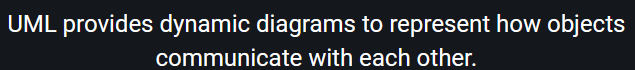
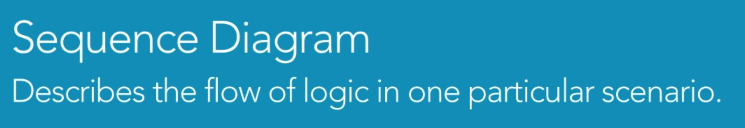
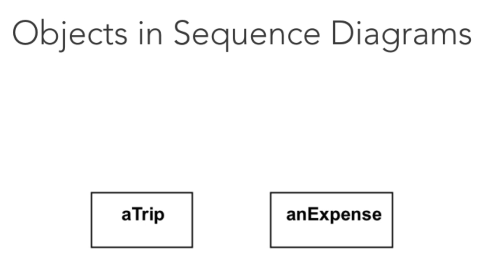
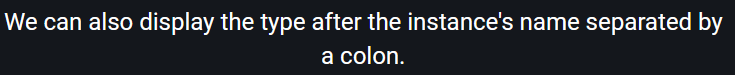
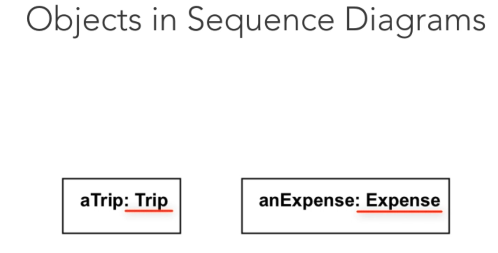
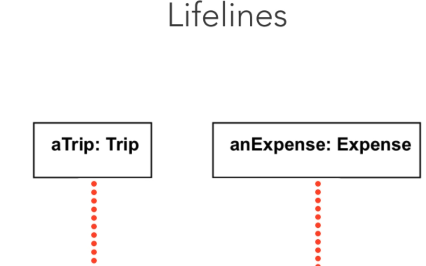
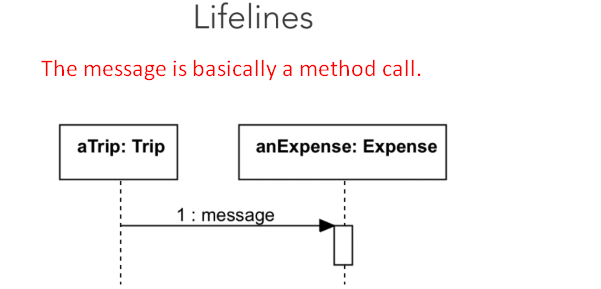
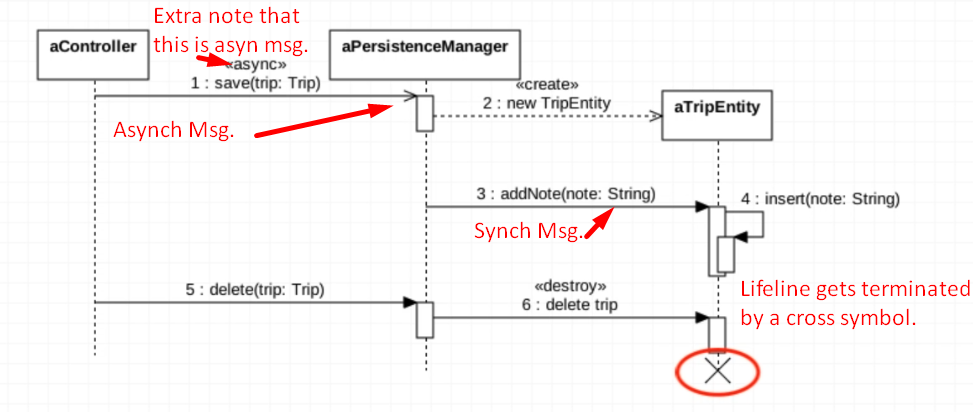
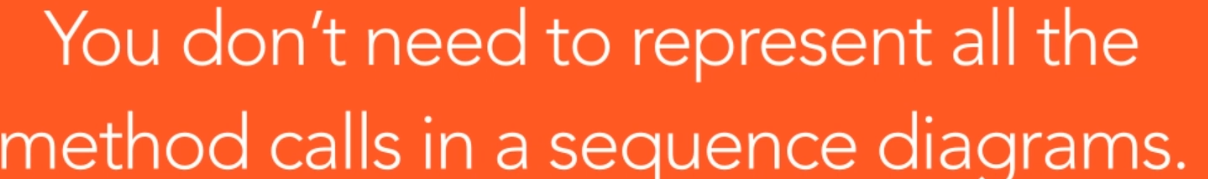
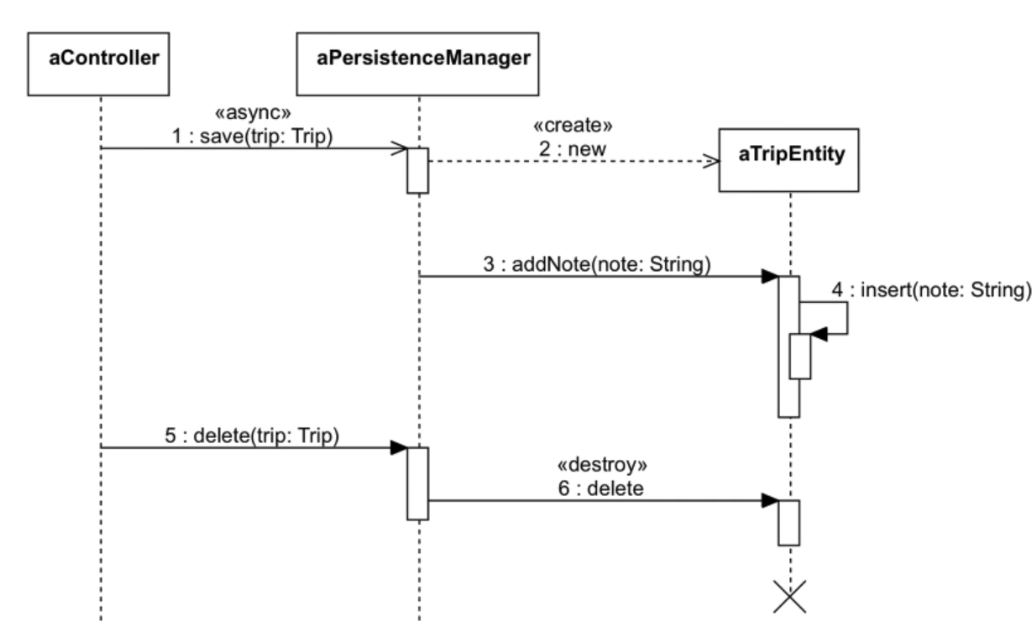
1. 
2. Use Case Diagram and Class Diagrams are **static diagrams**. They are great at representing the structure of our system.
3. But what if
   1. We need to show how the objects interact with each other?
   2. When objects are created and for how long are they around?
4. Static diagrams (Use Case Diagram, Class Diagram) can’t answer these questions.
5. **Solution**:
   1. Sequence Diagram.
6. Static diagram can’t answer these questions.
7. 
8.  
9. Sequence diagram starts by drawing boxes at the top of the page.
10. Each box represents an object.
11. Since these are objects so we name them differently.
12.  
13. 
14. 
15. The lifeline of an object is represented by the dotted lines beneath each box. The line shows the time the instance exists during the scenario. 
16. The sequence diagram also lets us show that the message is sent from one object to the other. 
17. Download UML Modelling Tool from <https://staruml.io/>
18. Let’s assume we have persistence manager object.
    1. This object is responsible to create and store Trip objects into local database.
    2. First I add the TripEntity object.
    3. The persistence manager instance sends a “create” message to instantiate TripEntity Object.
    4. The “Create” message is represented as a dashed line with a stick arrowhead.
    5. Next the persistence manager sends a regular message to the created TripEntity.
    6. This message corresponds to calling the addNote() method on the TripEntity instance.
    7. A regular message is represented with a solid line with filled arrowhead.
    8. We could also add parameters to a message if we wish.
19. Asynchronous message.
20. Self Message.
21. 
22. 
23. 
24. Sequence diagrams help us in clarifying the interactions b/w objects in a specific scenario.  
    By getting more profound insights into the inner workings of our objects we may need to refine their behavior. Or even add new classes or establish new relationships b/w our classes.  
    
25. The process of designing a software system is all about finding out what’s missing, what needs to be enhanced or changed.