What a sequence diagram actually shows

A Sequence Diagram shows the INTERACTION between participants over TIME. Three things it captures:

- **Who's involved**: actors and objects (participants)
- **What they say to each other**: messages with method calls and parameters
- **When things happen**: vertical axis goes top-to-bottom in time order

Sequence diagrams are different from class diagrams (which show static structure) and from activity diagrams (which show flow without who-does-what). Use sequence diagrams when the order of who-talks-to-who matters, banking transactions are the textbook example.

Sequence diagrams live in Chapter 3, Section 3.6 (System Design), typically after the static diagrams (Use Case, Class, ER) and before Activity Diagrams. For the general sequence diagram guide covering all UML conventions, see our Sequence Diagram guide.

The 6 elements of every sequence diagram

Six building blocks.

1. Actor or Object (rectangle at top)

A participant in the interaction. Actors are people (Customer); objects are system components (BankingService, Database).

Drawn as a rectangle at the top of the diagram with the name inside. Actors may also use the stick-figure notation from use case diagrams.

2. Lifeline (vertical dashed line)

A dashed vertical line going down from each participant. Represents the lifetime of that participant during the interaction.

If a participant is "destroyed" mid-sequence, their lifeline ends with an X.

3. Activation Box (narrow vertical rectangle)

A thin vertical rectangle on a lifeline showing when that object is actively doing work (executing a method).

Activation boxes appear when an object receives a message and disappear when the work is done. They're optional in some notations but recommended for clarity.

4. Message (horizontal arrow)

A horizontal arrow from one lifeline to another, labeled with the method name and parameters being called.

The arrow direction shows who's calling whom. Vertical position shows when in time it happens (earlier messages higher on the page).

5. Return Message (dashed horizontal arrow)

A dashed arrow returning a value. Optional but recommended when the return value is meaningful.

Example: success or account_balance: 5000.00

6. Self-message (curved arrow)

A message from an object to itself. Drawn as a small arc that goes out and comes back to the same lifeline.

Used when an object calls one of its own methods internally.

Message types: synchronous vs asynchronous

The arrow style tells you the message type.

Synchronous message (solid arrow with filled triangle)

————————► with filled triangle head

- The caller waits for the response before continuing
- Standard method call in most code
- Most common message type
- Example: withdraw(500.00), caller waits for success/failure return

Asynchronous message (solid arrow with open triangle)

————————▷ with open/skeletal triangle head

- The caller doesn't wait, fire and forget
- Used for events, notifications, queue messages
- Example: sendSMSNotification(message), caller continues immediately

Return message (dashed arrow)

- - - - - - -►

- Response from a synchronous call
- Drawn as dashed line, often with the return value labeled
- Can be omitted if the return is implicit or obvious

Create message (with <<create>> label)

When a new object is created during the sequence:

—<<create>>—► pointing to a NEW lifeline starting at that moment

Destroy message (X at end of lifeline)

When an object is destroyed:

——X at the bottom of the destroyed lifeline

Rarely needed in BSIT capstones unless you're modeling explicit memory management.

Here's the representation of the full money transfer sequence: