SYD366 Week 3

INTRODUCTION TO SEQUENCE DIAGRAMS

Agenda

1. Housekeeping
2. Purpose of a Sequence Diagram
3. Components of a Sequence Diagram
4. Adding Calls and Returns
5. Summary

Housekeeping

- Your first test is in two weeks!
  - Be sure to practice the diagrams, especially in the context of next week's topics
- You are expected to show up to the labs for discussions and exercises
  - I go over extra content and walk through things! It's important!

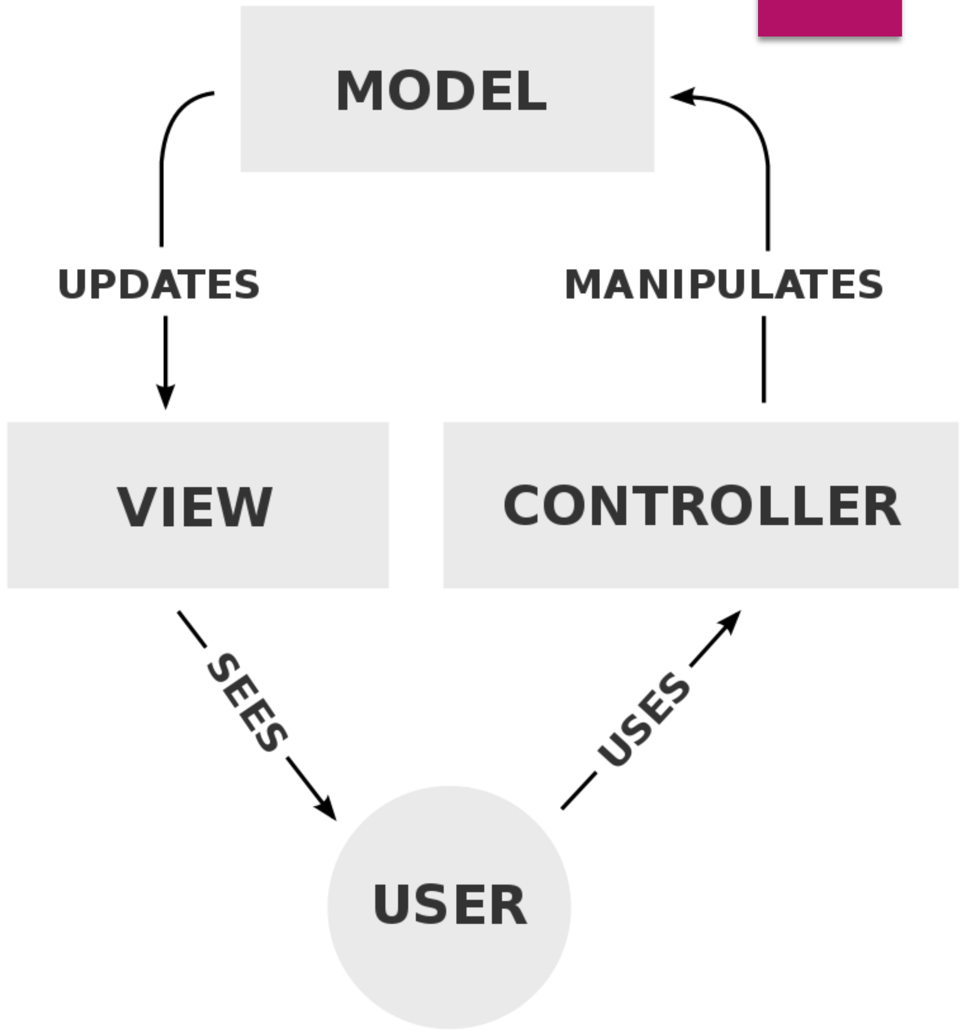
Purpose of a Sequence Diagram

Sequence Diagrams

- What is a sequence diagram?
  - In short, it is a diagram which shows the flow and creation of data
  - It is a detailed list of what functions are to be carried out in a sequence within your software system
- What does it capture?
  - The interaction that takes place in a collaboration that either realizes a use case or an operation
  - High-level interactions between user of the system and the system, between the system and other systems, or between subsystems

MVC Model

- What is the MVC model?
  - Stands for "Model-View-Controller"
  - Divides a program into three elements allowing for information to be represented to the user in ways different from what is stored in the system
  - Traditionally used for systems with GUIs

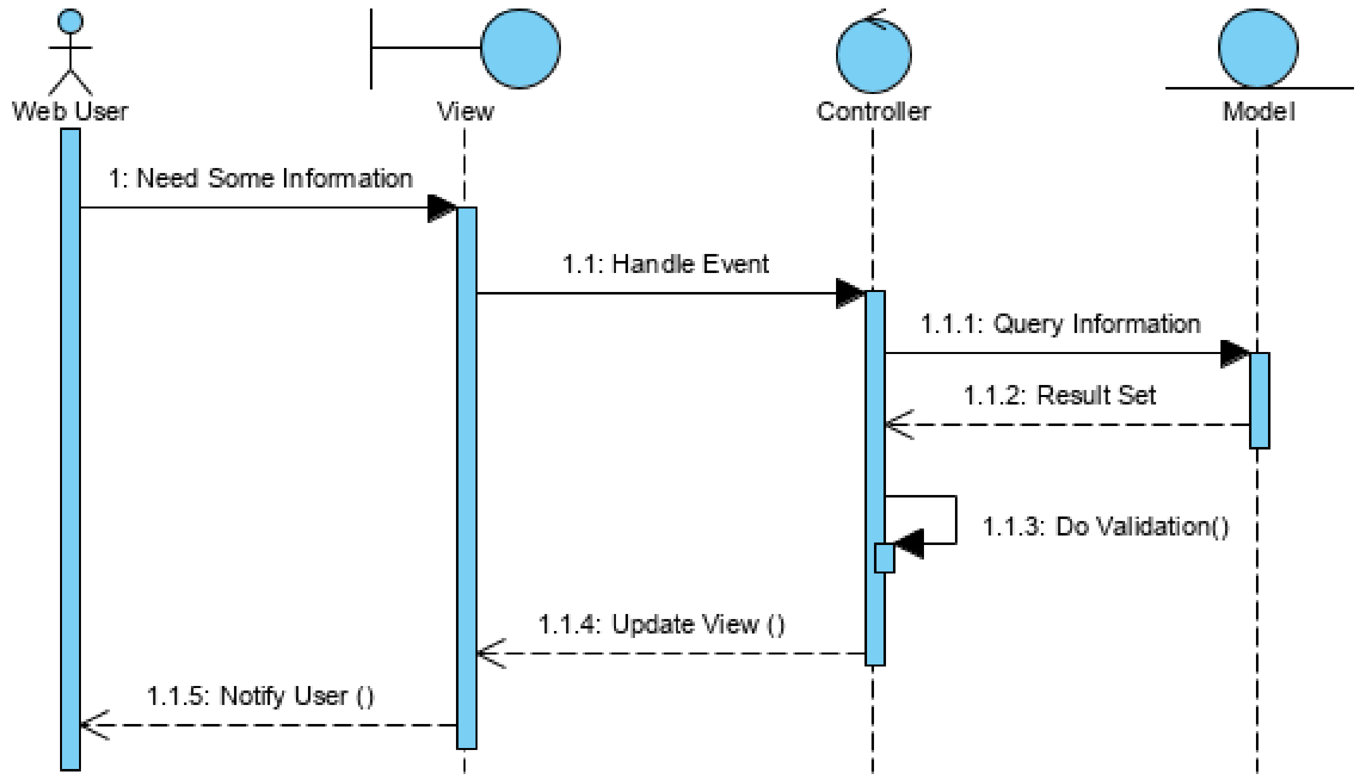


MVC Model (cont.)

- The Model
  - The application's data structure, completely removed from the UI
  - Manages data, logic, rules, etc.
- The Controller
  - Converts data for either the model or the view
  - Can validate data if that's how the program is designed
- The View
  - Fancy word for UI/GUI
  - Represents data in a human readable format

MVC Sequence Diagram

- This is the basic structure of a sequence diagram in the MVC framework
- Notice the initial flow of actions and commands?
  - From user, to view, to controller, finall to model



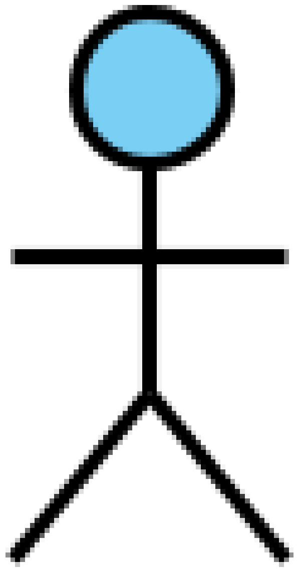
Stepping Back

- Our approach for sequence diagrams will follow the MVC pattern, with slightly more generic variations
- In our diagrams, instead of the model, controller and view, we have the entity manager, domain controller and UI controller
  - Similar, but more flexible in terms of applications!

Components of a Sequence Diagram

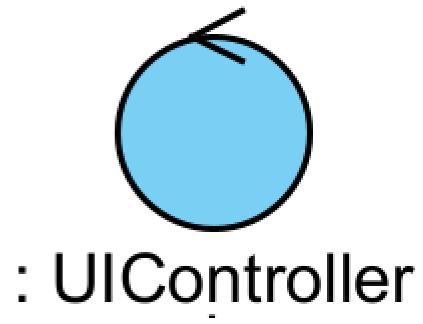
Actor

- The stick figure!
- He is the user for a given sequence diagram
- Will initiate all sequences
- Very important to consider *who* your actor is



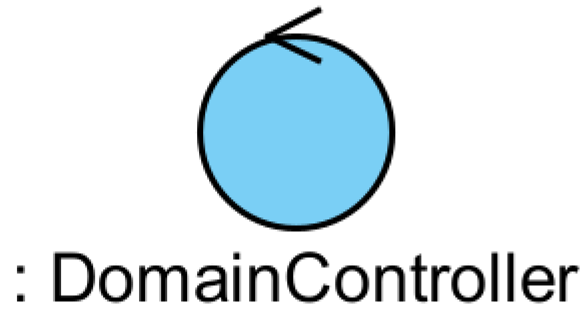
UI Controller

- This is the immediate component of the system which the user interacts with
- Equivalent to the View component



Domain Controller

- The domain controller converts data between the UI controller and the entity manager
- Can (and should) do data validation



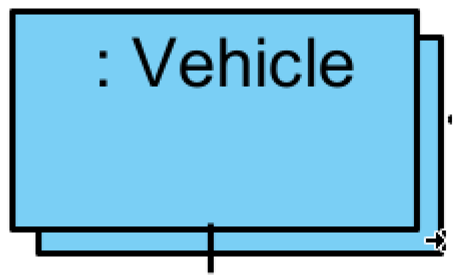
Entity Manager

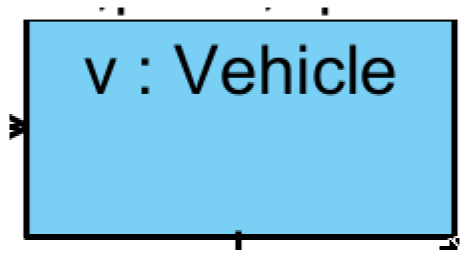
- Where your data is modelled
- Sometimes the database
- Also referred to as the model
- Should be touched sparingly!

: EntityManager

Classes

- When returning data from the entity manager, or creating new data from user input, you will need to build classes from it
- Generally, if you are to *get* data from the entity manager, you will return multiple instances of a class
  - Like how there are multiple vehicles to the right
- If you are creating a new instance of a class, you will generally only create one at a time
  - Again, as what is shown on the right

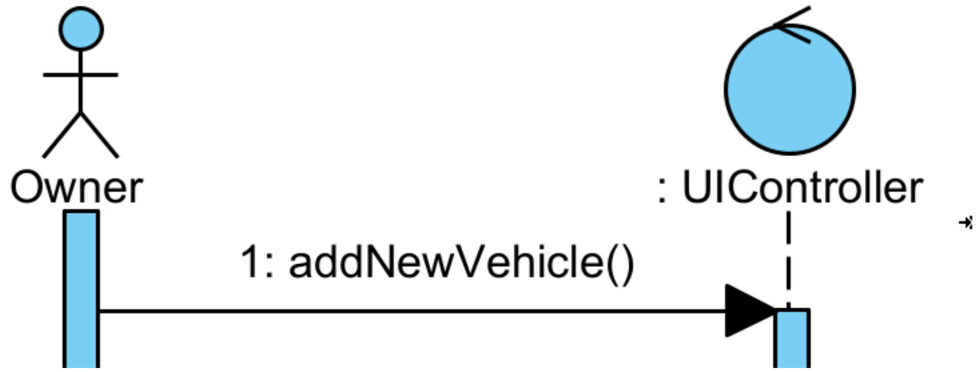




Messages

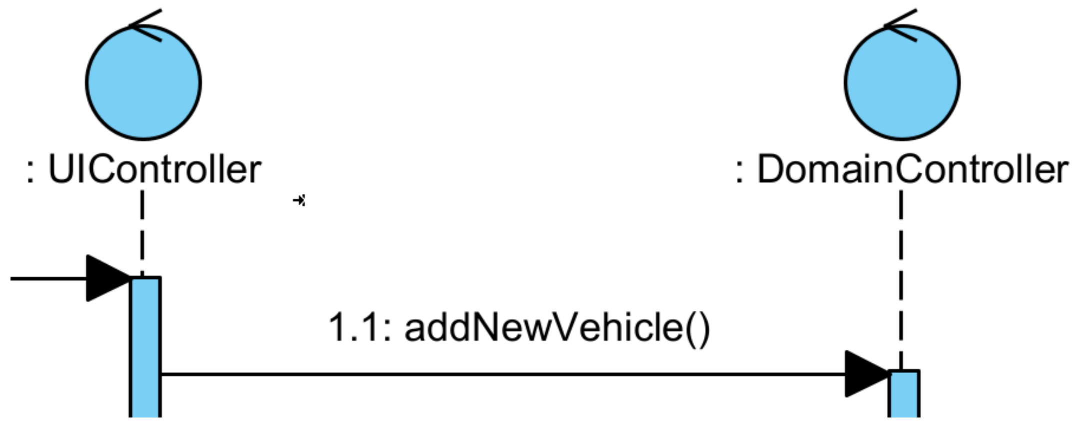
Adding Data

- Your sequence diagram will always start with your user attempting to accomplish something
  - Usually, they want to add new data, query data/edit data or delete data
- Your user will call a function which belongs to the UI controller, like the example on the right
- This will be a solid line
  - Sometimes, these functions will have arguments!



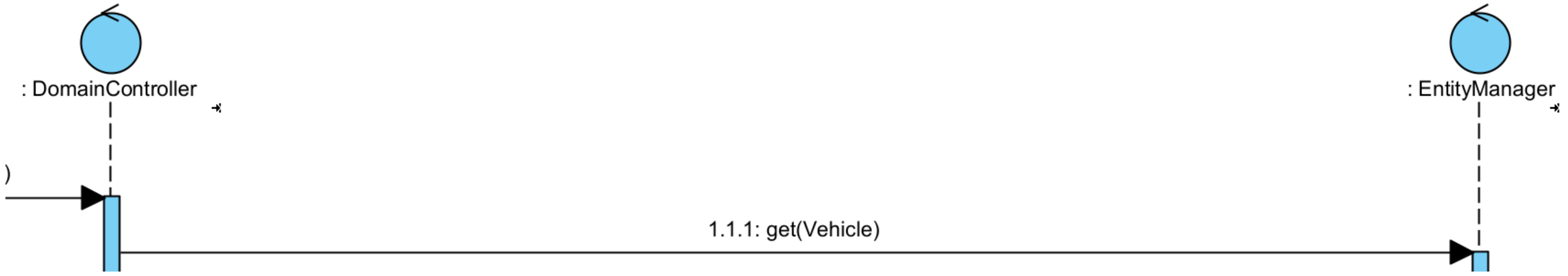
Adding Data

- Your UI controller will then call a function that converts any data entered into a readable format for the model
  - This example does not have a differing function because there is no data to add



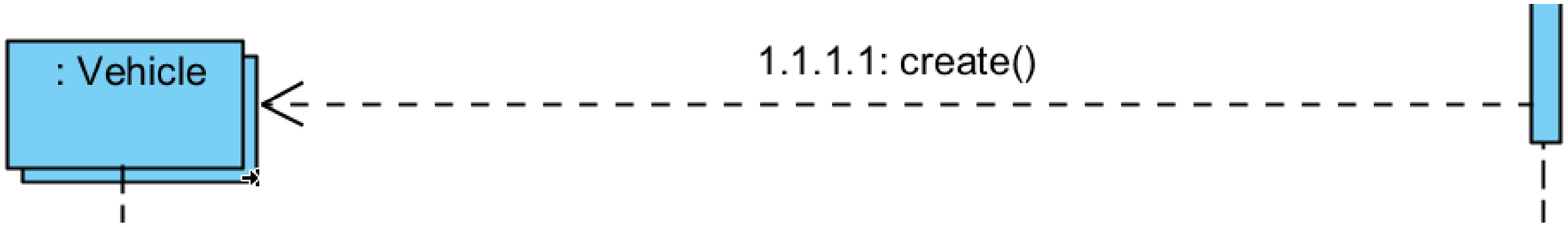
Adding Data

- Finally, once the data has been converted into a model/entity manager readable format, the entity manager/database/model is queried



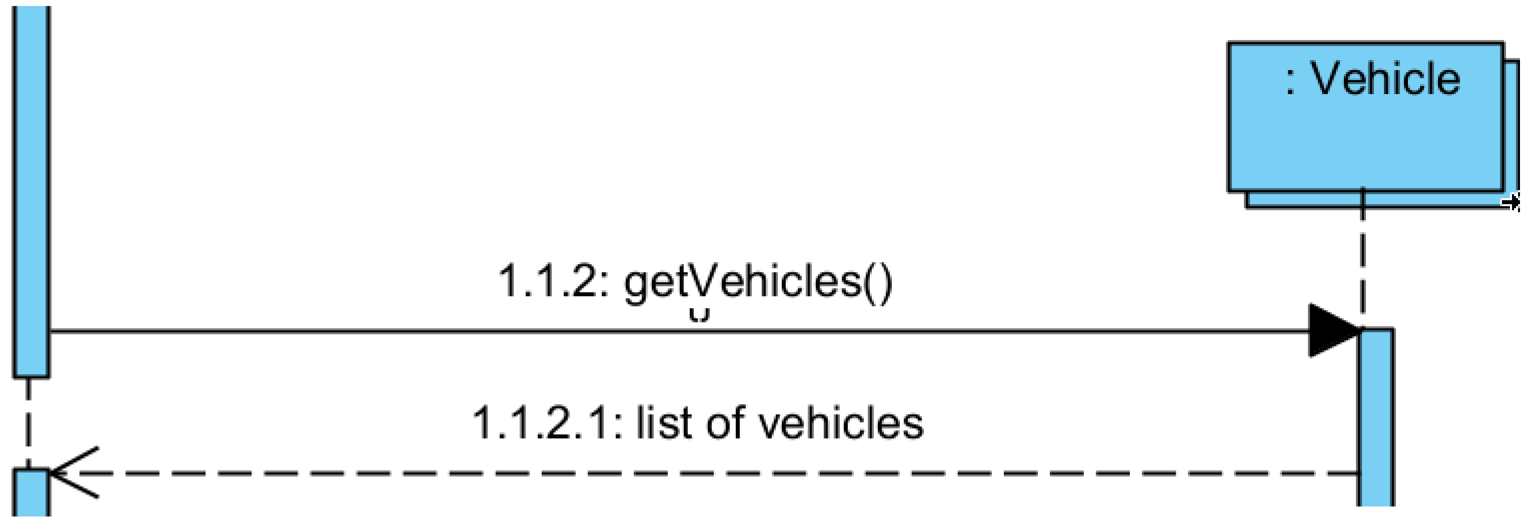
Potential Responses

- The previous slide queries the entity manager for vehicles, so now we must return vehicles
- This response calls a create function which creates these objects in a useable format for the domain controller to convert for the UI controller



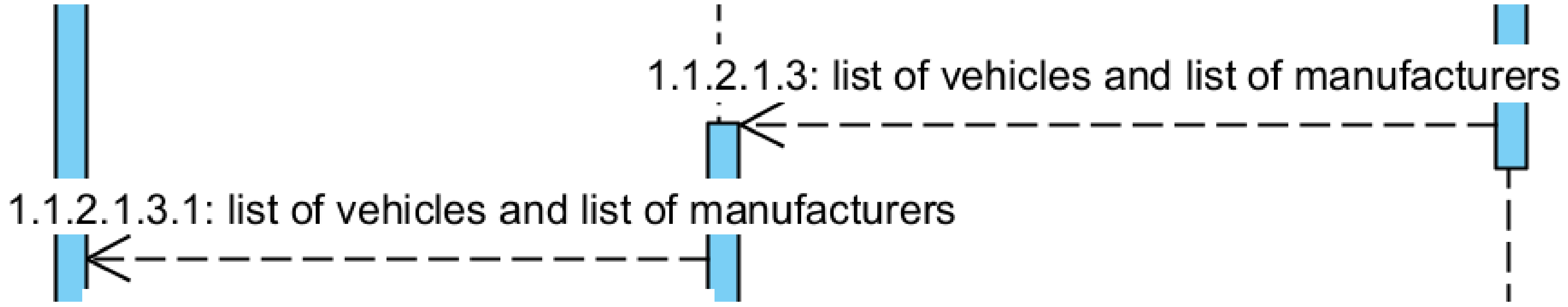
Converting Data

- This dataflow tidbit shows how the domain controller interacts with the data the entity manager has produced in order to make it meaningful for the UI controller



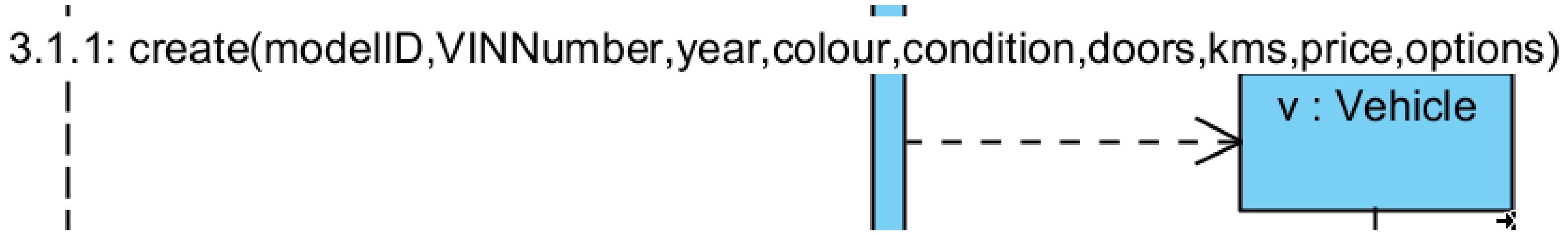
Potential Responses

- If you are returning values, plain English for return messages is acceptable
- These return responses have dotted lines


1.1.2.1.3: list of vehicles and list of manufacturers
1.1.2.1.3.1: list of vehicles and list of manufacturers

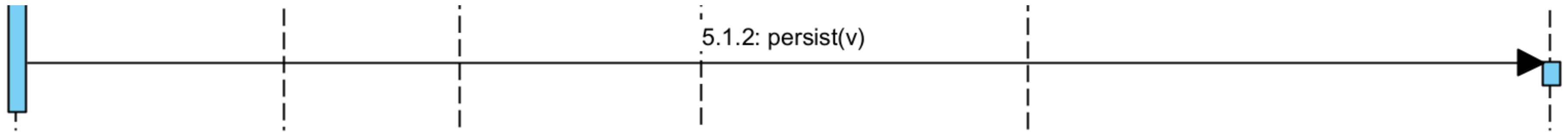
Adding/Modifying Data

- To create, search or modify data, you will need to pass arguments into these functions flowing from the UI controller to the domain controller to the entity manager
- Remember: this flow starts from the actor's interaction with the UI controller
- Notice that when the create function is created, a single vehicle object is made? That must be done by the domain controller


3.1.1: create(modelID,VINNumber,year,colour,condition,doors,kms,price,options)
v : Vehicle

Persisting

- The final step when adding, modifying or deleting data in a sequence diagram is that of persisting
- It is simply ensuring any changes are committed to the entity manager
- This is done from the domain controller



Summary

- Our sequence diagrams follow a generalized version of the MVC control pattern
- The entity manager is where your data is modelled/stored/handled
- The domain controller is the intermediary between your UI and the entity manager
- The UI controller is what your actor sees when interacting with the system
- When working with data, it is important to remember how to get information from the entity manager in a human readable format when returning it to the actor
- Always persist changes right at the end!

Questions?