
# THE ULTIMATE SEQUENCE DIAGRAM TUTORIAL

http://creately.com | @creately

- There are 3 types of Interaction diagrams in UML
  - Sequence diagrams
  - Communication diagrams
  - Timing diagrams

- Interaction diagrams are used to illustrate interactions of parts within a system.

- Out of these 3 types, sequence diagrams are preferred by both developers and readers alike for their simplicity.

# What is a Sequence Diagram?

- Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case.

- They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

# Drawing Sequence Diagrams with Creately

- Creately is a web based diagramming tool that can be used to draw sequence diagrams including all other UML diagram types as well as flowcharts, Gantt charts and many other diagram types

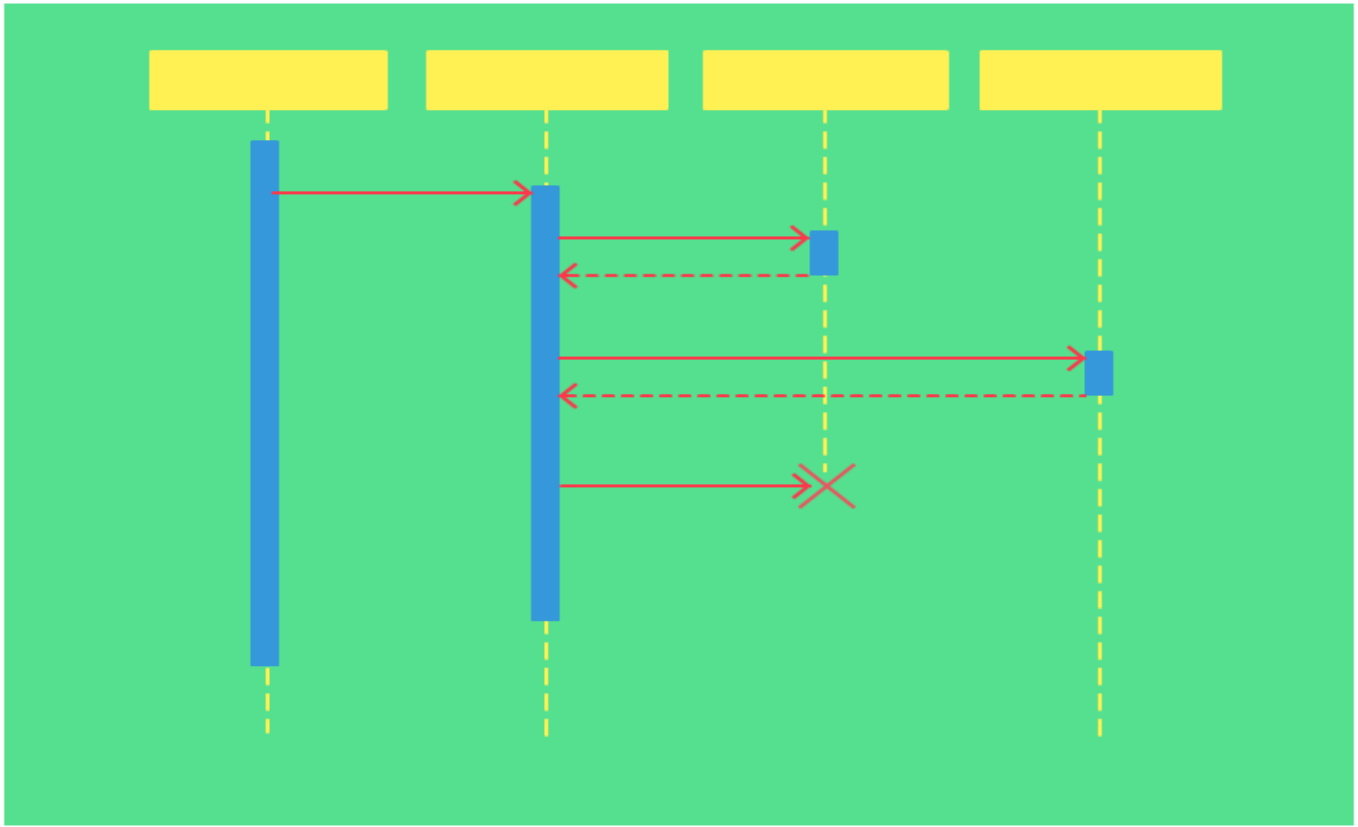
- It offers tools, professionally designed sequence diagram templates and resources to help draw sequence diagrams

- It comes with real-time collaboration features that allow users to work together and share diagrams with a team

- It has a desktop version for offline diagramming and a mobile app to manage and browse diagrams on the go

creately

# Sequence Diagram Notations

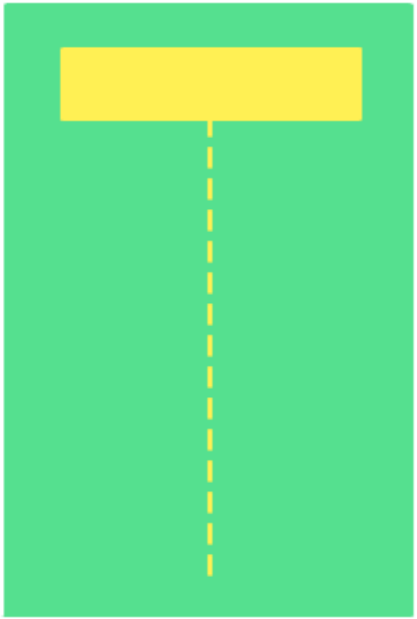


- A sequence diagram is structured in such a way that it represents a timeline which begins at the top and descends gradually to mark the sequence of interactions.

- Each object has a column and the messages exchanged between them are represented with arrows.

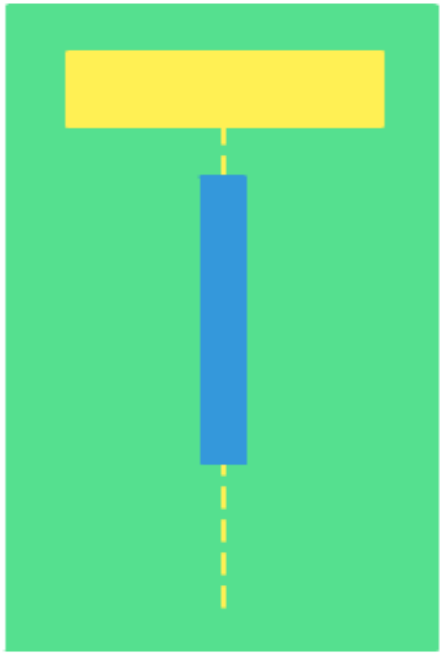
- Lifeline Notation
- Activation Bars
- Message Arrows
- Comment

# Lifeline Notation



- A sequence diagram is made up of several of these lifeline notations
- They should be arranged horizontally across the top of the diagram
- No two lifeline notations should overlap each other
- They represent the different objects that interact with each other in the system
- A lifeline notation with an actor element symbol is used when the sequence diagram is owned by a use case

# Activation Bars



- Activation bar is the box placed on the lifeline
- It indicates that an object is active ((or instantiated) during an interaction between two objects
- The length of the rectangle indicates the duration of the interaction

# Message Arrows

- An arrow from the Message Caller to the Message Receiver specifies a message
- The message can flow in any direction; from left to right, right to left and back to the caller itself
- The description of the message should go on the arrow
- Arrow heads may change according to different message types

## ➢ Different message types;

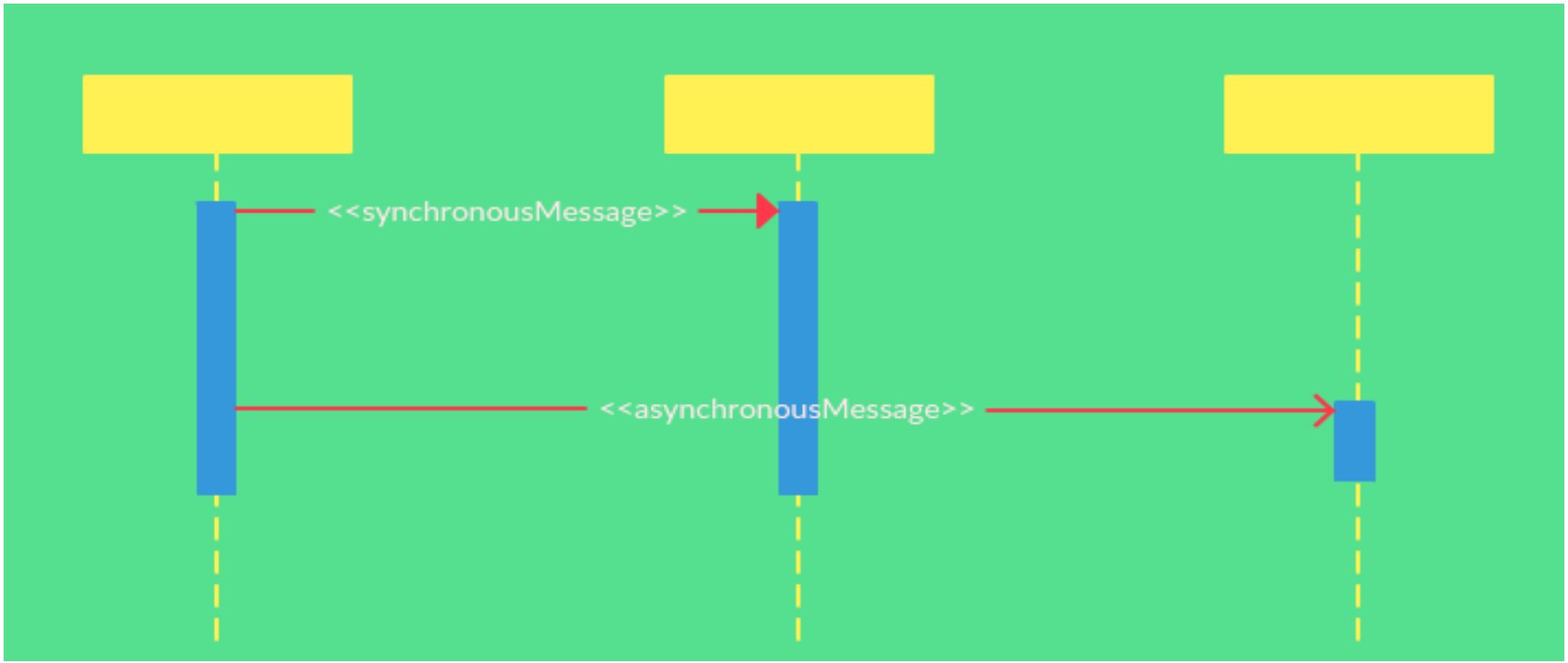
- Synchronous message
- Asynchronous message
- Return message
- Participant creation message
- Participant destruction message
- Reflexive message

- The message signature is (all parts except the message_name is optional);

*attribute = message_name (arguments): return_type*

- *A synchronous message* is used when the sender waits for the receiver to process the message and return before carrying on with another message

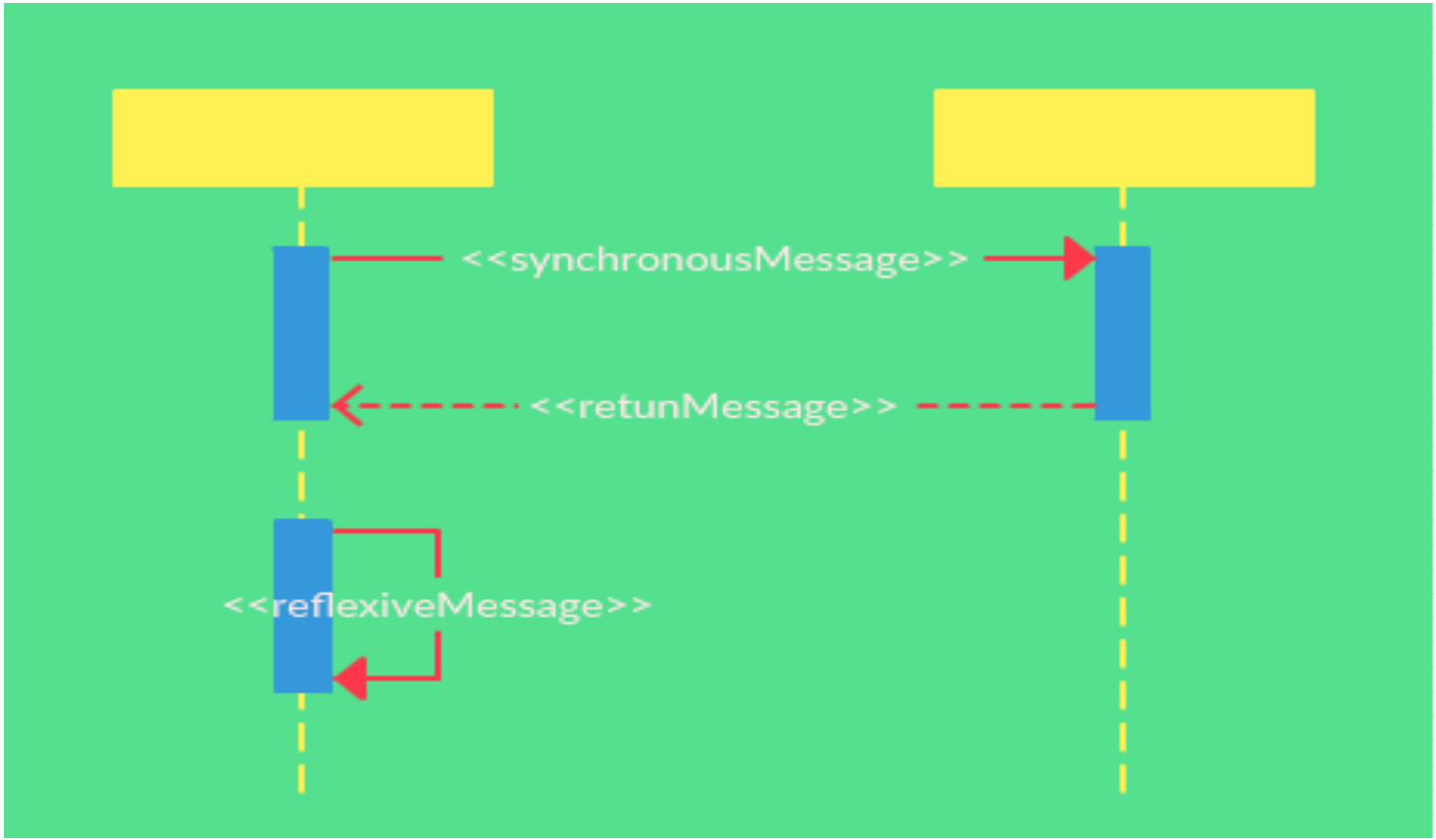
- *An asynchronous message* is used when the message caller does not wait for the receiver to process the message and return before sending other messages to other objects within the system



- *A return message* is used to indicate that the message receiver is done processing the message and is returning control over to the message caller.

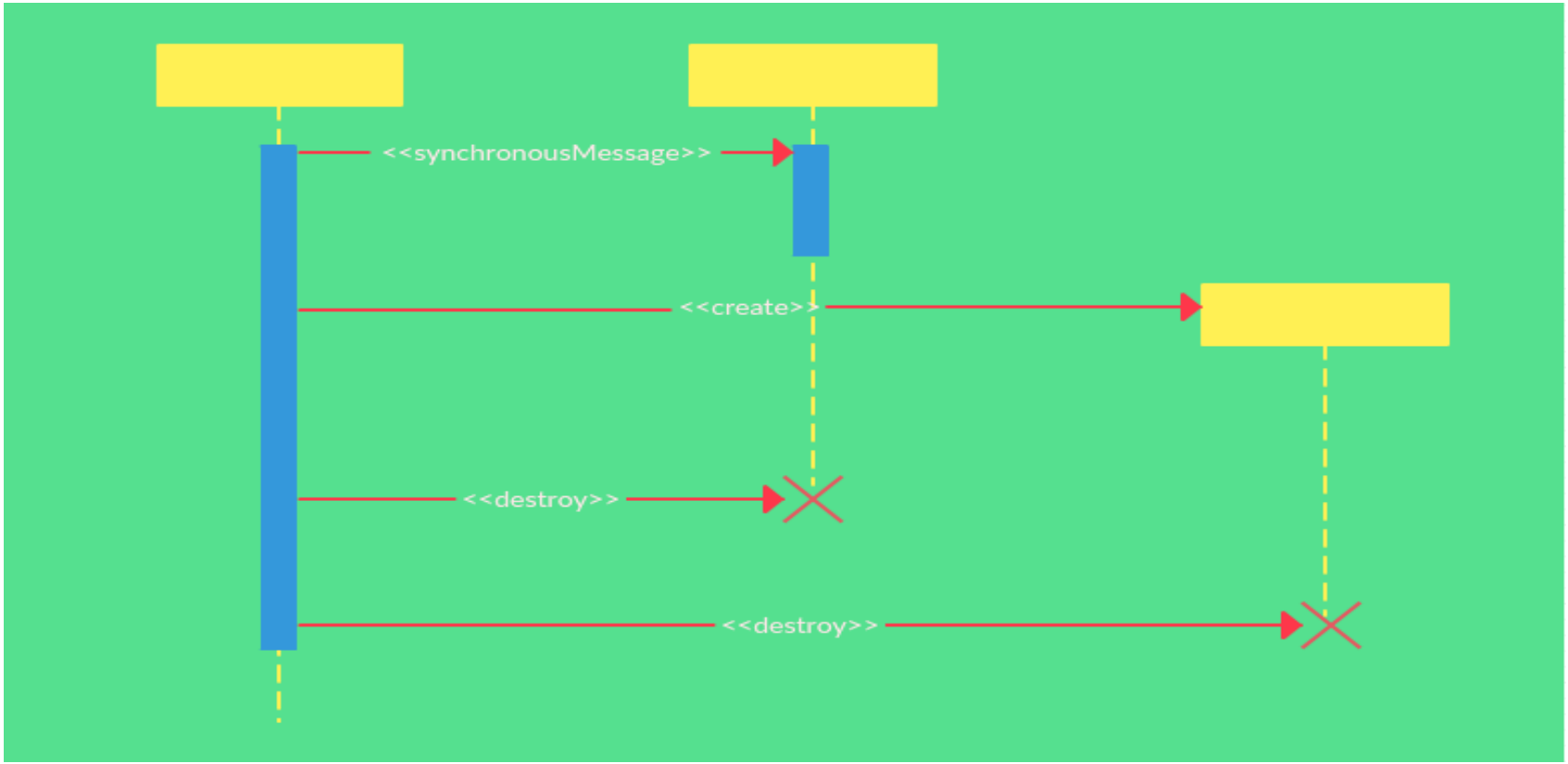
Tip: You can avoid cluttering up your diagrams by minimizing the use of return messages since the return value can be specified in the initial message arrow itself.

- When an object sends a message to itself, it is called *a reflexive message*. It is indicated with a message arrow that starts and ends at the same lifeline



- *Participant  creation message*; objects can be created in the middle of a sequence. The dropped participant box notation is used when you need to show that the particular participant did not exist until the create call was sent.

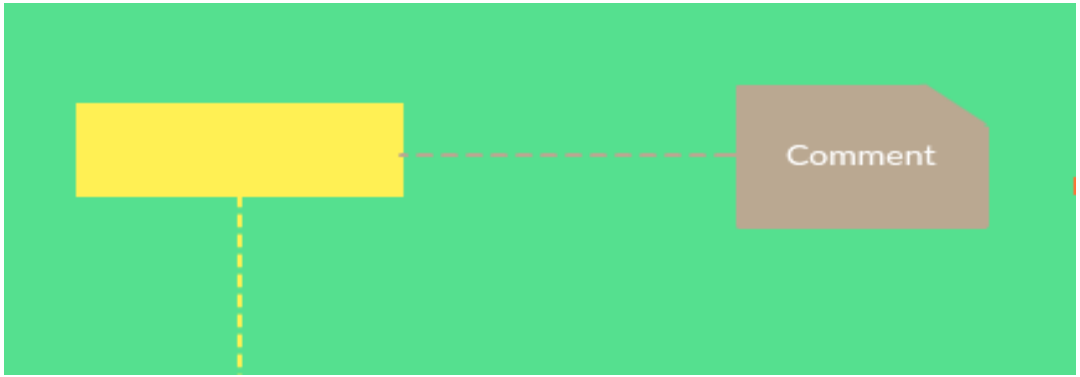
- *Participant destruction message*; participants, when no longer needed, can also be deleted from a sequence diagram. This is done by adding an ‘X’ at the end of the lifeline of the said participant.



# Comments

- UML generally permits the annotation of comments in all UML diagram types.

- The comment object is a rectangle with a folded-over corner as shown below. The comment can be linked to the related object with a dashed line.



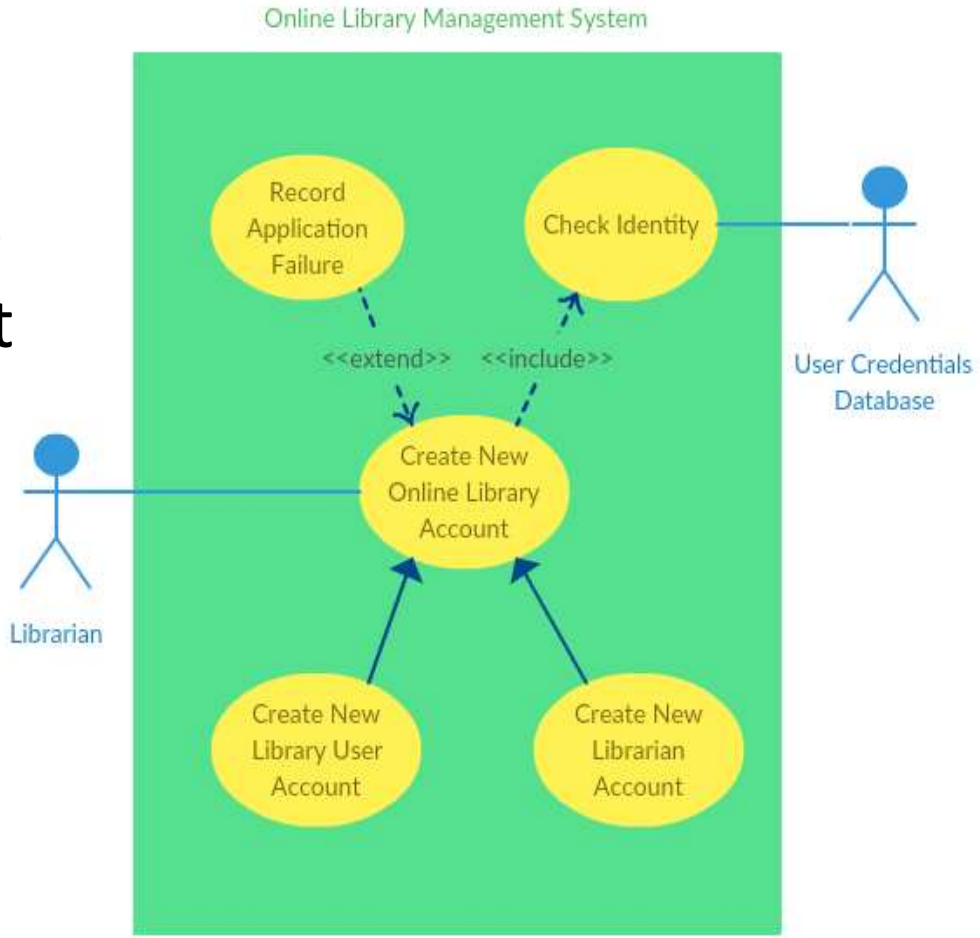
# How to Draw a Sequence Diagram

- A sequence diagram represents the scenario or flow of events in one single use case. The message flow of the sequence diagram is based on the narrative of the particular use case.

- Before you start drawing the sequence diagram or decide what interactions should be included in it, you need to ready a comprehensive description of  the particular use case.

# Example:

Use case example - Create new online library account



- From the use case diagram example of ‘Create New Online Library Account’, we will focus on the use case named ‘Create New User Account’ to draw our sequence diagram.

**Step 1**

Identify the objects or participants in the use case ‘Create New User Account’

- Librarian
- Online Library Management system
- User credentials data base
- Email system

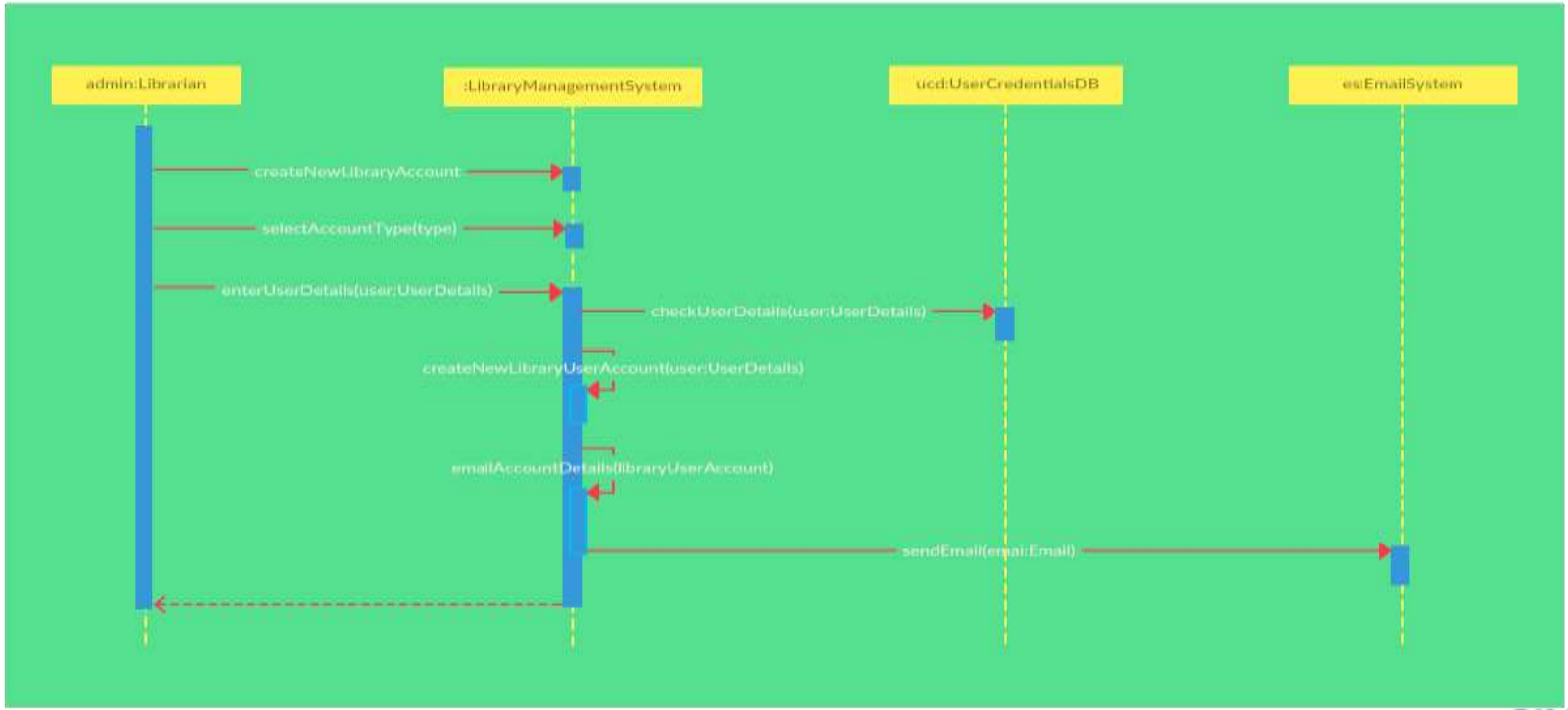
## Step 2

➢ List down the steps involved in the execution of the use case

- The librarian requires the system to create a new online library account
- The librarian selects the library user account type
- The librarian enters the user's details
- The user's details are checked using the user Credentials Database
- The new library user account is created
- A summary of the new account's details is emailed to the user

# Step 3

Identify which messages should be passed between the objects we identified earlier as the system executes these steps. Then draw the sequence diagram.



# Sequence Diagram Best Practices

➢ Manage complex interactions with sequence fragments

- Sequence fragments are used to show complex interactions such as alternative flows and loops in a more structured way.

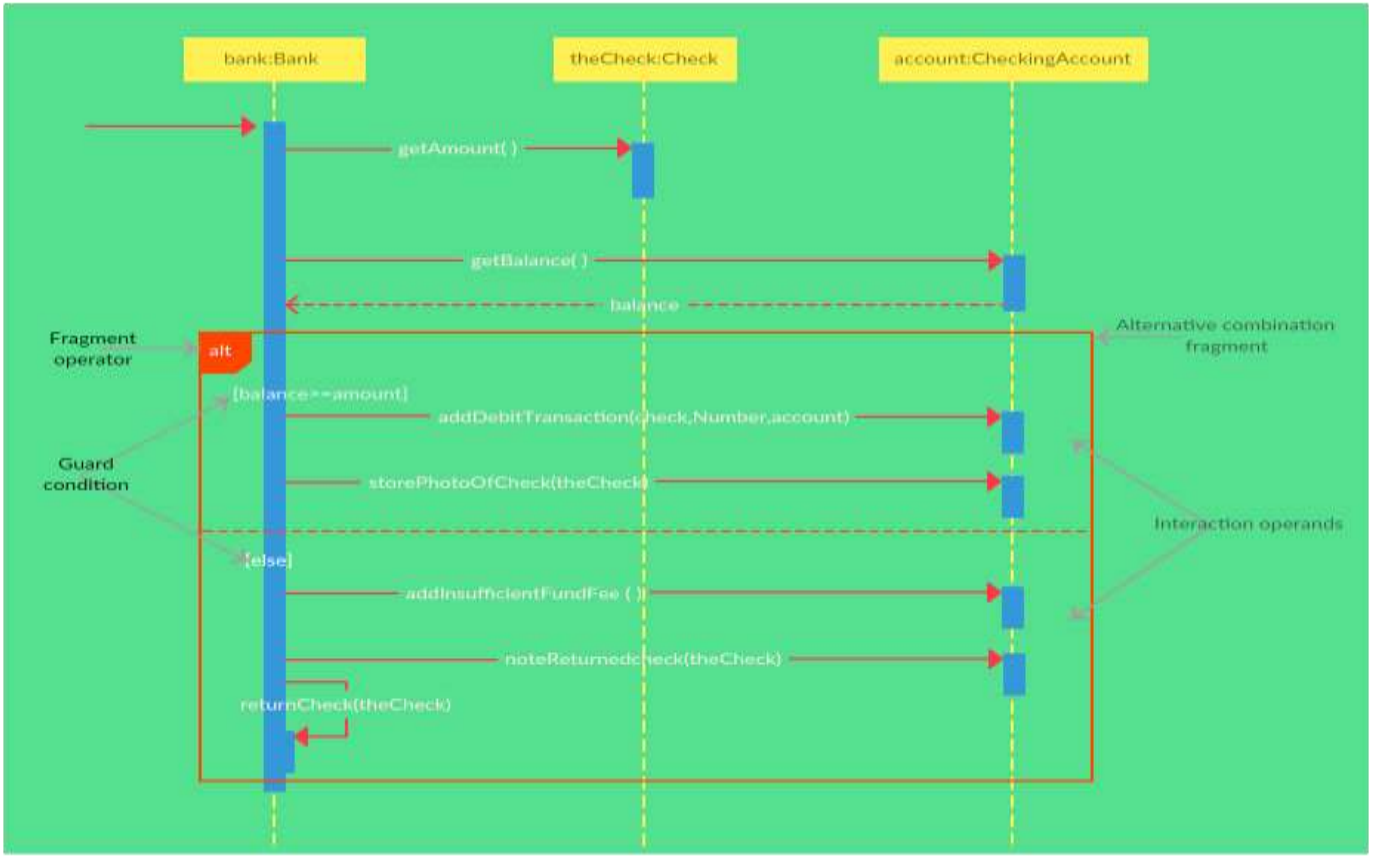
- A sequence diagram is drawn as a box that frames a section of interactions between objects in a sequence diagrams.

- On the top left corner of the fragment sits an operator. This – the fragment operator - specifies what sort of a fragment it is.

# Alternative Combination Fragment

- It is used when a choice needs to be made between two or more message sequences. It models the “if then else” logic
- It is specified by mentioning ‘alt’ inside the frame’s name box
- To show 2 or more alternatives, the frame is divided into what is called interaction operands using a dashed line
- Each operand has a guard to test against and it is placed at the top left corner of the operand

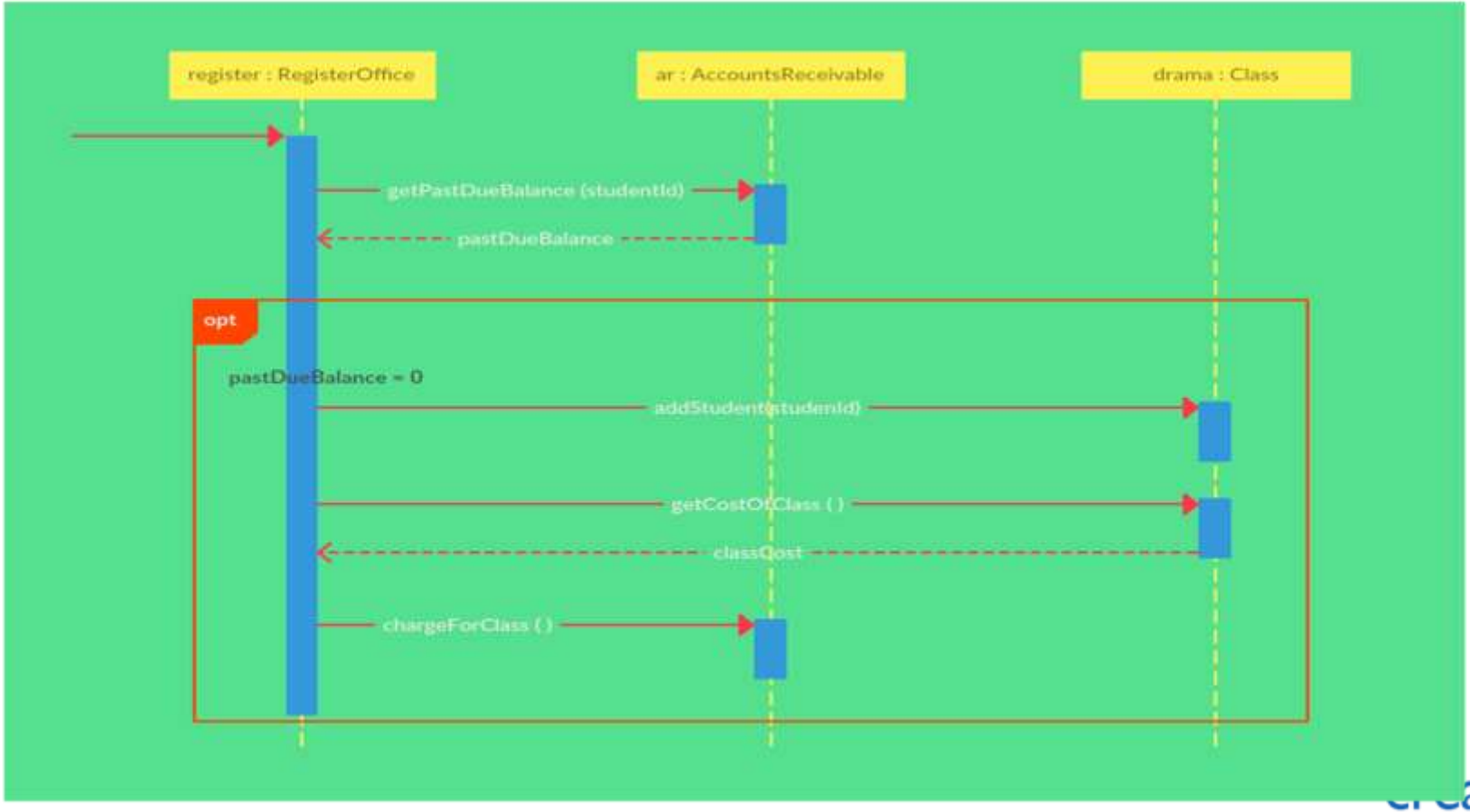
# Example of an alternative combination fragment



# Option Combination Fragment

- The option fragment is used to indicate a sequence that will only occur under a certain condition, otherwise the sequence won’t occur
- It models the “if then” statement
- It is represented with a rectangular frame where ‘opt’ is placed inside the name box
- Unlike the alternative fragment, the option fragment is not divided to operands

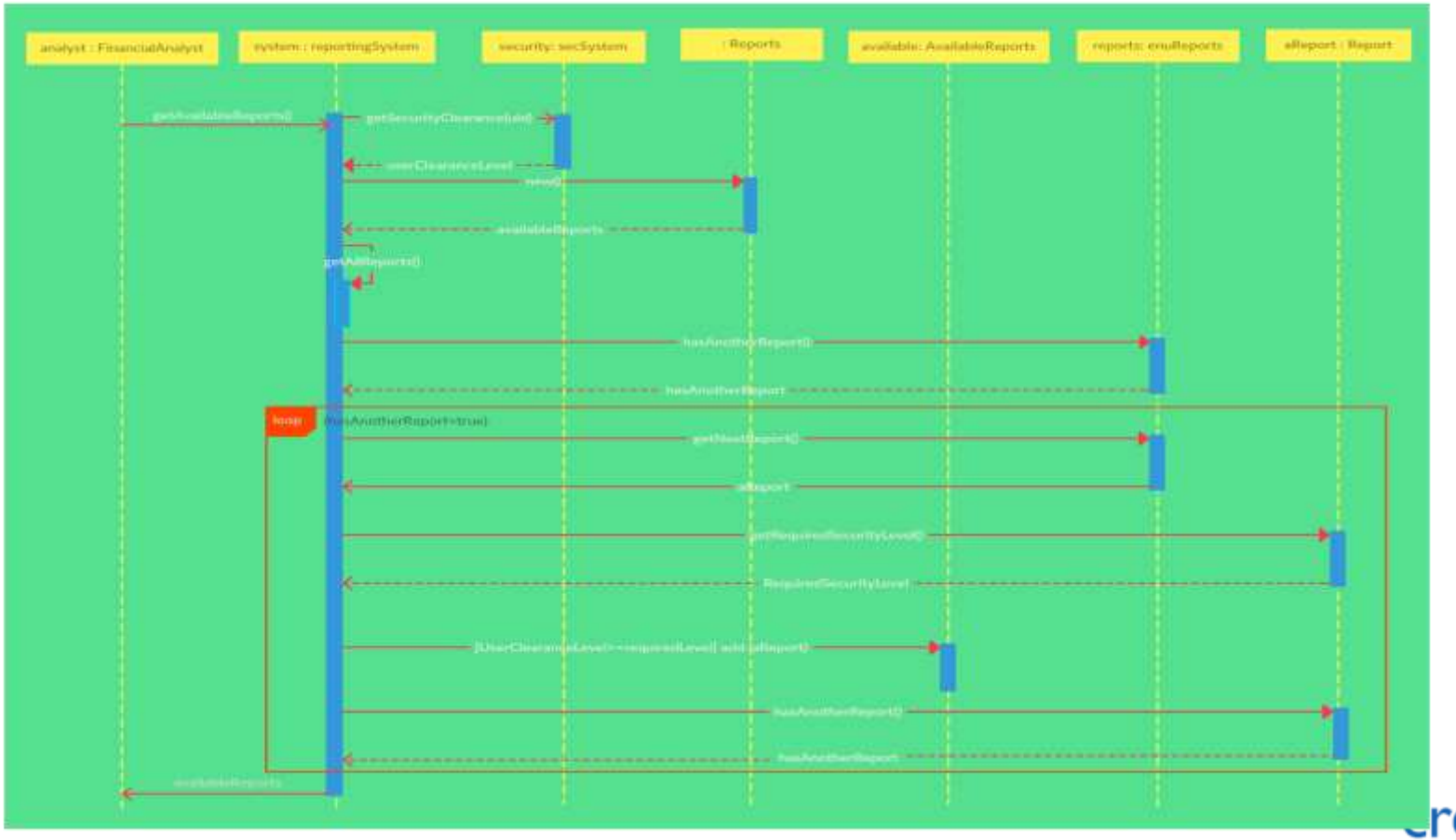
# Example of an option combination fragment



# Loop Fragment

- It is used to show a repetitive sequence
- It is drawn as a frame and specified by placing ‘loop’ in the name box
- It can be used
  - for the Boolean test
  - to test minimum iterations (the loop must execute not less than the number mentioned)
  - to test maximum iterations (the loop mustn’t execute more than the number mentioned)

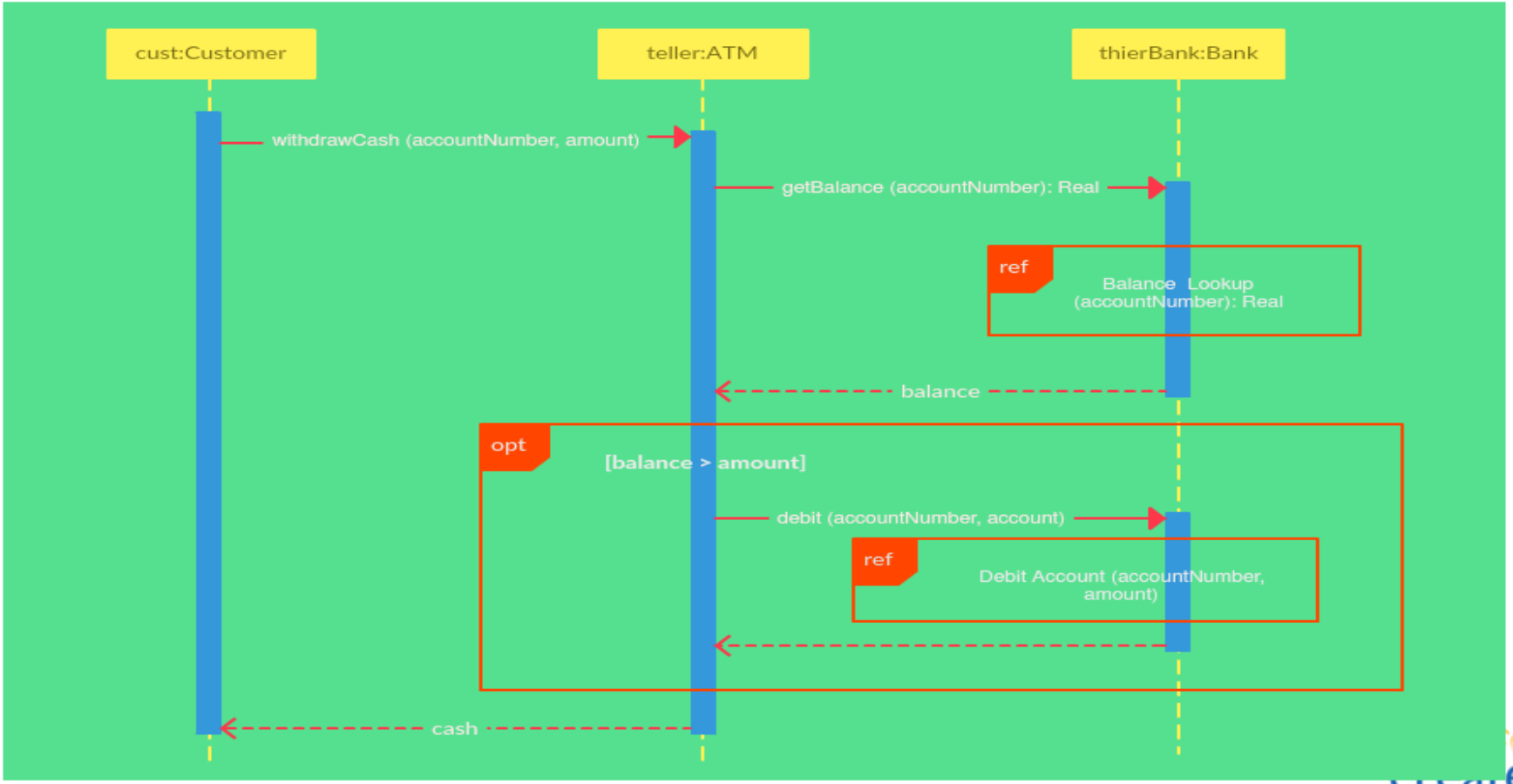
# Example of a Loop Fragment



# Reference Fragment

- It allows you to reuse or refer to a part of one sequence diagram in another
- It helps manage the size of large sequence diagrams
- Include ‘ref’ in the name box of the frame to specify the reference fragment
- Mention the name of the sequence diagram being referred to inside the frame

# Example of a reference fragment



## ➢ Draw smaller sequence diagrams that capture the essence of the use case

- Several objects and messages across  a sequence diagram clutters it up
- Convey how your system functions with a few smaller sequence diagrams
- Make sure the diagram fits on a single page and leaves room for additional comments
- Find out what is common about the diagrams and focus on the commonality when drawing

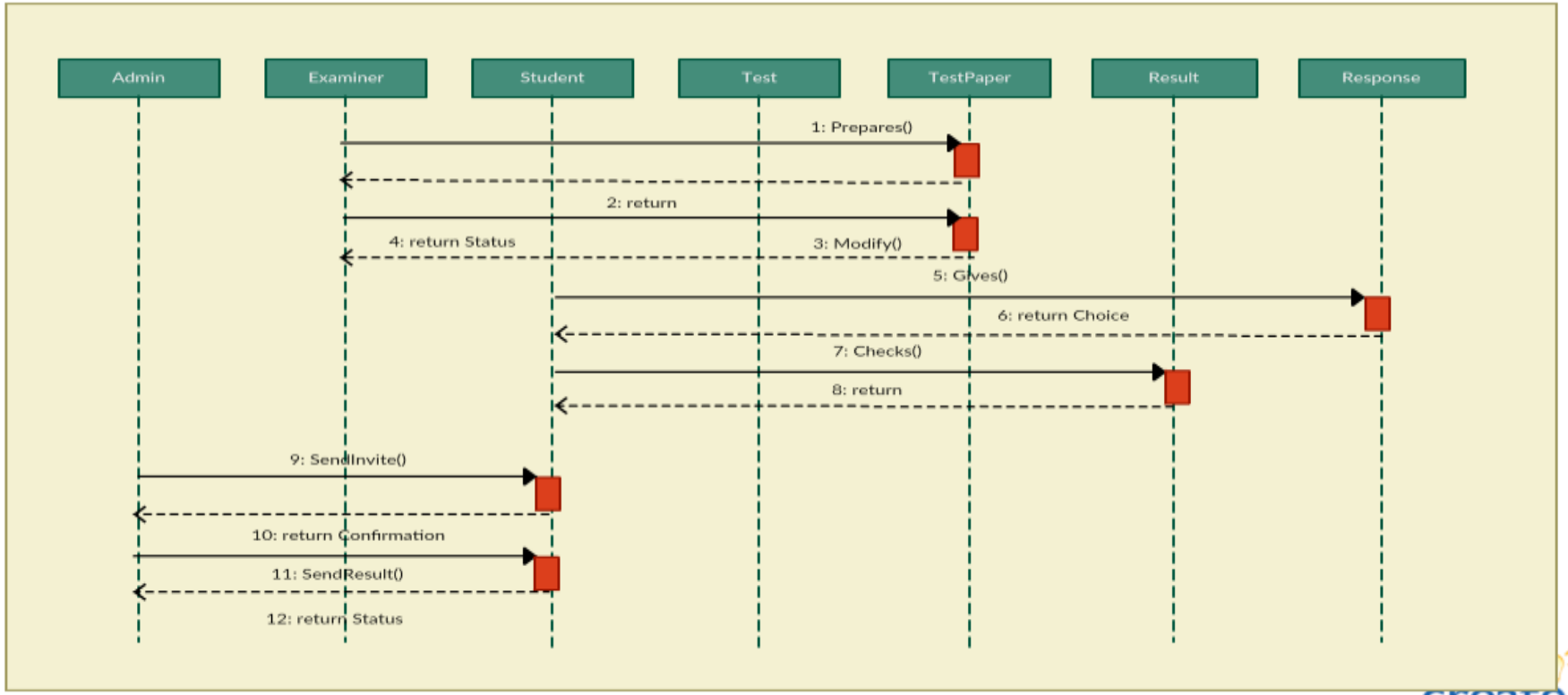
# Sequence Diagram Common Mistakes

- Adding too much detail that makes it difficult to read the diagram
- Obsolete sequence diagrams that are irrelevant when compared to the interfaces, actual  architectures of the system
- Leaving no blank space between the use case text and the message arrow
- Not considering the origins of message arrows carefully

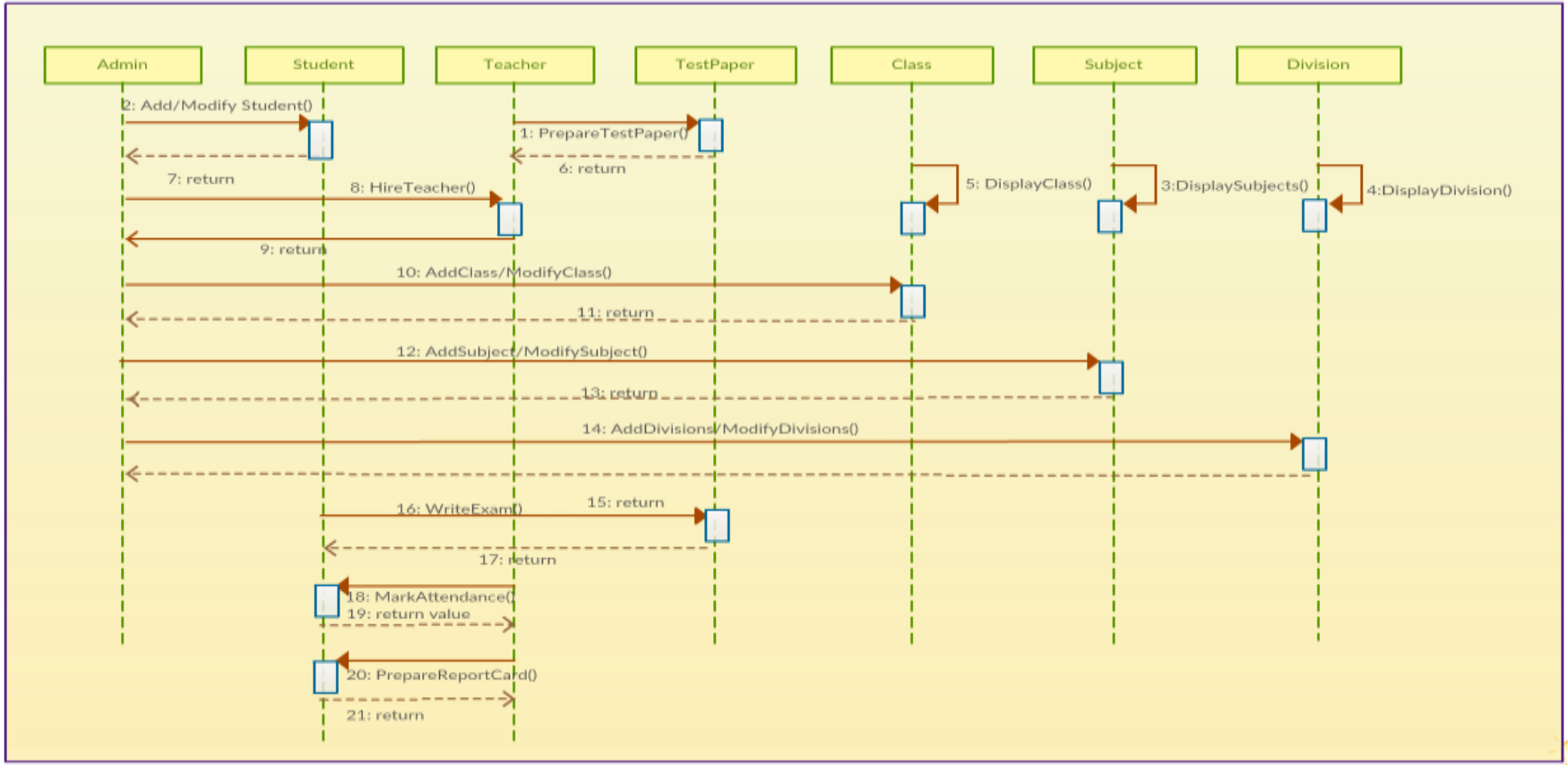
See these common mistakes explained in detail here.

# Templates and examples

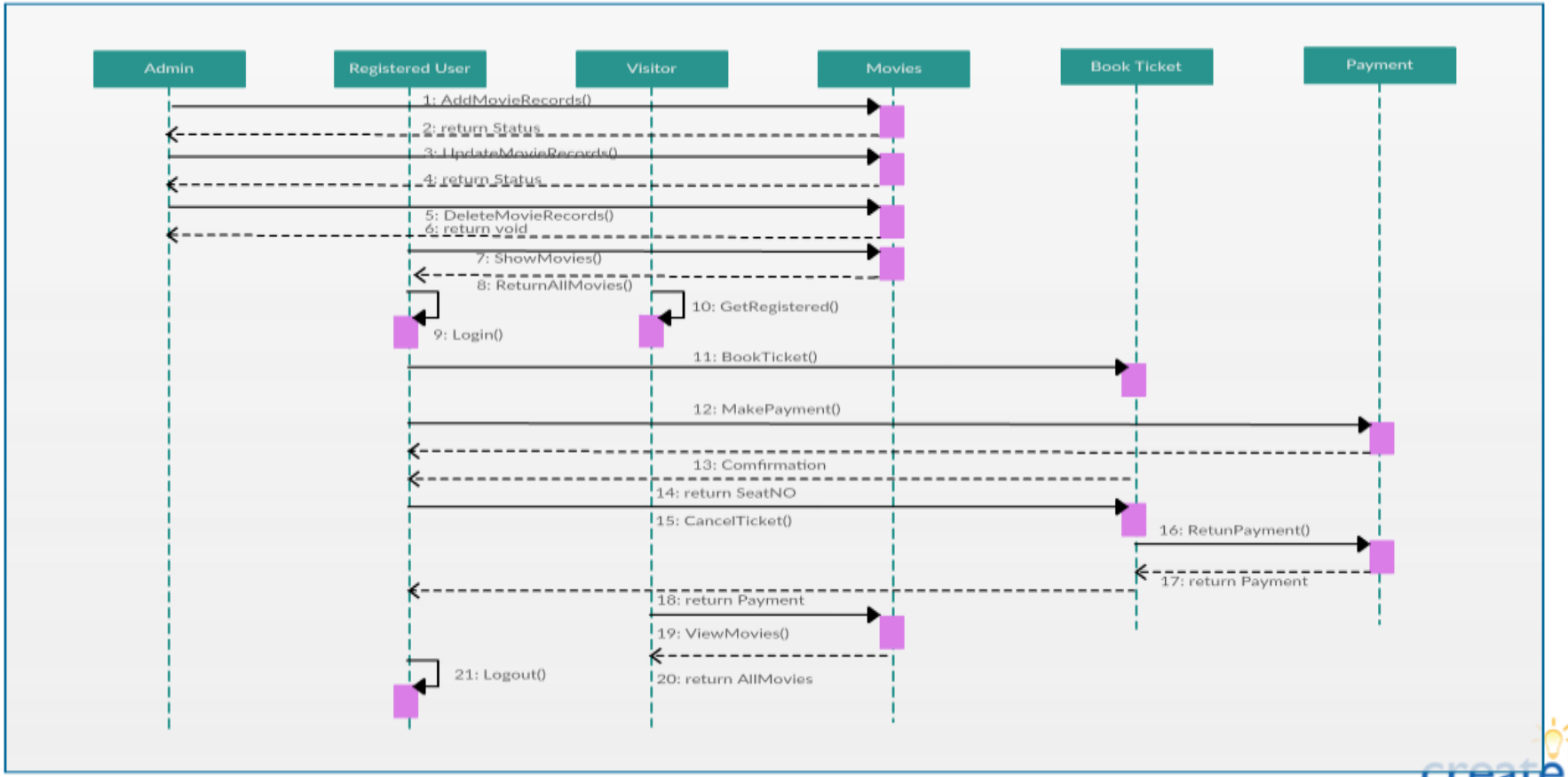
## Sequence Diagram Example of an Online Exam System



# Sequence Diagram Example of a School Management System



# Sequence Diagram Example of an Online Movie Ticket Booking System




Diagramming and Collaboration

Follow Us On




http://creately.com | @creately