


CSE 416



*You should be working on
your server design during
mid-term exam week*

UML Sequence Diagrams



*For the Object-Oriented
(Java) part of your system*

© Robert Kelly, 2012-2020



2 Reading / Reference

▀ Reading

www.lucidchart.com/pages/uml-sequence-diagram

www.ibm.com/developerworks/rational/library/3101.html

https://en.wikipedia.org/wiki/Activity_diagram

© Robert Kelly, 2012-2020

3

Interaction Diagrams

- ▶ Sequence diagrams and collaboration diagrams
- ▶ Describe the *dynamic behavior* of an OO system
- ▶ Often used to model a use case
- ▶ Purpose:
 - ▶ Model interactions between objects
 - ▶ Verify/revise the class diagram
 - ▶ Assign responsibilities/operations to classes

*We focus on
sequence diagrams*

© Robert Kelly, 2012-2020

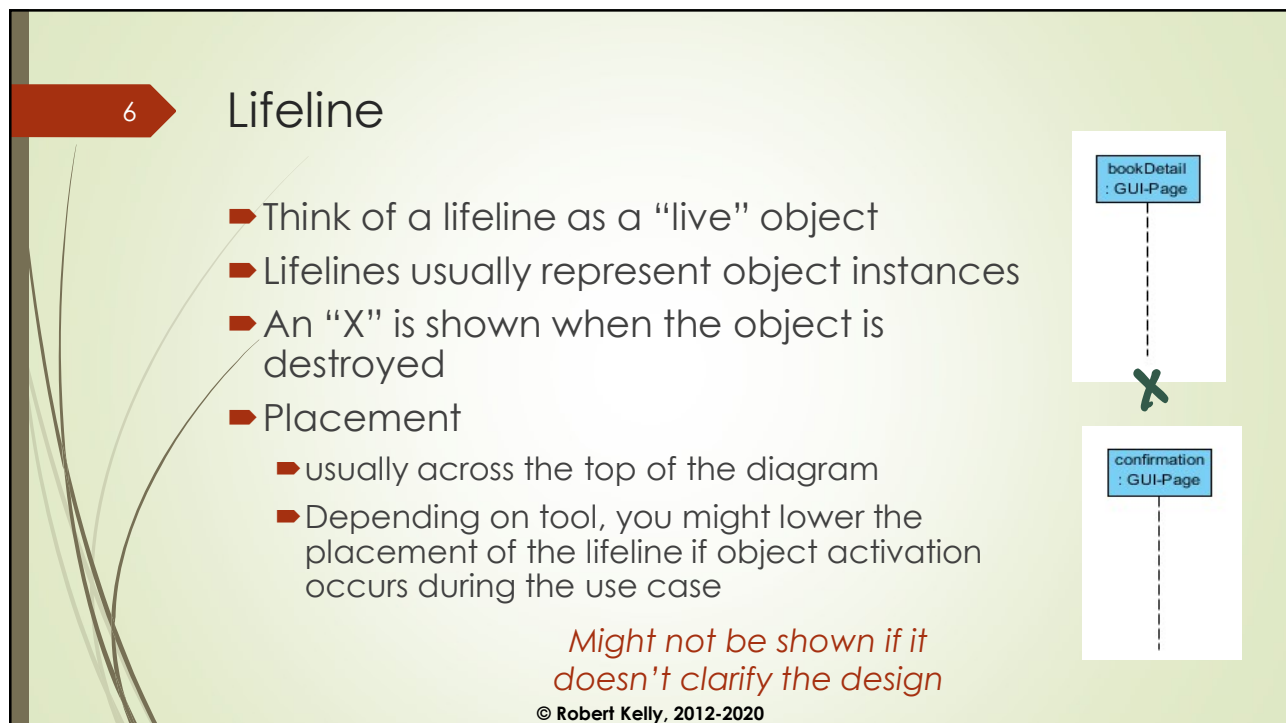
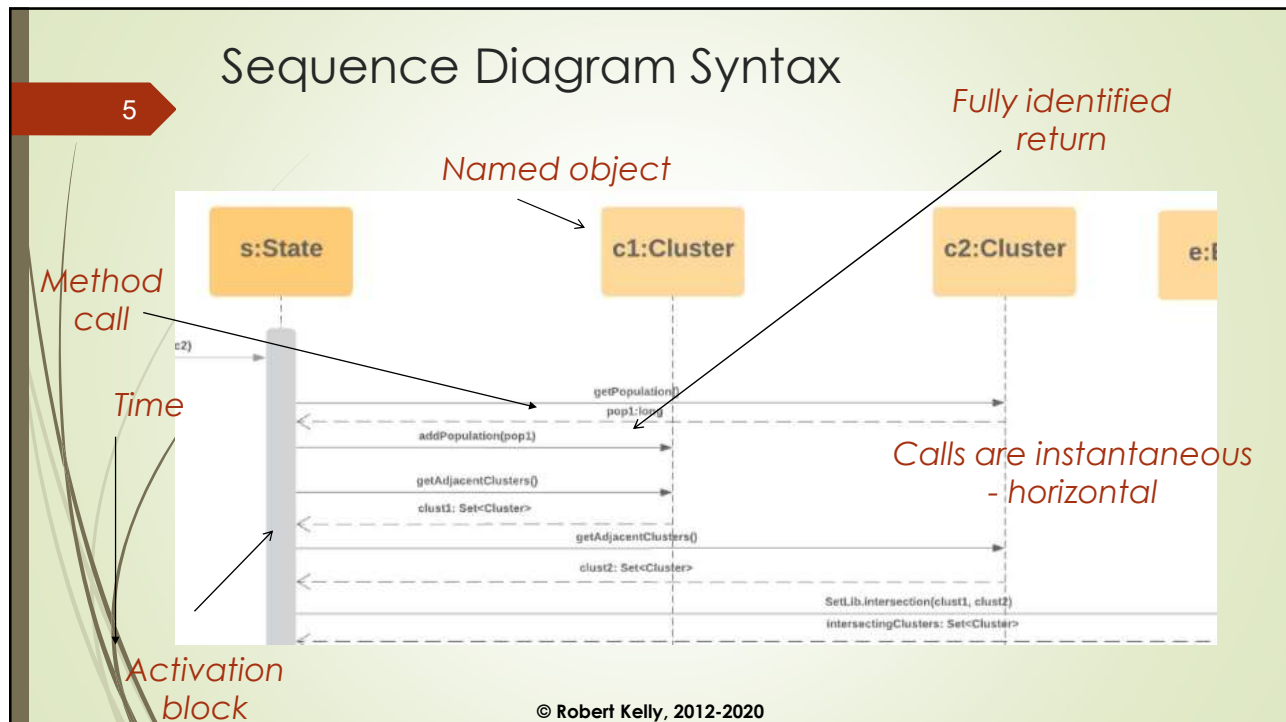
4

UML Sequence Diagram

- ▶ **Sequence diagram** - an interaction diagram that models a single scenario (use case) executing in the system
 - ▶ perhaps 2nd most used UML diagram (behind class diagram)
- ▶ Illustrates how objects interact with each other
- ▶ Emphasizes time ordering of messages
- ▶ Can model simple sequential flow, branching, iteration, recursion and concurrency

*Helps you design proper
encapsulation of your data*

© Robert Kelly, 2012-2020

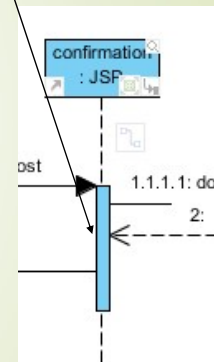


7

Indicating Method Calls

- Activation box: thick box over object's life line; drawn when object is on the stack
 - Either that object is running its code, or it is on the stack waiting for another object's method to finish
 - Nest to indicate recursion

Activation
box



© Robert Kelly, 2012-2020

8

Key Components

- **Participant:** an object or entity that acts in the sequence diagram
 - sequence diagram starts with an unattached arrow or an arrow attached to an actor
- **Message:** communication between objects/actors
- Axes in a sequence diagram:
 - horizontal: which object/participant is acting
 - vertical: time (down -> forward in time)


For complex logic, the diagram can start with another use case

In a GUI system the initial participant is usually an actor

© Robert Kelly, 2012-2020

9

Messages

- An interaction between two objects is performed as a message sent from one object to another (e.g., method call)
- If an object sends a message to another object, object 1 must have visibility to object 2 (i.e., have a handle)
- A message is represented by an arrow between the life lines of two objects 
 - Self calls are also allowed
 - The time required by the receiver object to process the message is denoted by an *activation-box*.
- A message is labeled at a minimum with the method name, and if space permits, the parameters

© Robert Kelly, 2012-2020

10

Arrow Labels

- Method call
 - Label the call arrow with the method name
 - Include parameters if they are not obvious
- Return
 - Model a return value when you need to refer to it elsewhere, e.g. as a parameter passed in another message

In general, don't model obvious interactions if the modeling tool is not able to automatically generate code

© Robert Kelly, 2012-2020

11

Simplification

- In your design review, you can simplify some cases, once you have shown an understanding
- Examples
 - Do not show any client logic other than the user trigger on DOM object and client communications object (XMLHttpRequest/Window)
 - Do not show client server detail – show a message from client (e.g., Window object) to server Controller object
 - Do not show any DB access – stop at message to EntityManager object

*Show HTTP
method*

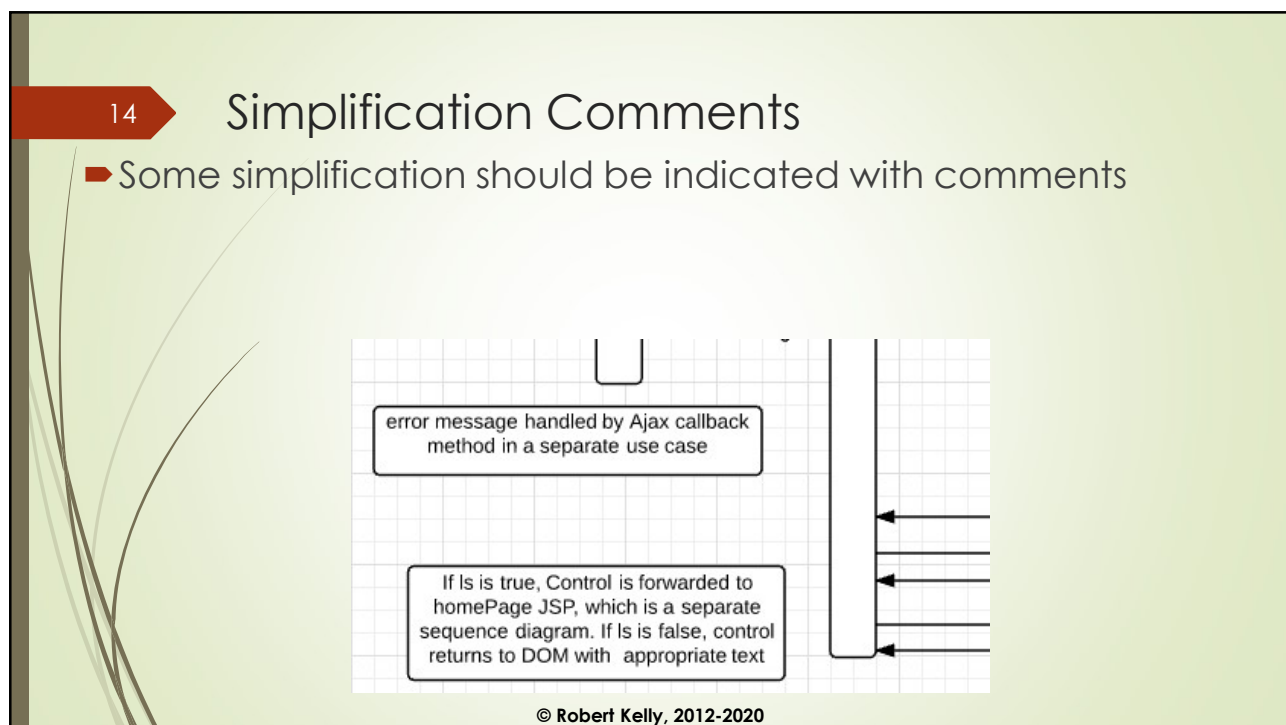
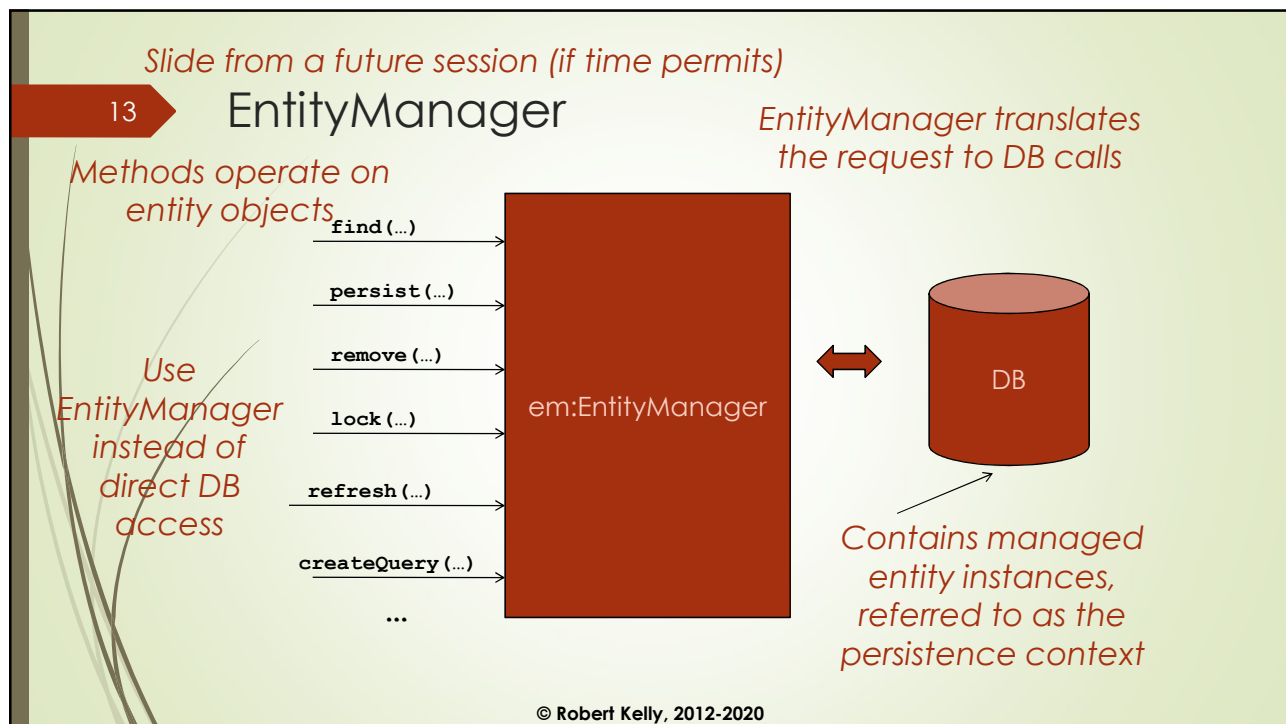
© Robert Kelly, 2012-2020

12

Simplification – Fundamental Objects

- DOM – object representation of the GUI
- Client request
 - XMLHttpRequest– Standard browser object - interacts with the server
 - Window object uses a fetch method to interact with server
- Persistence layer – Standard server object to receive object requests for the DB
 - Best represented as the em:EntityManager object
 - Receives calls as in JPA

© Robert Kelly, 2012-2020



15

Realistic Design Approach

- Use your sequence diagrams to identify classes and class attributes needed in your class diagram
- Work both simultaneously (e.g. add methods to your class diagram once you see that you need it)
- Don't be reluctant to modify your design during this stage

© Robert Kelly, 2012-2020

16

Project Team Approach

- The first few sequence diagrams will be very difficult to do
- Do the first few as a team (with lots of team interaction)
- Once your team begins to understand your design philosophy and framework philosophy, you will be able to assign parts to team members
- Look for common design approaches (e.g., DB access, server access, session management), you might be able to use sub-diagrams

© Robert Kelly, 2012-2020

17

Design Review

- Design review will be organized along the lines of use cases (and corresponding sequence diagrams and activity diagrams)
- Your team gets to pick the first use case to show
- Clarity of thinking and consistency are more important than getting the best possible design approach
- OO design will be covered more heavily than procedural design
- Think encapsulation in your OO design

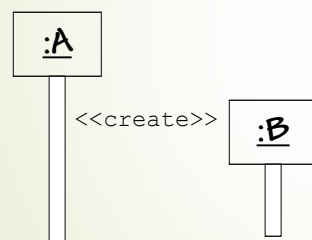
© Robert Kelly, 2012-2020

18

Object Instantiation

- An object may create another object via a `<<create>>` message.

Preferred



Using new is OK, but you will be probably use the factory design pattern

© Robert Kelly, 2012-2020

19

Indicating Selection and Loops

- frame: box around part of a sequence diagram to indicate selection or loop
 - if -> (opt) [condition]
 - if/else -> (alt) [condition], separated by horiz. dashed line
 - loop -> (loop) [condition or items to loop over]

Loops are not very helpful in sequence diagrams. If you need to show a loop, you might just indicate it with a comment

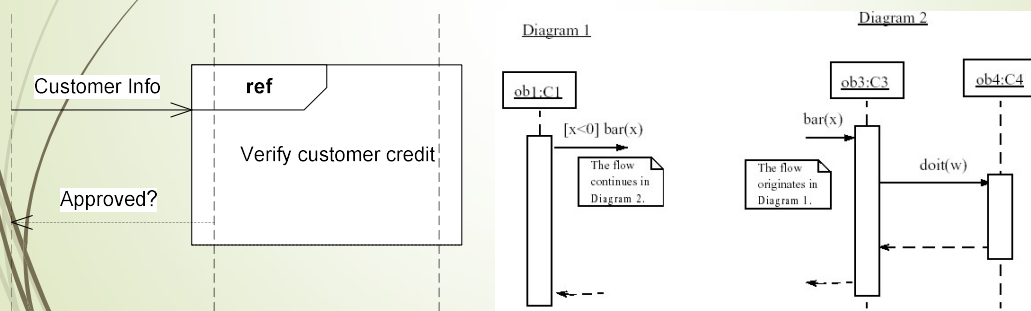
© Robert Kelly, 2012-2020

20

Linking Sequence Diagrams

Although this might result from the use case being too large

- If one sequence diagram is too large or refers to another diagram, indicate it with either:
 - An unfinished arrow and comment
 - A "ref" frame that names the other diagram



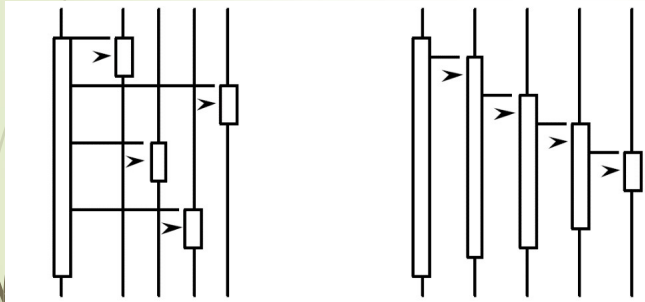
© Robert Kelly, 2012-2020

21

(De)centralized System Control

- What can you say about the control flow of each of the following systems?

centralized?



This will help you think about whether your design properly uses encapsulation

© Robert Kelly, 2012-2020

22

Why Not Just Code It?

- A sequence diagram allows you to think through design issues
- A sequence diagram is well above the level of code
- Tool might generate code
- Sequence diagrams are somewhat language-agnostic (can be implemented in many different OO languages)
- Easier to do as a team
- Can see many objects/classes at the same time

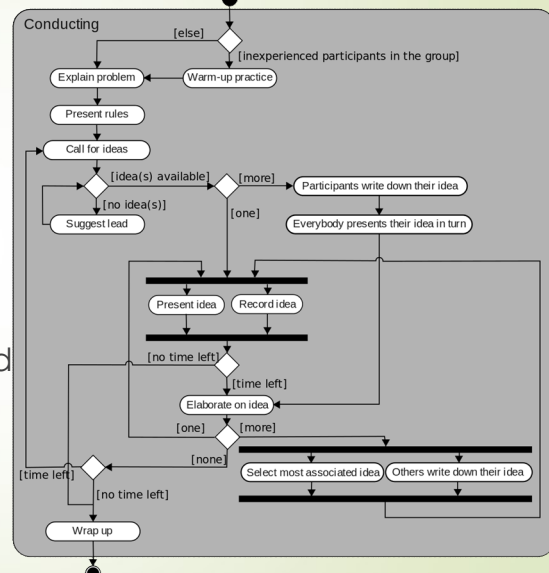
© Robert Kelly, 2012-2020

23

Activity Diagrams

- For the Python parts of the project, assume a procedural programming model
- To show your procedural design, use a UML activity diagram
- Show major activities (e.g., generate spanning tree) and data flows (e.g., node graph)
- Use ellipses (actions), diamonds (decisions), arrows, circles (start and end)

Lightly covered in design reviews



© Robert Kelly, 2012-2020

Image: Wikipedia