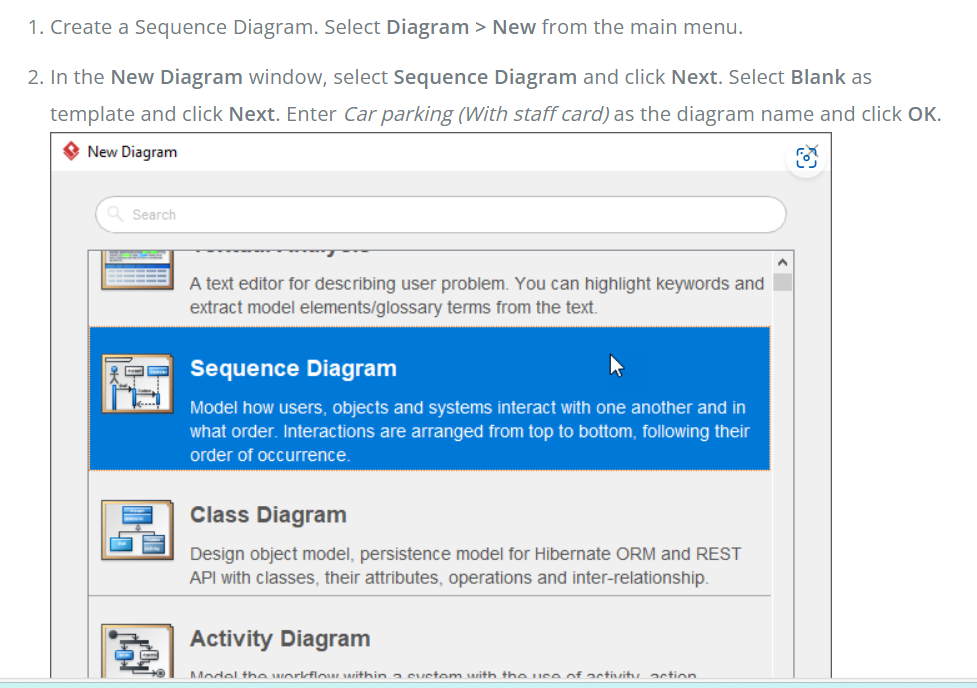
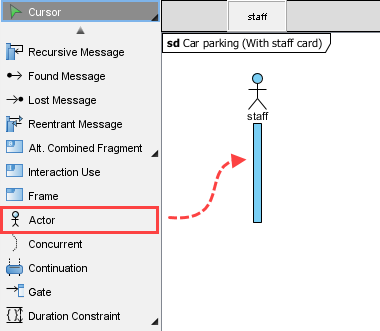
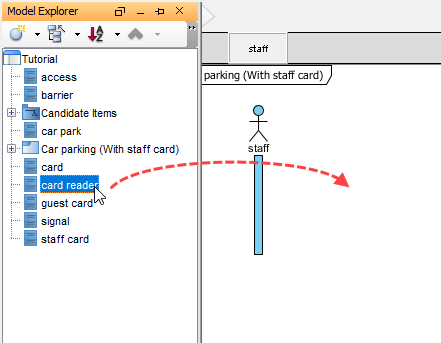
 Let’s create a sequence diagram to visualize the case when a company staff drives his car into the car park.



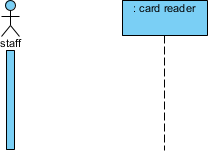
The scenario starts from a driver who inserts his staff card into the card reader. Let’s create an actor*Staff*. To create an actor, select the **Actor** tool from the diagram toolbar and drag it onto the diagram.



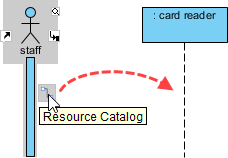
We need to show card reader as an object. Instead of creating a lifeline, let’s create an object instance from the *card reader* class. There are several ways to achieve this. One of the ways is to reuse the class via the **Model Explorer**. Click on the **Model Explorer** tab attached to the left of the application window (or select **View > Panes > Model Explorer** if the tab doesn’t present). Then, drag the class card reader onto the diagram.



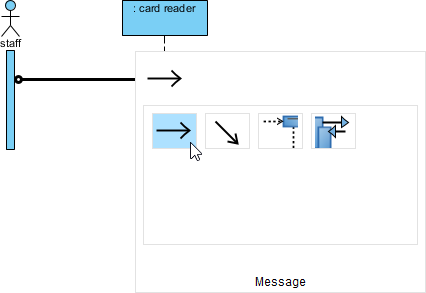
1. Select **LifeLine** to create when you are prompted to choose a type.
2. The lifeline is automatically named ‘card reader’, following the name of classifier. Since the name is optional, let’s remove it. This also makes the diagram easier to read.



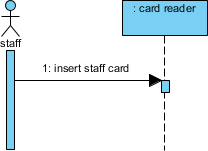
Connect *staff* with *: card reader*. Move your mouse pointer over the actor *staff*. Press on the **Resource Catalog** button and drag it to the lifeline *: card reader*.



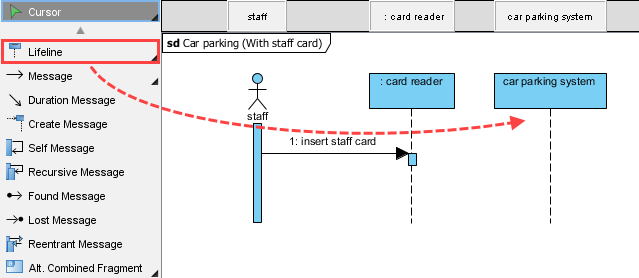
Release your mouse pointer and select **Message** from **Resource Catalog**.



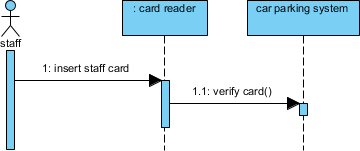
Enter *insert staff card* to be the name of message.



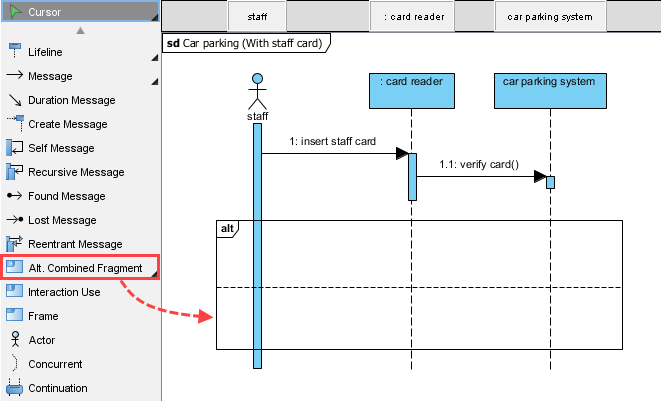
As an input device, card reader receives the card and then request the system to verify it. Let’s create a lifeline for the system. Since we don’t have such a class, we just create the lifeline directly. Drag **Lifeline** from the diagram toolbar to the diagram. Name the lifeline *car parking system*.



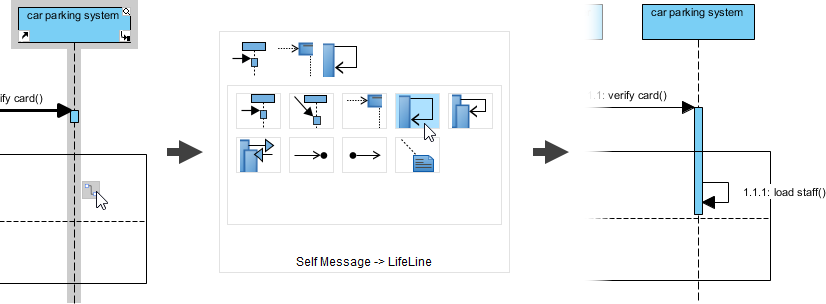
Create a message *verify card()* from *: card reader* to *car parking system*.



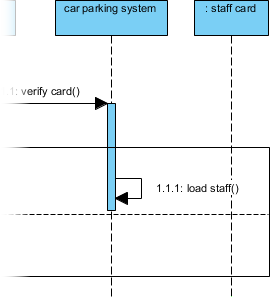
here are two possible results: success or fail. We shall model both cases. In a UML sequence diagram, an alternative combined fragment can be used to model multiple possible flows. Create an alternative combined fragment that covers the actor and lifelines. We will resize it to cover more lifelines as more and more lifelines are added.



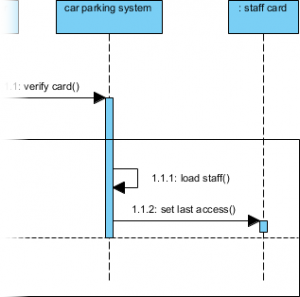
If success, the system will load the record of staff card. Create a self-message *load staff()*. To do this, move your mouse pointer over *car parking system* and click on **Resource Catalog**, then select **Self Message -> LifeLine** from **Resource Catalog**. Enter the name of the self-message.



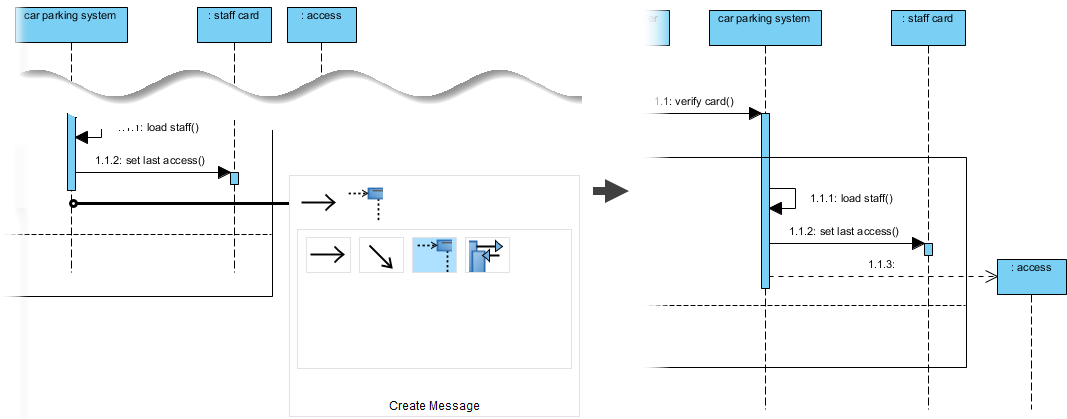
Follow step 4 to visualize the class *staff card* as a lifeline. Again, remove its name.



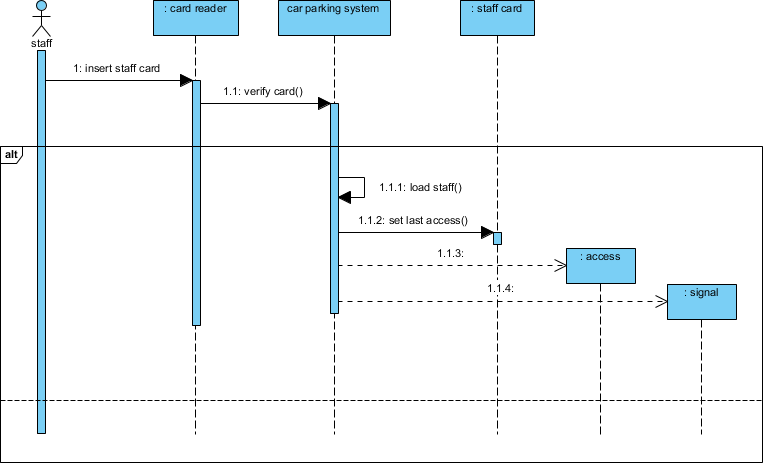
Let’s say the last access time will be set to the record of staff card. Represent this by a set message *set last access()*.



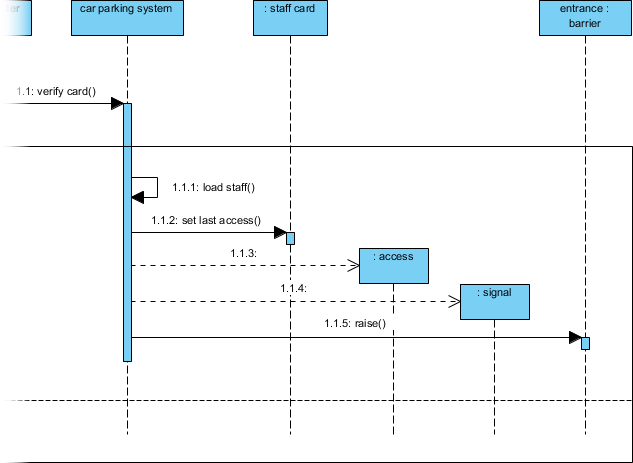
A record of access would be created. Let’s visualize the *access* class on the diagram as a lifeline, and then connect to it with a create message. Don’t forget to enlarge the alternative combined fragment to cover the lifeline.



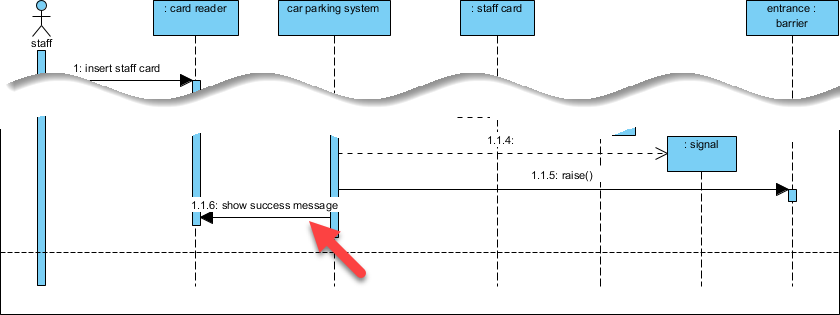
A signal would also be created to trigger the barrier to raise. Let’s visualize the *signal* class on the diagram and then connect to it with a create message. Up to now, your sequence diagram should look similar to the figure below.



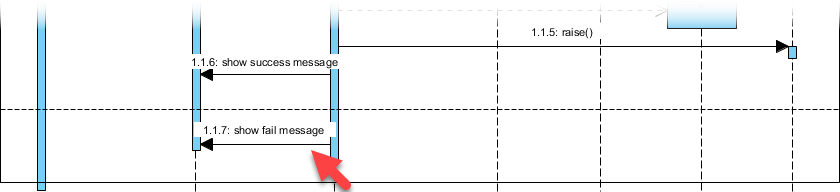
Raise the barrier. Visualize the *barrier* class as a lifeline and connect to it with a *raise()* message. Here we name the lifeline *entrance* because there is a barrier in at both the entrance and exit while we are referring to the one at entrance in this diagram.



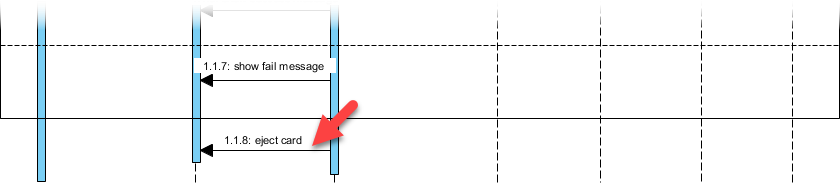
Finally, request the card reader to show a message to indicate the success verification of staff card. Create a sequence message show success message that connects between *car parking system* and *: card reader*.



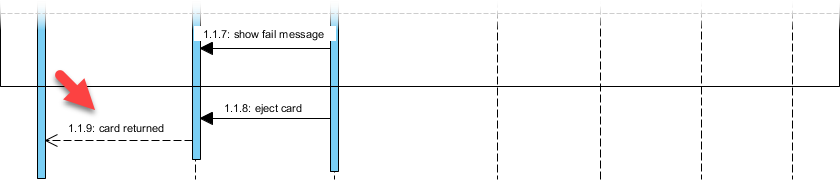
What will happen when the card cannot be verified? Let’s model that at the lower part of the alt. combined fragment. Let’s say the system will request the card reader to show a verification fail message. Create such a message in the diagram.



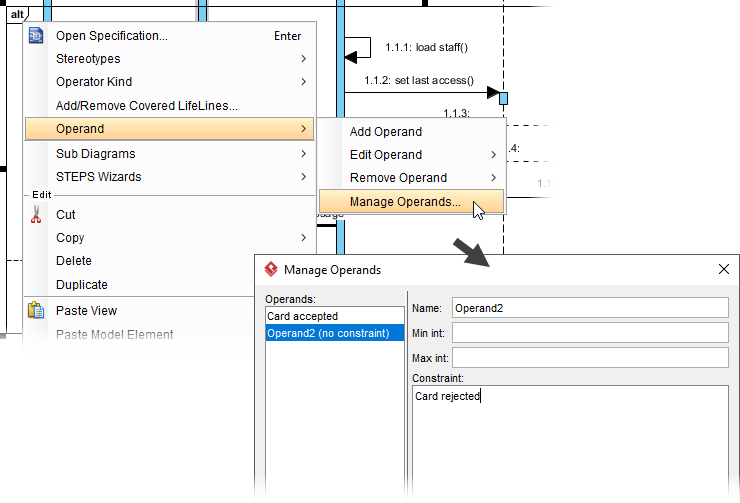
No matter success or fail, the staff card will be ejected after the verification process. Let’s create the sequence message *eject card* that connects the *car parking system* and the *: card reader*.



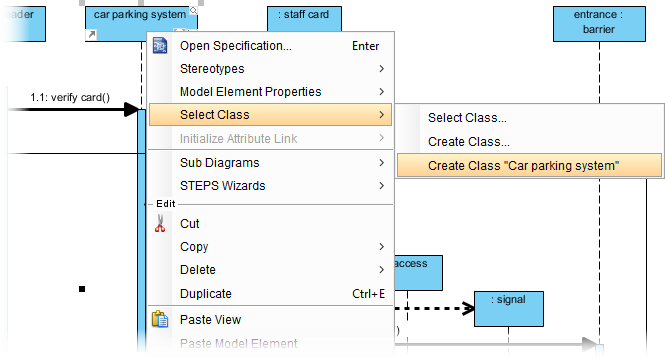
Finally, create a return message card returned from *: card reader* to the *actor* staff. To do this, create the message, and then right click on it and select **Type (Unspecified) > Reply** from the popup menu.



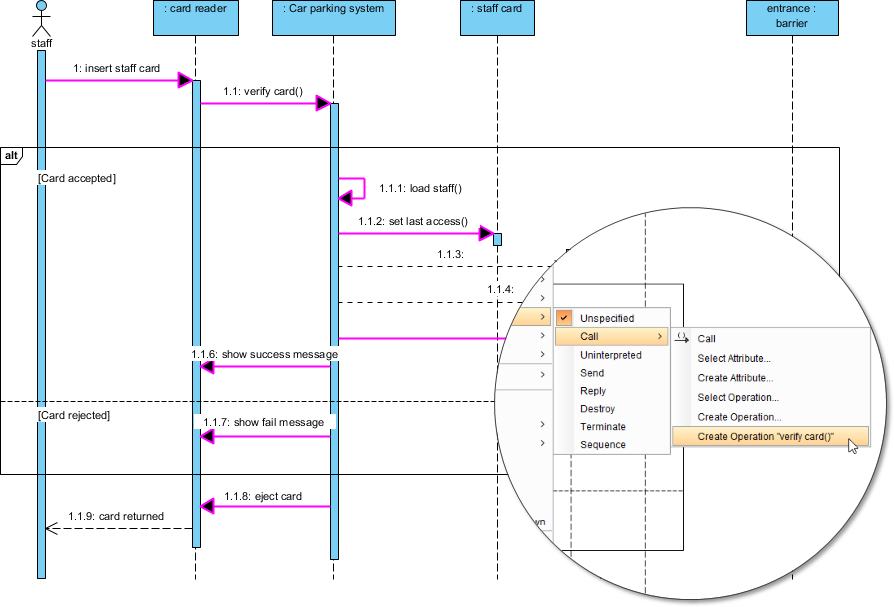
The two compartments of the alt. combined fragment are known as operands. Enter the conditions of the operands as constraints. Right click the label **alt** at the top left of the combined fragment and select **Operand > Manage Operands** from the popup menu. In the popup window, set the constraint of the two operands as *Card accepted* and *Card rejected* respectively.



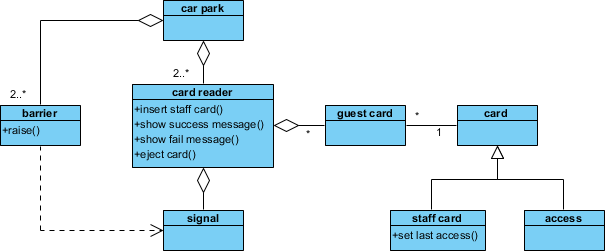
Let’s create operations from the sequence messages. First, create a class from the lifeline car parking system. Right click on the lifeline and select Select **Class > Create Class “car parking system”** from the popup menu.



Refer to the following image. For each of the highlighted connectors, right click on it and select **Type > Call > Create Operation** from the popup menu.



Go back to the class diagram. Operations are automatically presented in the class model.



By creating more and more sequence diagrams for different scenarios, the class model will be gradually enriched by including all the operations and attributes the system needs.

Ref: [UML Tutorial: From Problem Description to Models](https://www.visual-paradigm.com/tutorials/from-problem-description-to-models/)