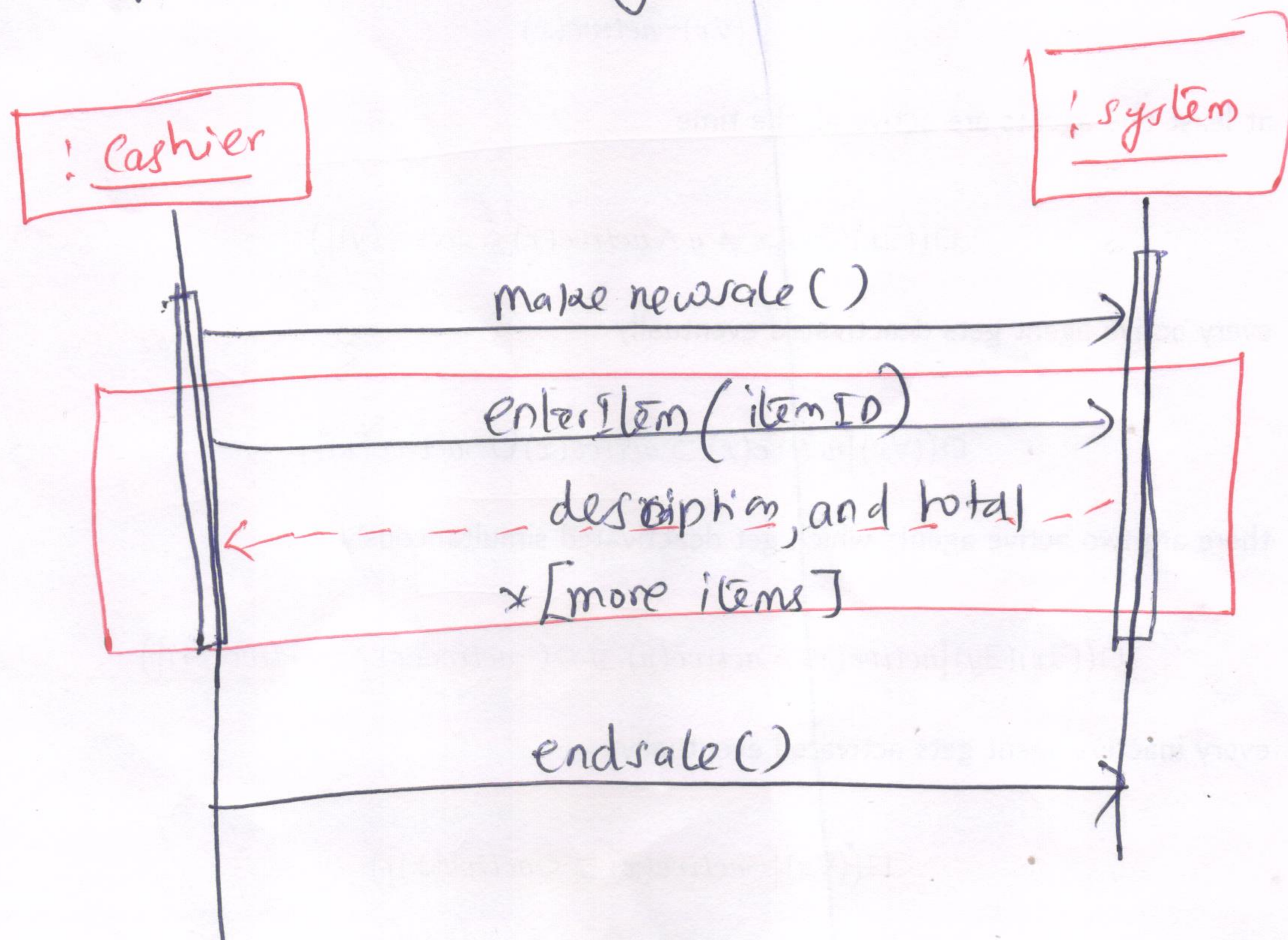
  _ information from the system back to the actor

    _ item description, running total, etc
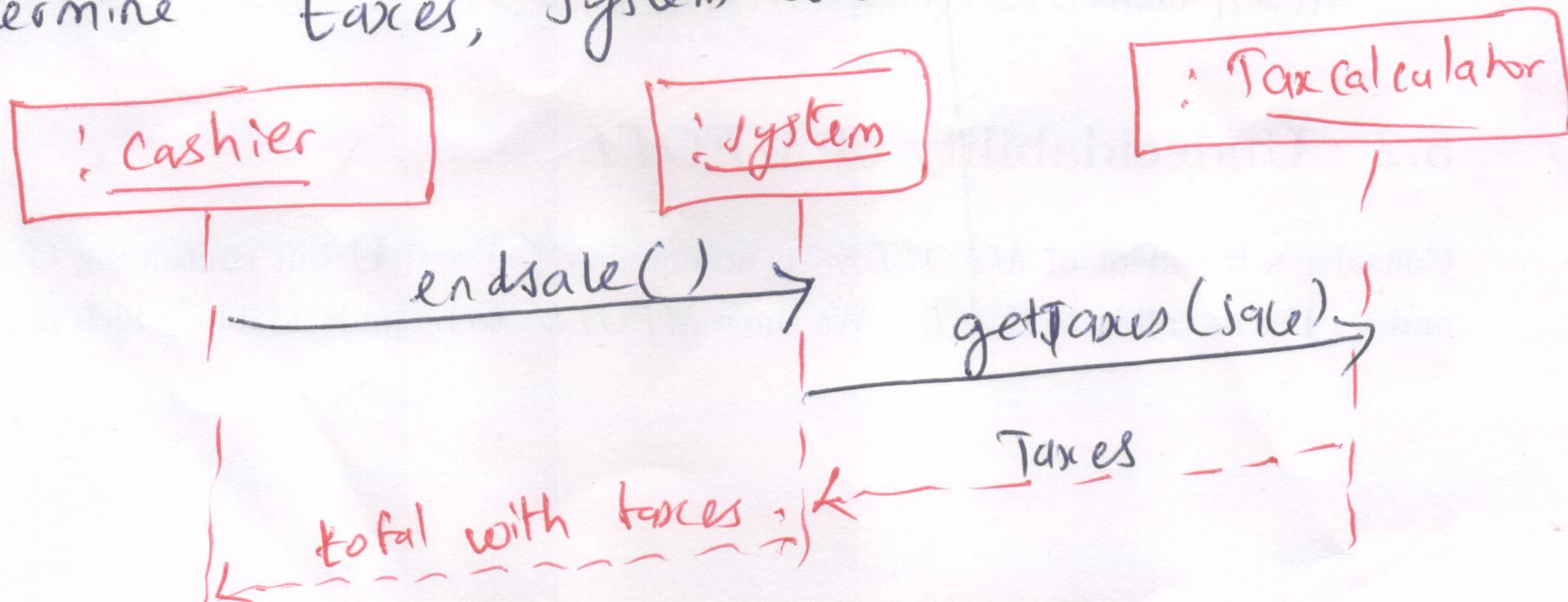
sequence diagram for "process sale" in Next cheappos

Problem statement

- Cashier starts a new sale
- Cashier enters itemid
- System records sale line item and presents

description and running total.

```
┌──────────┐                                        ┌──────────┐
│ : Cashier │                                        │ : system │
└──────────┘                                        └──────────┘
     │                 make newsale ( )                   │
     ├──────────────────────────────────────────────────►│
     │                 enterItem ( itemID)                │
     ├──────────────────────────────────────────────────►│
     │              description, and total                │
     │◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ │
     │              *[more items]                         │
     │                  endsale ( )                       │
     ├──────────────────────────────────────────────────►│
     │                                                    │
```

- System presents total with taxes. to

determine taxes, System uses an external TaxCaculator

```
┌──────────┐         ┌──────────┐         ┌──────────────────┐
│ : Cashier │         │ : System │         │ : TaxCalculator  │
└──────────┘         └──────────┘         └──────────────────┘
     │      endsale()      │                        │
     ├────────────────────►│      getTaxes (sale)   │
     │                     ├───────────────────────►│
     │                     │         Taxes          │
     │                     │◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─│
     │  total with taxes   │                        │
     │◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─│                        │
```

- cashier enters cash amount tendered, the system presents change due.

- System presents receipt

- System logs completed sale and sends sale ifo to the external accounting system and to the external inventory system.