

The background features a complex geometric composition. A diagonal line runs from the top-left towards the bottom-right. To the left of this line, there are several distinct patterns: a white circle at the top-left corner, a grey semi-circle, a series of concentric blue circles, a pink area with diagonal lines, a solid pink square, a grey triangle, and a series of concentric pink lines. The right side of the image is a solid blue field containing the main text.

ELEMENTS OF GRAPHICAL NOTATION OF INTERACTION DIAGRAM. BRANCHING AND EXECUTION CONDITIONS



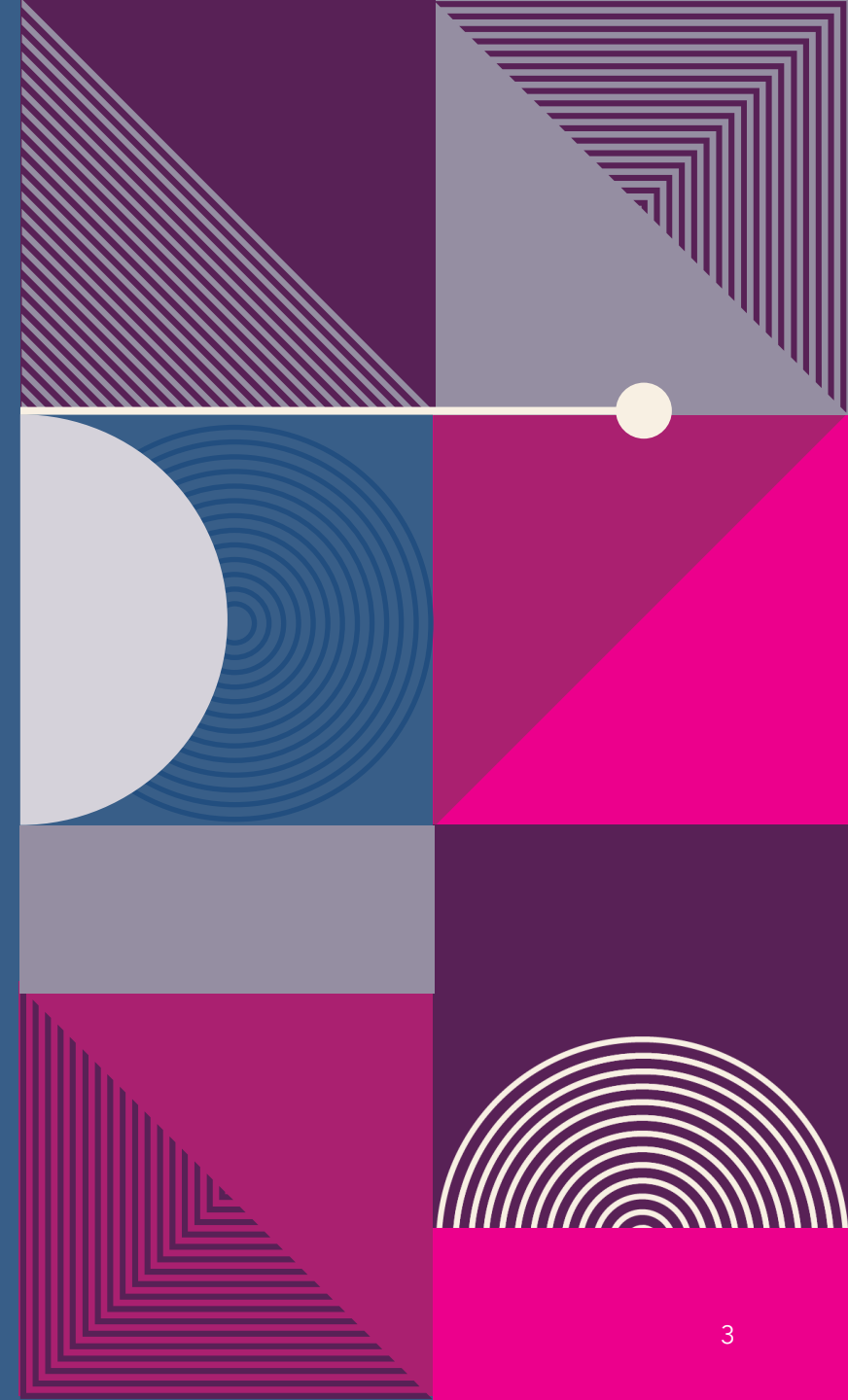
PURPOSE OF INTERACTION DIAGRAMS

The purpose of interaction diagrams is to visualize the interactive behavior of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction.

Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle.

THE PURPOSE

- To capture the dynamic behavior of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.
- To describe the interaction among objects.





FOLLOWING THINGS ARE TO BE IDENTIFIED CLEARLY BEFORE DRAWING THE INTERACTION DIAGRAM

- Objects taking part in the interaction.
- Message flows among the objects.
- The sequence in which the messages are flowing.
- Object organization.

HOW TO DRAW AN INTERACTION DIAGRAM?

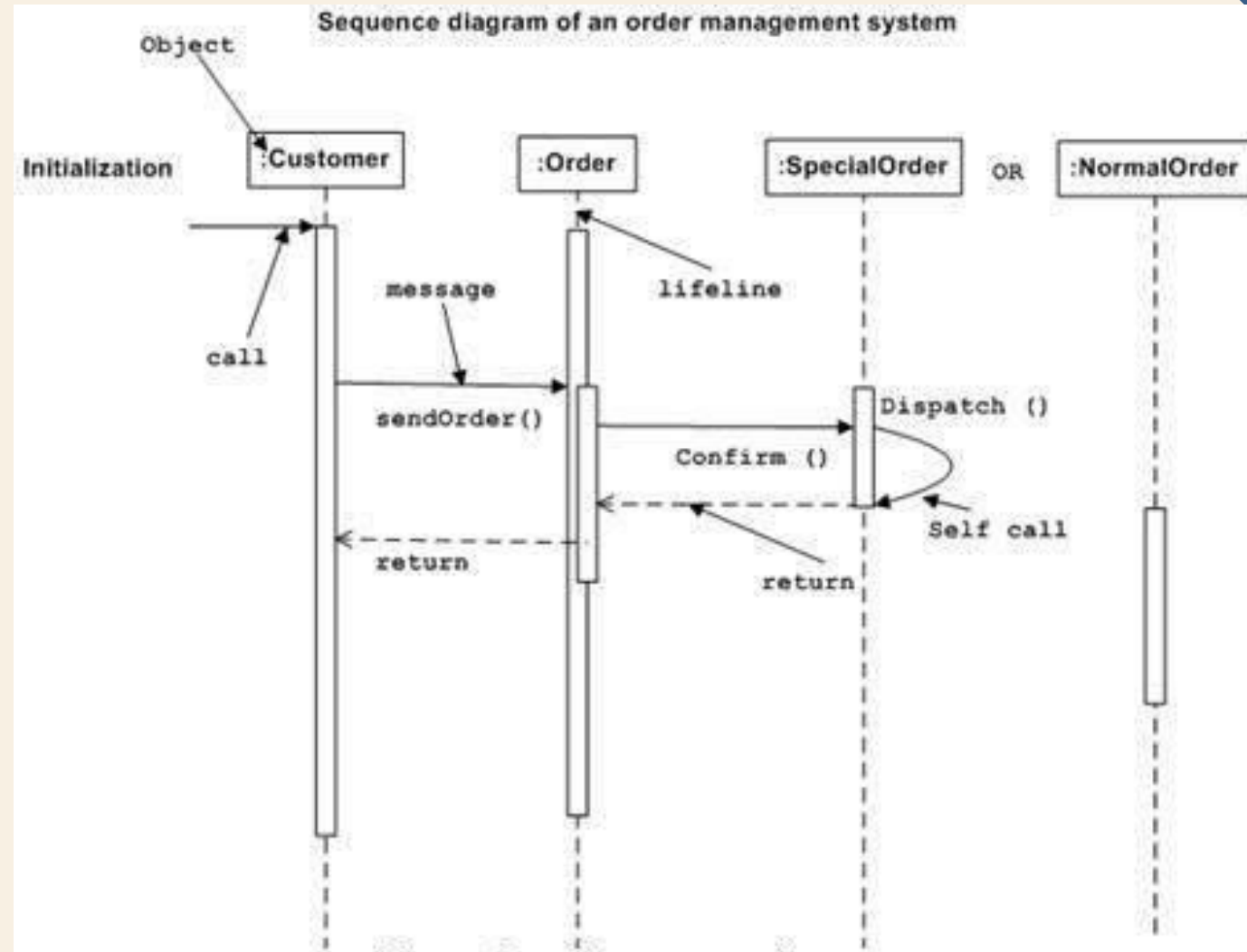
- As we have already discussed, the purpose of interaction diagrams is to capture the dynamic aspect of a system. So, to capture the dynamic aspect, we need to understand what a dynamic aspect is and how it is visualized. Dynamic aspect can be defined as the snapshot of the running system at a particular moment
- We have two types of interaction diagrams in UML. One is the sequence diagram, and the other is the communication diagram. The sequence diagram captures the time sequence of the message flow from one object to another and the communication diagram describes the organization of objects in a system taking part in the message flow.



THE SEQUENCE DIAGRAM

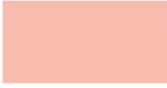

- The sequence diagram has four objects (Customer, Order, SpecialOrder and NormalOrder).
- The following diagram shows the message sequence for *SpecialOrder* object and the same can be used in case of *NormalOrder* object. It is important to understand the time sequence of message flows. The message flow is nothing but a method call of an object.

- The first call is *sendOrder()* which is a method of *Order* object. The next call is *confirm()* which is a method of *SpecialOrder* object and the last call is *Dispatch()* which is a method of *SpecialOrder* object. The following diagram mainly describes the method calls from one object to another, and this is also the actual scenario when the system is running.



A SEQUENCE DIAGRAM PROVIDES THE FOLLOWING BENEFITS:

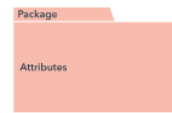
- They're easy to maintain and generate.
- They're easy to update according to changes in a system.
- They allow for reverse and forward engineering.
- Sequence diagrams can also have these possible downsides:
- They can become complex, with too many lifelines and varied notations.
- They're easy to produce incorrectly and depend on your sequence being entered correctly.

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.



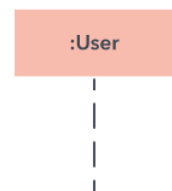
Actor symbol

Shows entities that interact with or are external to the system.



Package symbol

Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.



Lifeline symbol

Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.



Option loop symbol




Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.






Alternative symbol

Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

BASIC SYMBOLS AND COMPONENTS

Symbol	Name	Description
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.

	Asynchronous create message symbol	Represented by a dashed line with a lined arrowhead. This message creates a new object.
	Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

COMMON MESSAGE SYMBOLS

COMMUNICATION DIAGRAM

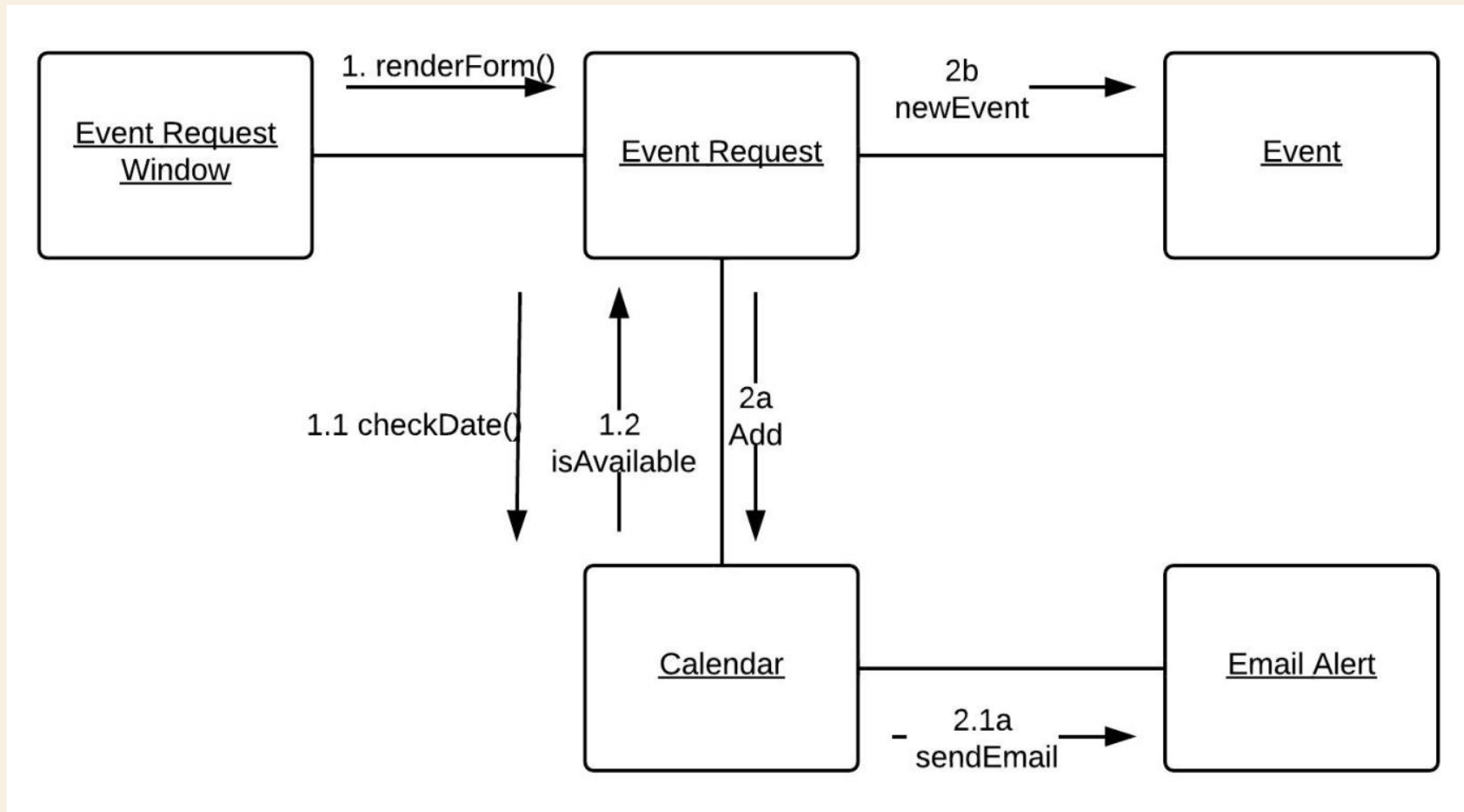
- A communication diagram offers the same information as a [sequence diagram](#), but while a sequence diagram emphasizes the time and order of events, a communication diagram emphasizes the messages exchanged between objects in an application. Sequence diagrams can fall short of offering the "big picture."

BASIC COMPONENTS OF A COMMUNICATION DIAGRAM

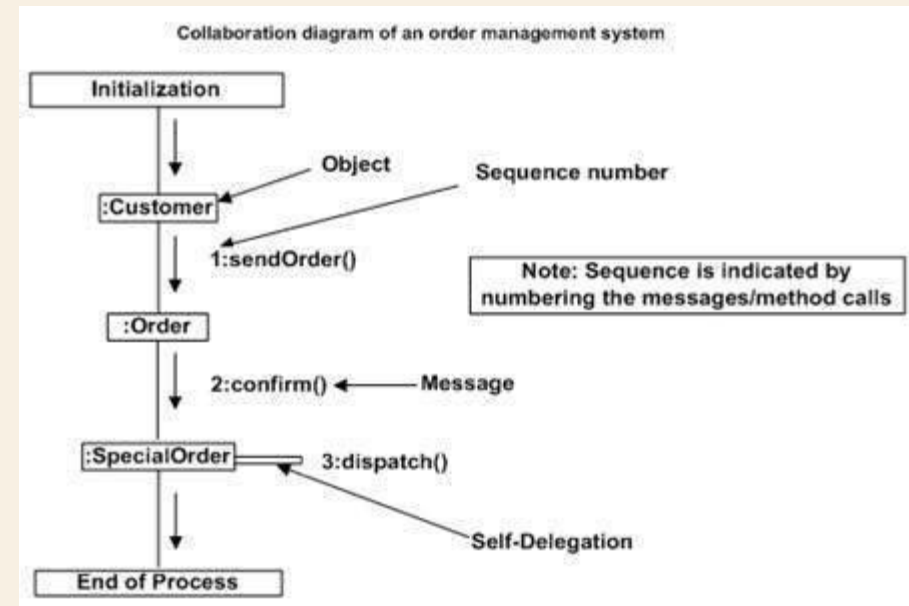
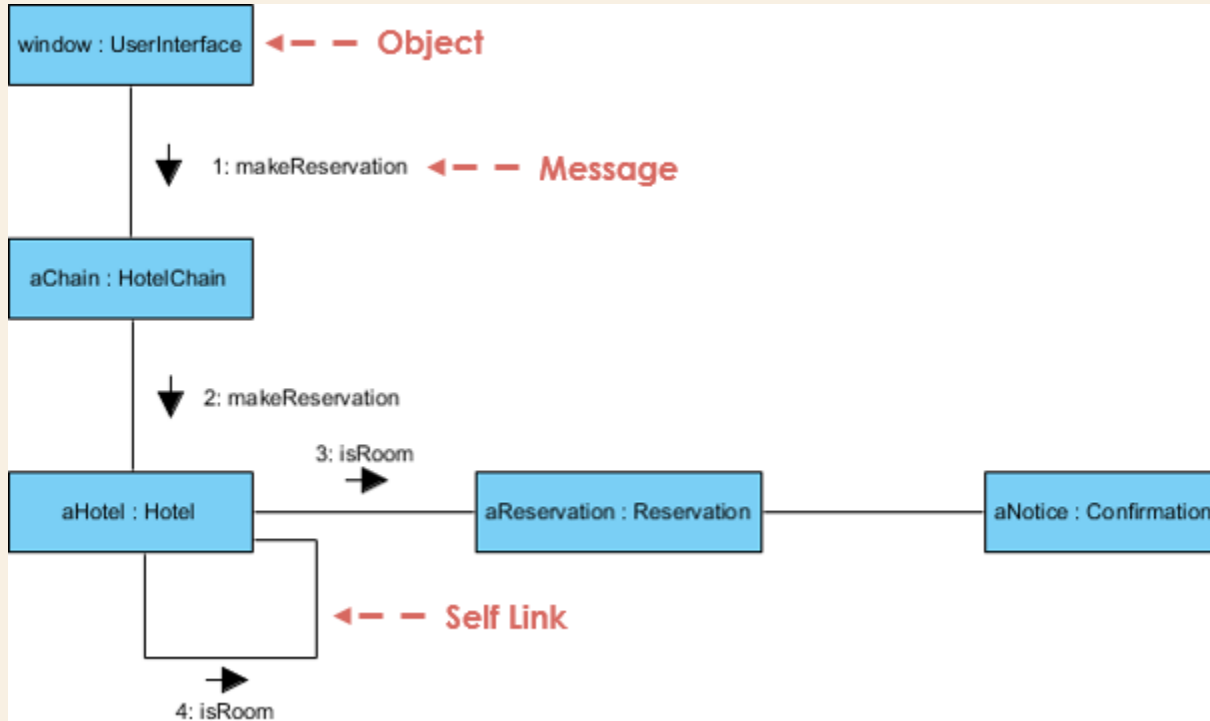
- Communication diagrams offer benefits similar to sequence diagrams, but they will offer a better understanding of how components communicate and interact with each other rather than solely emphasizing the sequence of events. They can be a useful reference for businesses, organizations, and engineers who need to visualize and understand the physical communications within a program.

THE SYMBOLS AND NOTATIONS USED IN COMMUNICATION DIAGRAMS ARE THE SAME NOTATIONS FOR SEQUENCE DIAGRAMS.

- Rectangles represent objects that make up the application.
- Lines between class instances represent the relationships between different parts of the application.
- Arrows represent the messages that are sent between objects.
- Numbering lets you know in what order the messages are sent and how many messages are required to finish a process.



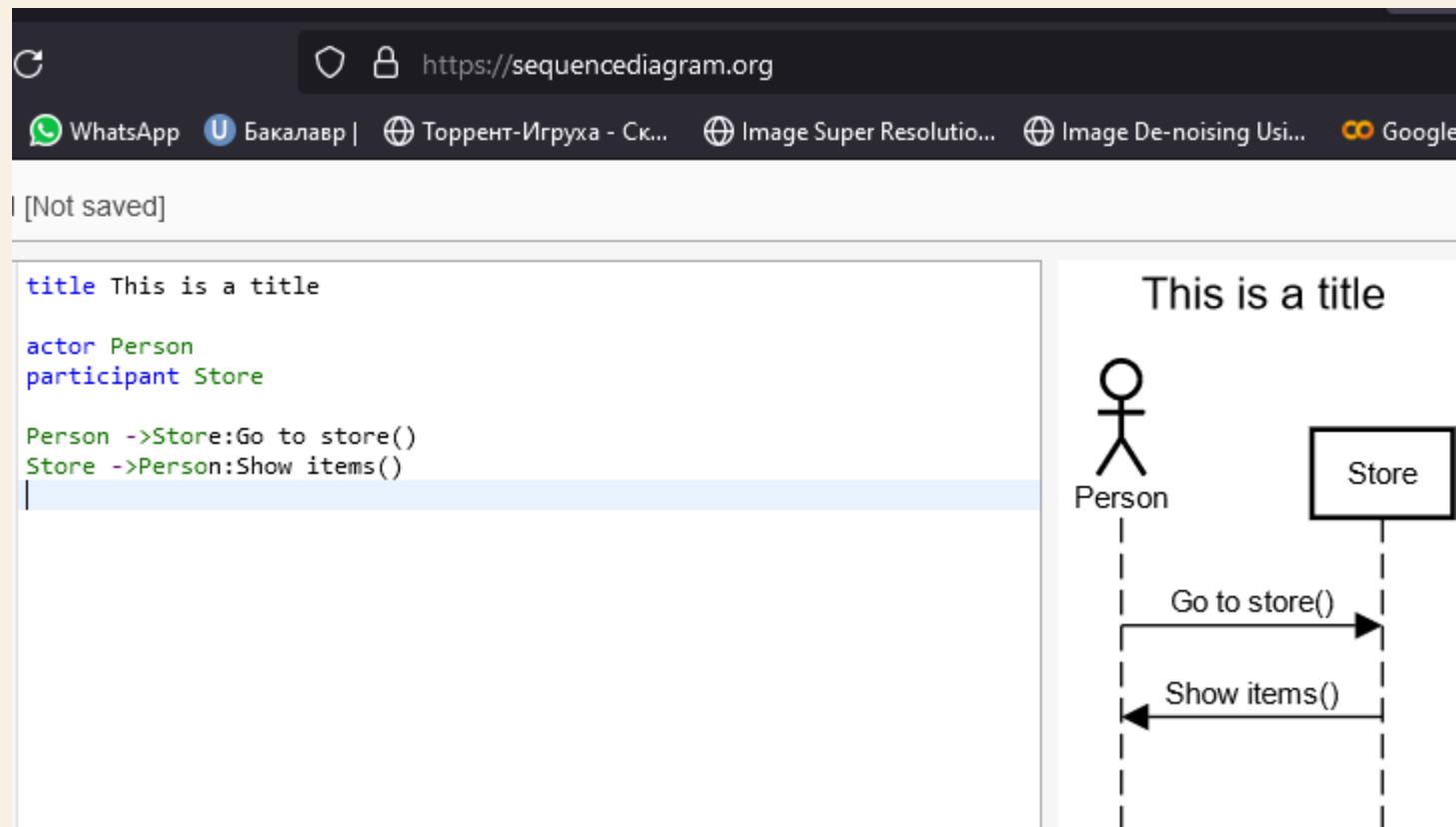
- The numbers on each line represent the order and options in which they are activated. We know that some actions happen concurrently because of the use of letters.



Examples of a communication diagram

HOW TO EASILY CREATE SEQUENCE DIAGRAMS

- <https://sequencediagram.org/>



EXERCISE 1

- Create a **SIMPLE** sequence diagram with no more than 3 objects modeling an everyday scenario
 - 10 minutes

