Interaction Diagrams

SEQUENCE DIAGRAM

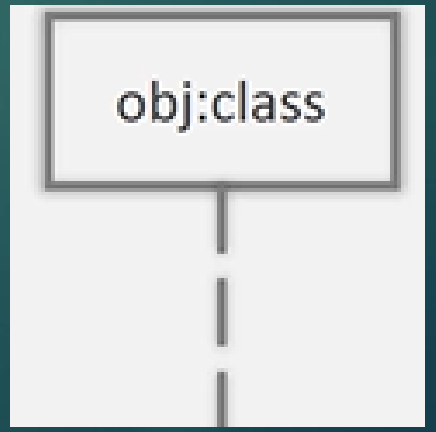
Sequence Diagrams

- Used to show interactions between objects in a sequential order
- Often generated as an outcome of refinement of use case
- Useful to stakeholders to understand the system's behavior
- May be used for designing new systems or depicting interactions in existing systems
- Helps understand how the control/events flow between the objects

Participant

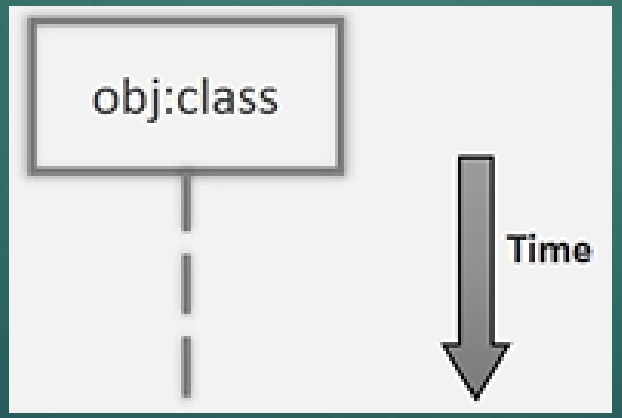
- A participant is some part in the system that interacts with other participants
- It is always placed at the top of the diagram
- Participants are arranged horizontally
- Each participant has a correspond lifeline that runs down the page
- The name of the participant in put in the box in the format

  object: class



Time

- Ordering is important in a sequence diagram
- The ordering is represented through time as lifelines
- The lifelines run down from the top
- Note that the lifeline does not represent duration

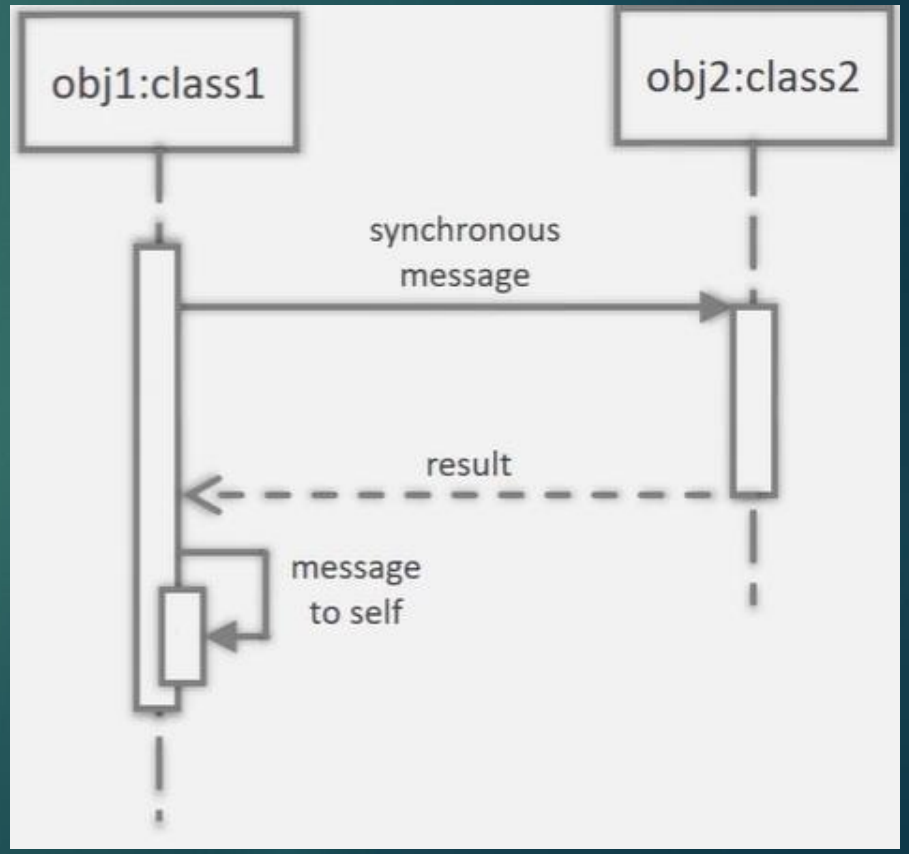


Message

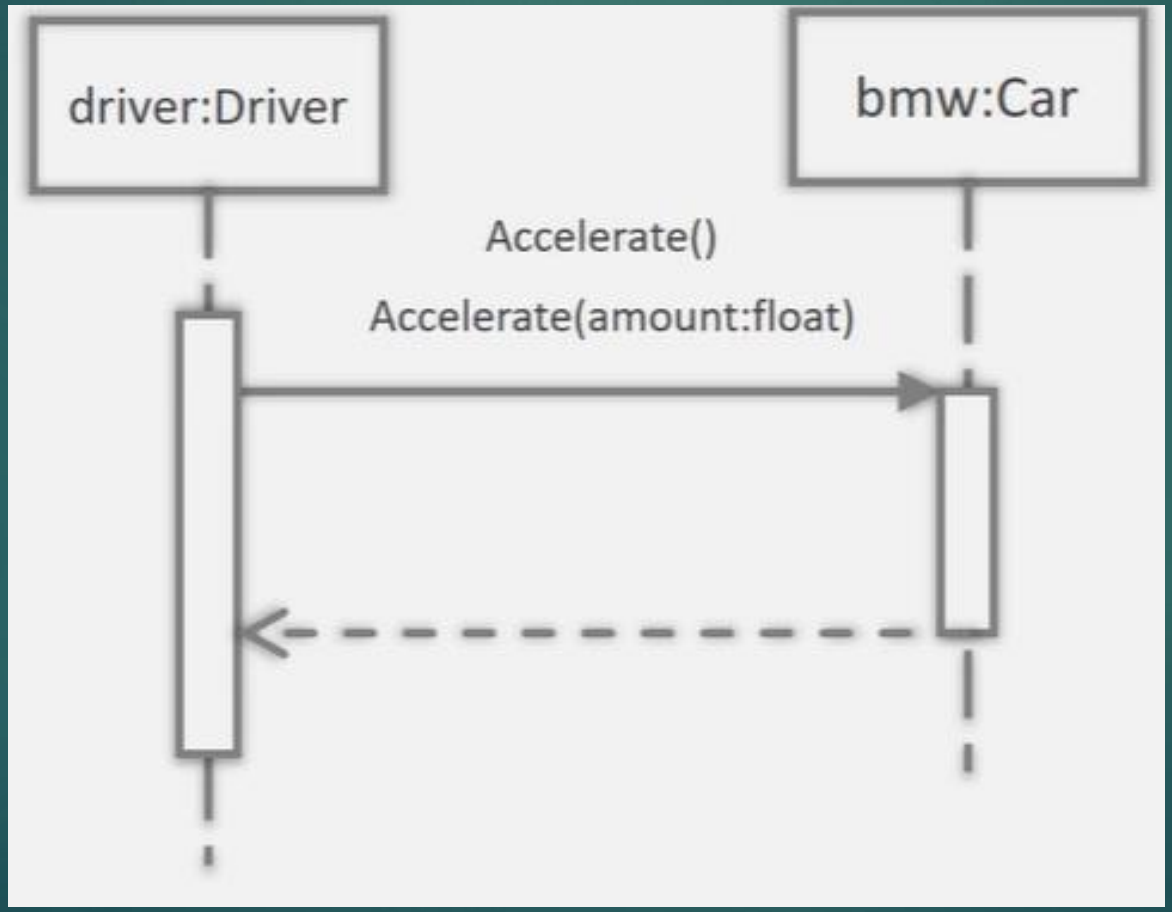
- An interaction allows the participants to communicate with each other
- Communication happens through messages
- Specified as an arrow towards the participant accepting the message
- A message can flow from any direction
- Participant to other & back
- Participant to self
- A message may optionally show the message signature

Activation/execution bar

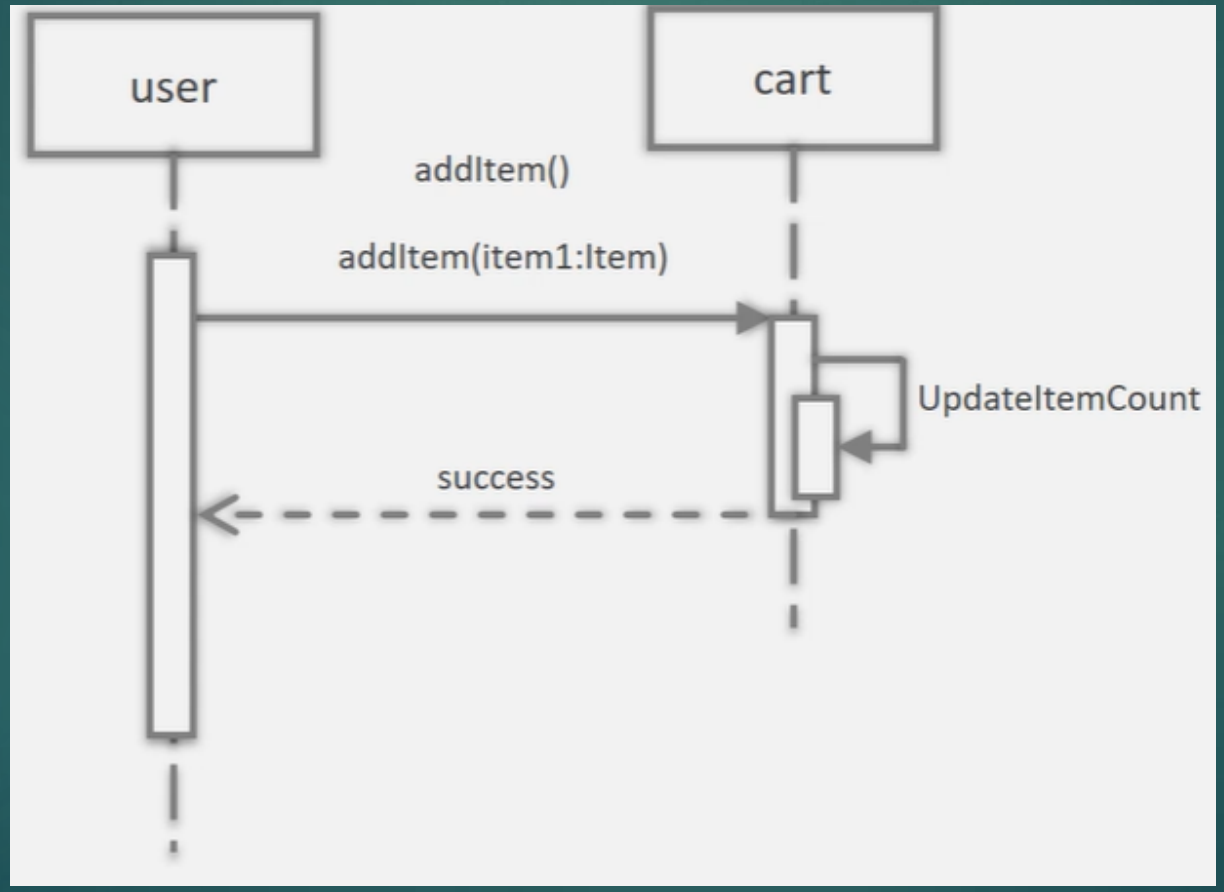
- Participants are shown as boxes with name inside (instance of a class)
- Arrow represent messages
- Activation bar indicates execution and is optional
- Note that it only represents time, but not duration



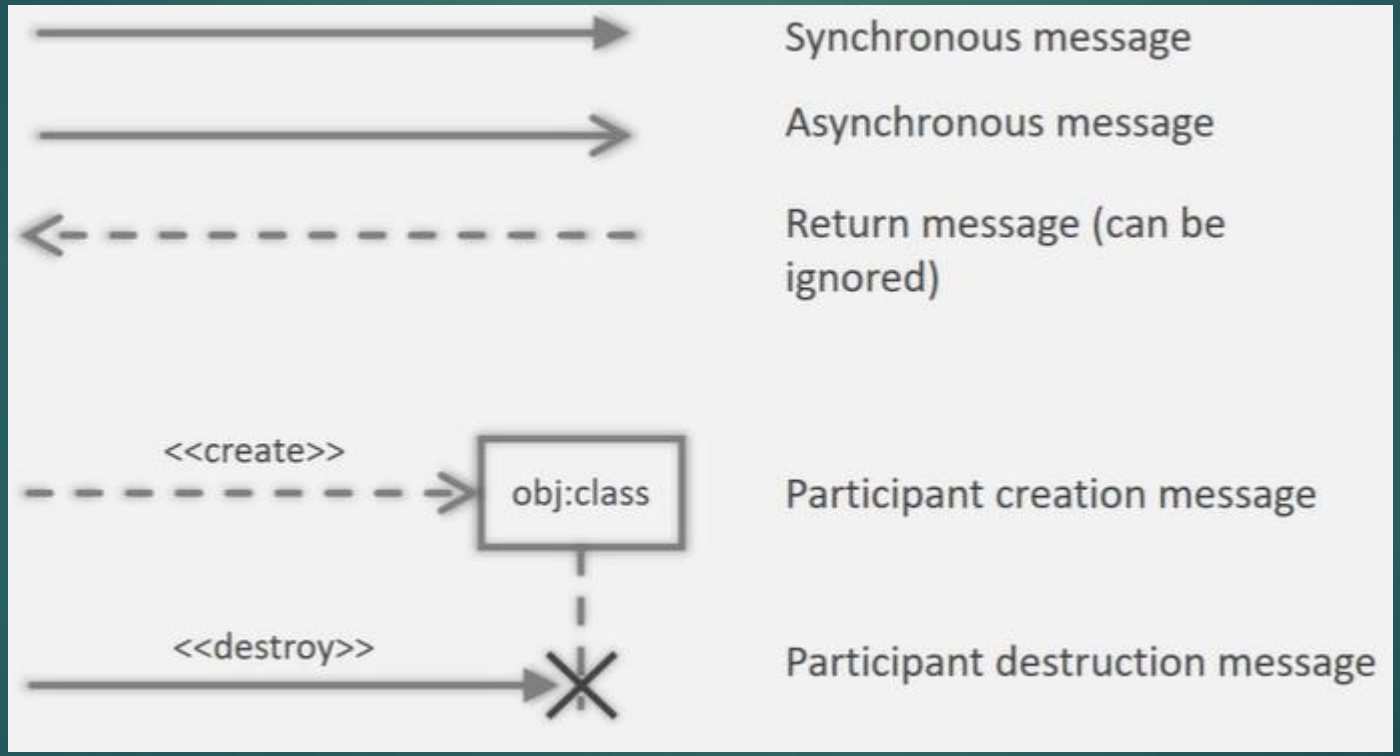
Example



Example


user
cart
addItem()
addItem(item1:Item)
UpdateItemCount
success

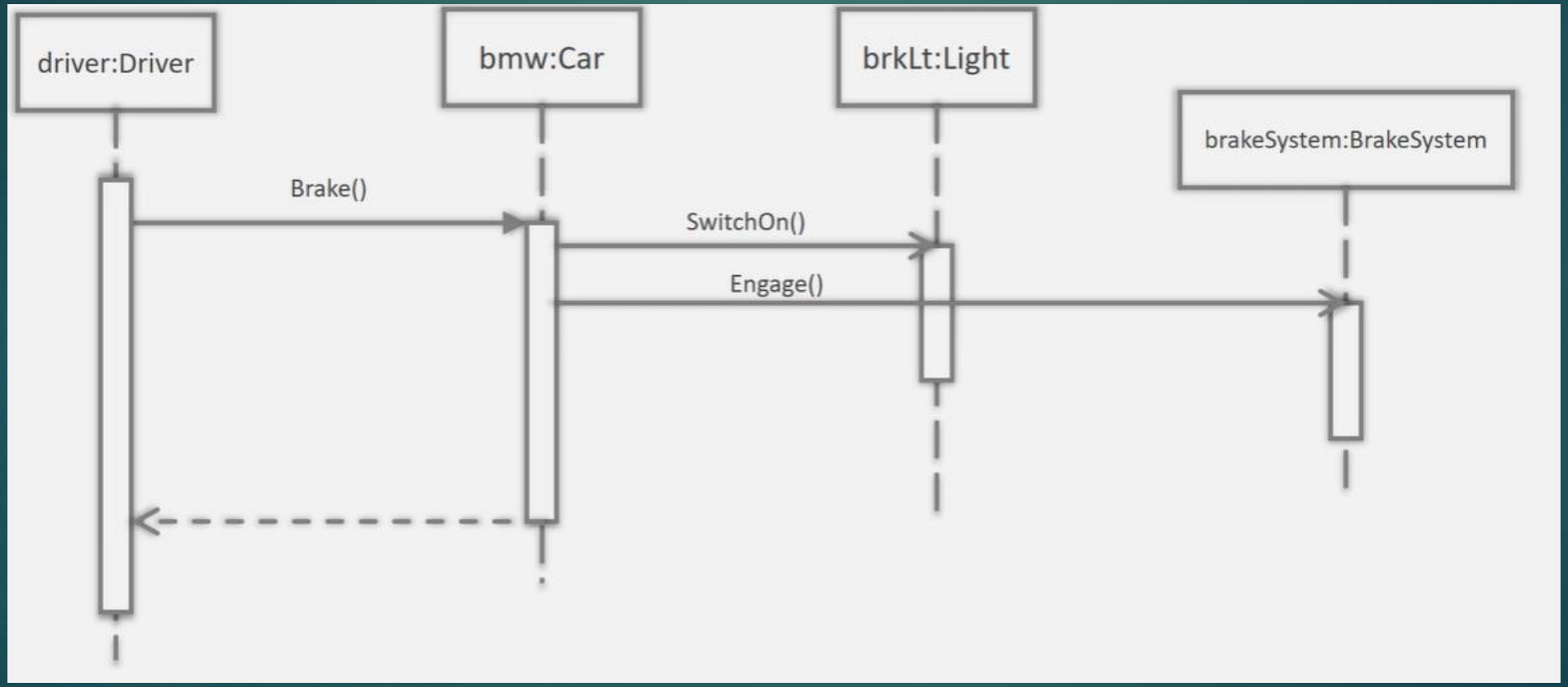
Message type



Asynchronous Message

- In some case, we want to execute a set of interactions at the same time
- We may or may not wait for them to finish
- This is required during asynchronous execution of some process
- This is required during asynchronous execution of some process
  - Prevent UI from getting blocked
  - Execute multiple tasks at the same time for efficeincy

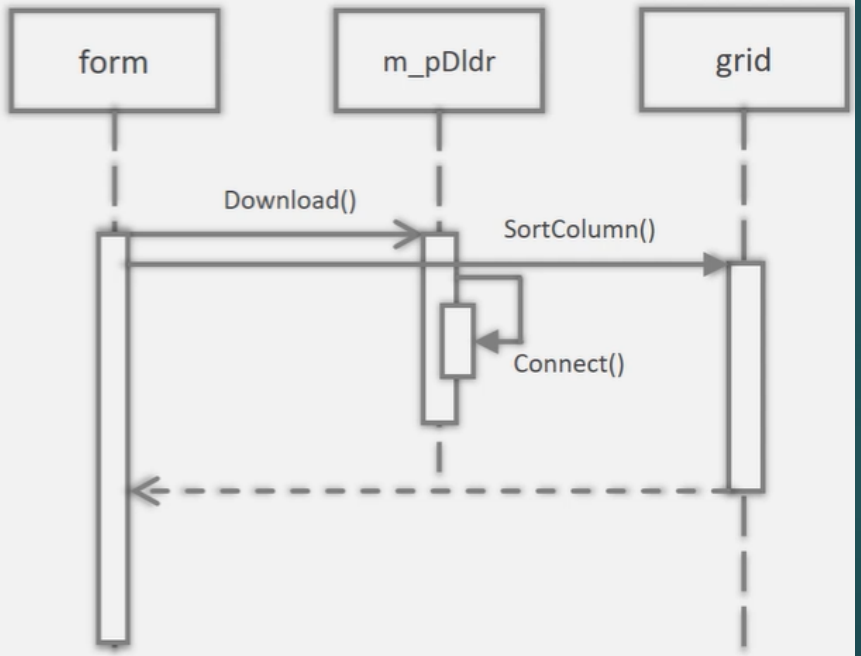
Example


driver:Driver
bmw:Car
brkLt:Light
brakeSystem:BrakeSystem
Brake()
SwitchOn()
Engage()

Example

```
class Form {
    std::future<void> m_Future;
    Downloader *m_pDownloader;
public:
    void OnClickHandler(Component *p) {
        std::function<
            void(Downloader &, const std::string&)
        > fn = &Downloader::Download;
        m_Future = std::async(fn, std::ref(*m_pDownloader), "file");
    }
    void ShowDownloadedFile() {
        if (m_Future.wait_for(std::chrono::microseconds(1))
            == std::future_status::ready) {
            m_Future.get();
            //Display the file
```

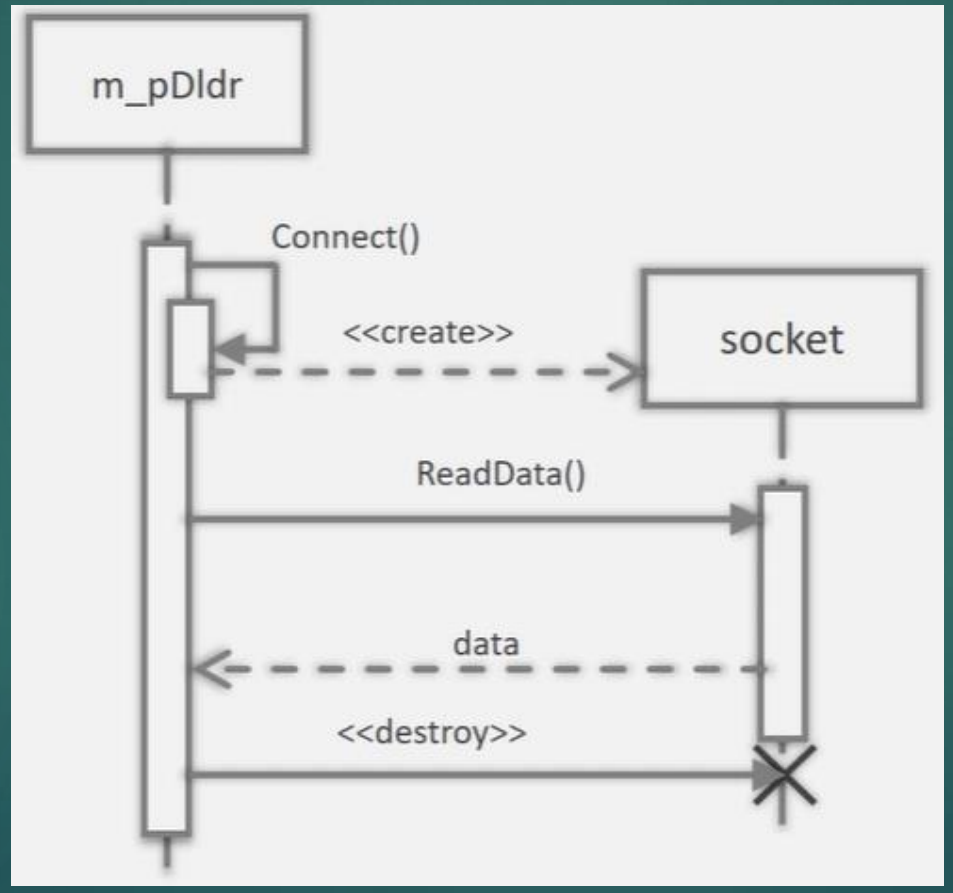
```
class Downloader {
public:
    //Asynchronous function
    void Download(const std::string &name) { ... }
```



Create & destroy Message

- Create message creates instance of a class
- Shown as stereotype <<create>> over the instance
- Destroy message destroys the instance
- It is shown as stereotype <<destroy>>
- Can be used to show explicit destruction of objects e.g.
- in C++, we use delete to destroy a heap-based object

Example



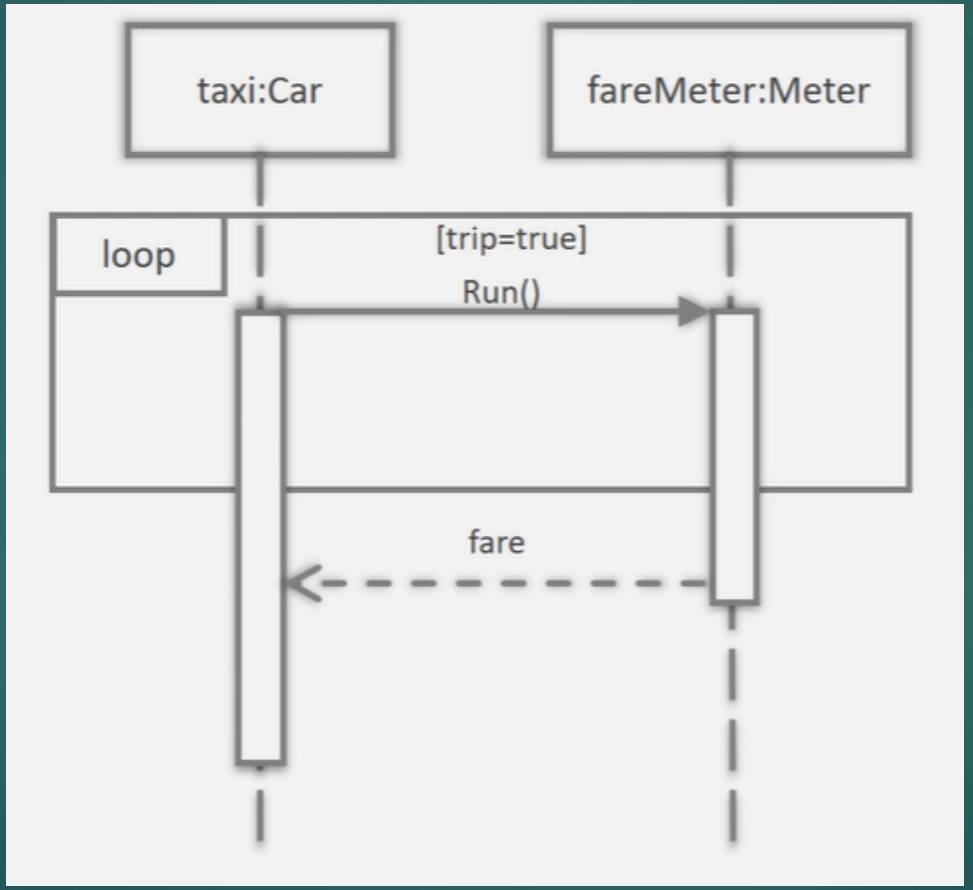
Fragments

- Sequence diagrams also depict loops, conditional execution, alternative execution, etc.
  - Can quickly grow and become unmanageable
- These interactions can be shown through fragments
- A fragment is a box that encloses the portion of the interaction
  - Can contain any number of interactions
  - Can contain other fragments

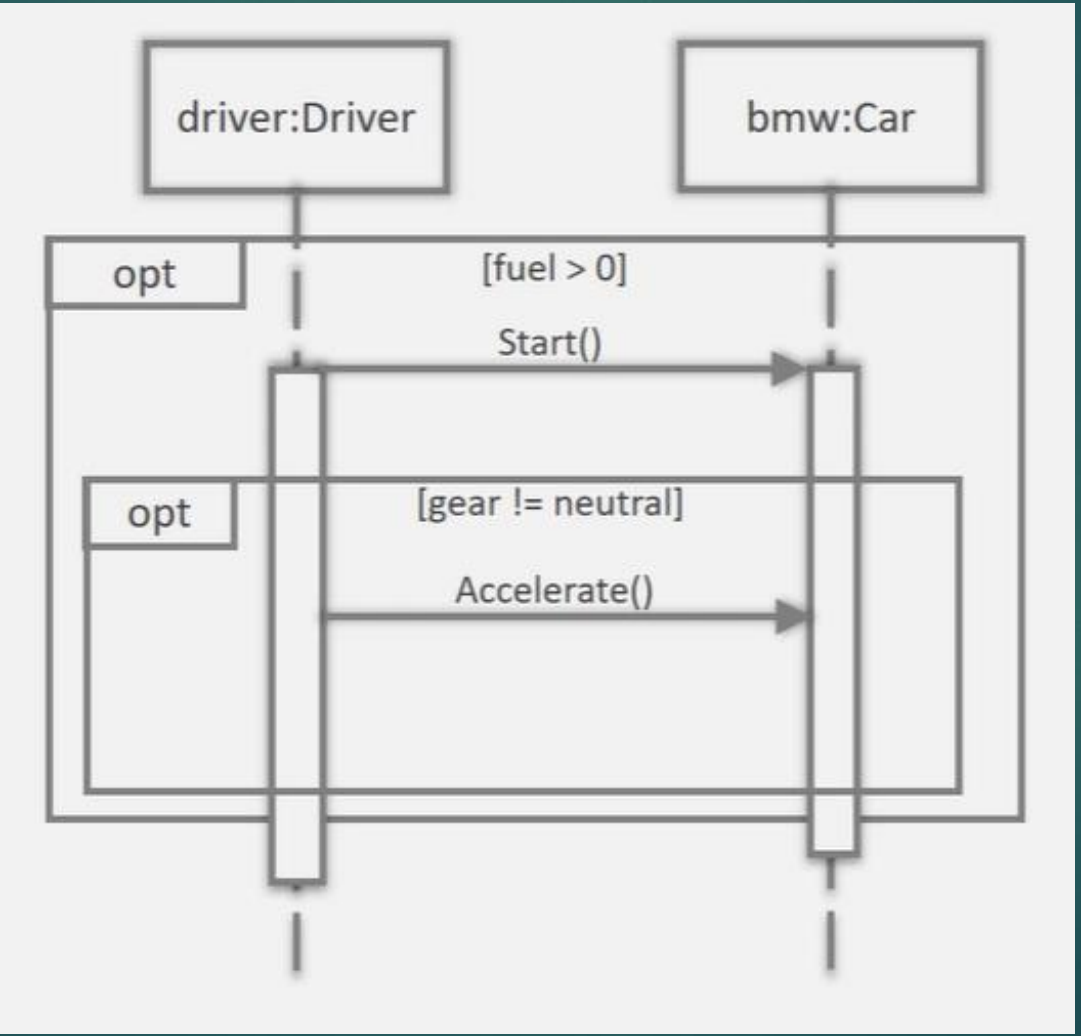
Fragment Types

- ref
  - used to represent an interaction that is defined elsewhere in the model
- loop
  - loops through interactions within the fragment any number of times (condition is shown as a guard statement)
- break
  - used to break out of a loop fragment (similar to break statement)
- opt
  - the interaction in this fragment is executed only if the guard condition is true

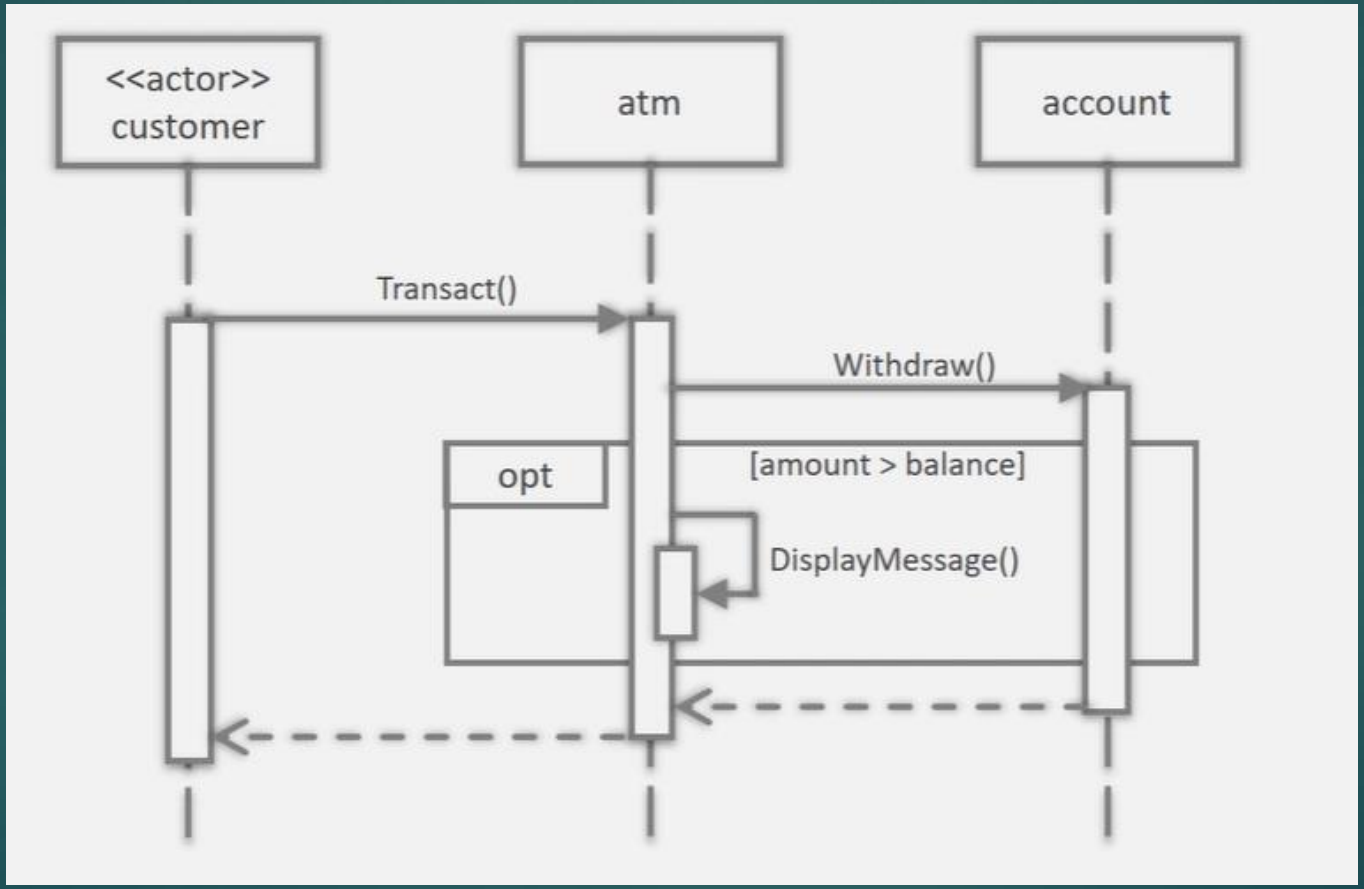
Example: Loop fragment



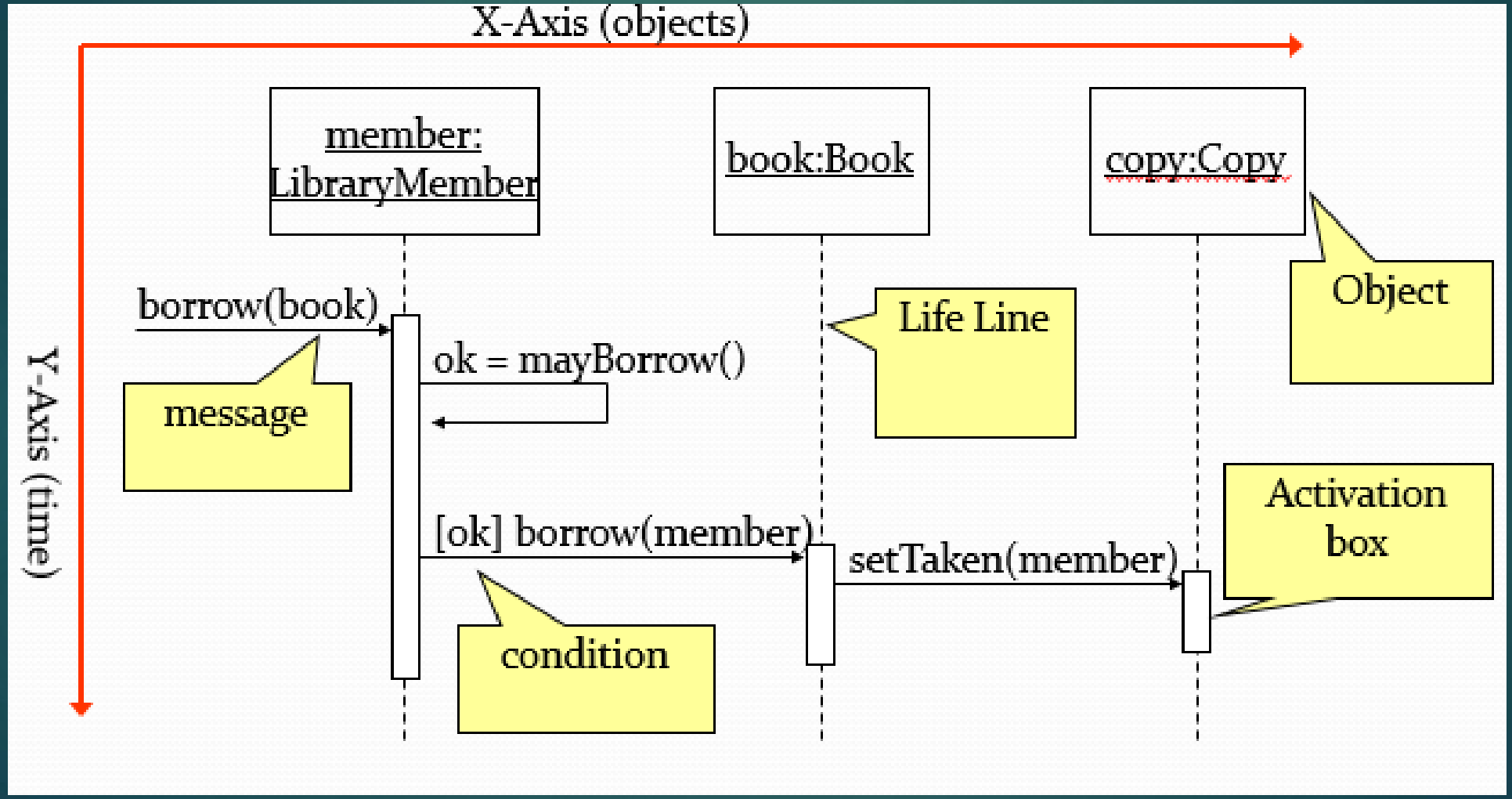
Example : opt Fragment



Example : opt Fragment



Example


X-Axis (objects)
Y-Axis (time)
member: LibraryMember
book:Book
copy:Copy
borrow(book)
message
ok = mayBorrow()
Life Line
Object
[ok] borrow(member)
condition
setTaken(member)
Activation box

Sequence Vs Communication

Sequence Diagram	Communication Diagram
• Shows the sequence or ordering of messages	• Focus on the links and the organization of objects
• Large set of detailed notations	• Fewer notation options
• Consumes horizontal space	• Requires less space
• Difficult to use during brainstorming	• Easy & flexible to draw during brainstorming
• Message order is depicted clearly	• Difficult to see the sequence of messages