Sommerville Chapter 7
Fowler Chapters 4 and 11

Dynamic Models

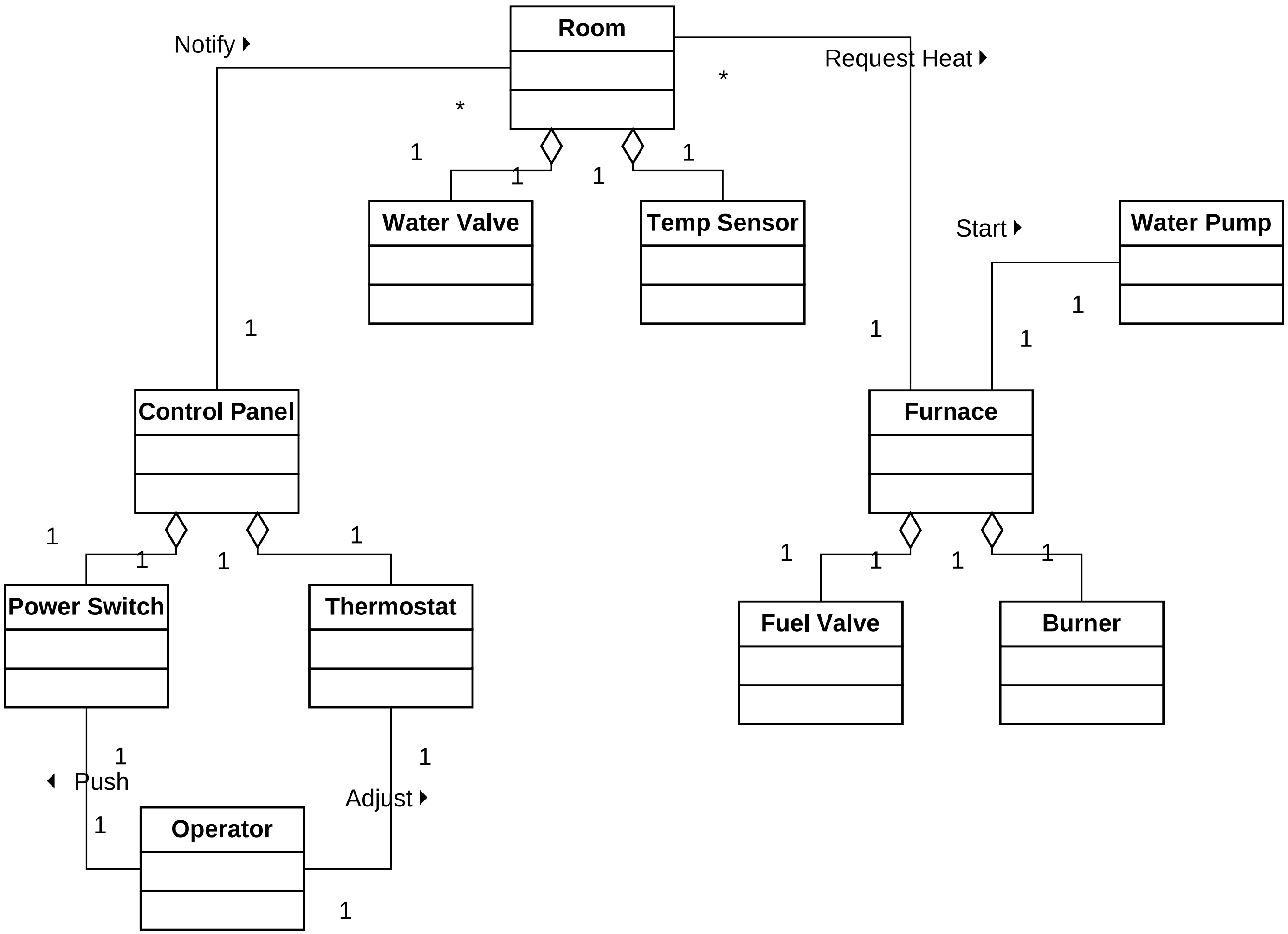
Announcements

- As an alternative to the Papyrus Tool on Eclipse, try the free online drawing tool

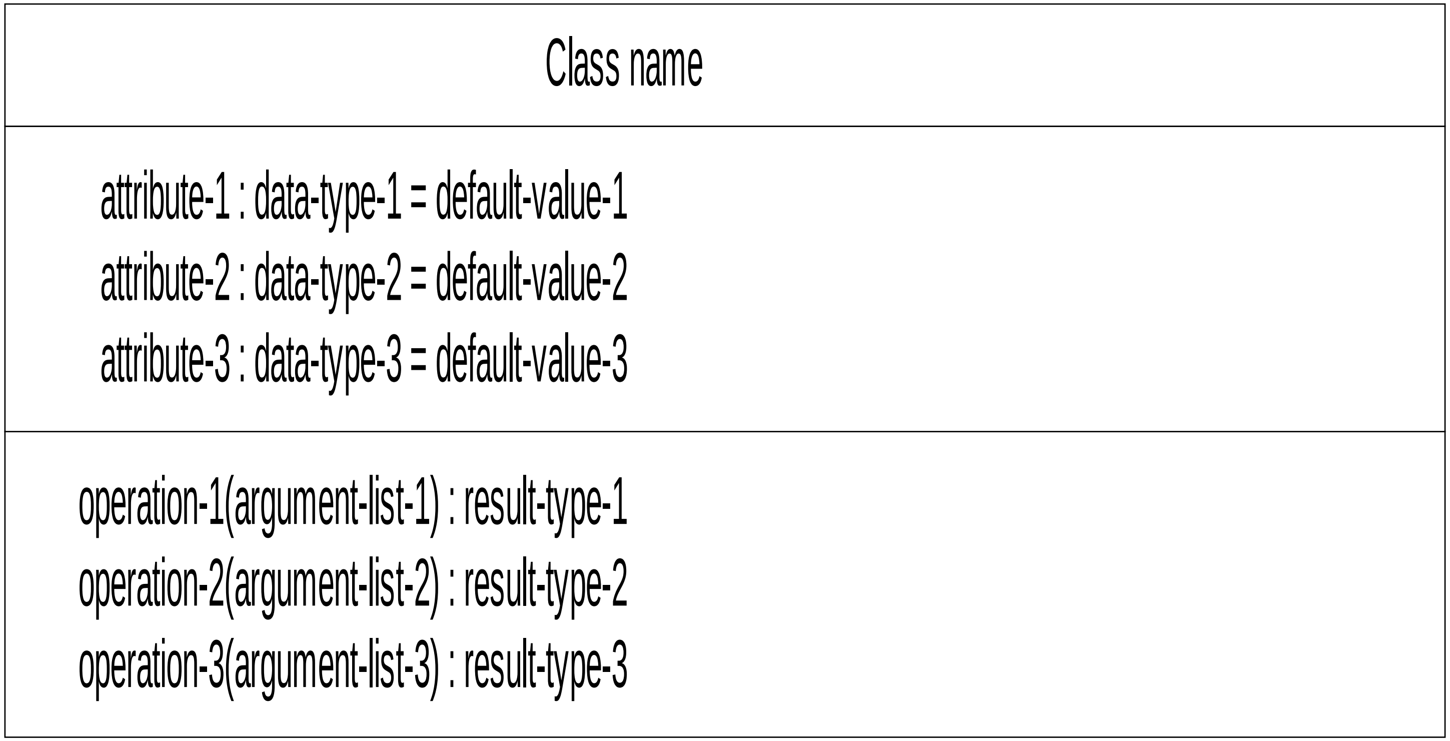
  https://www.draw.io/

  - Use the UML class and sequence diagram objects from the menu on the left side panel (under “UML”).

Class Diagram



Object Notation - Summary



Class name

attribute-1 : data-type-1 = default-value-1

attribute-2 : data-type-2 = default-value-2

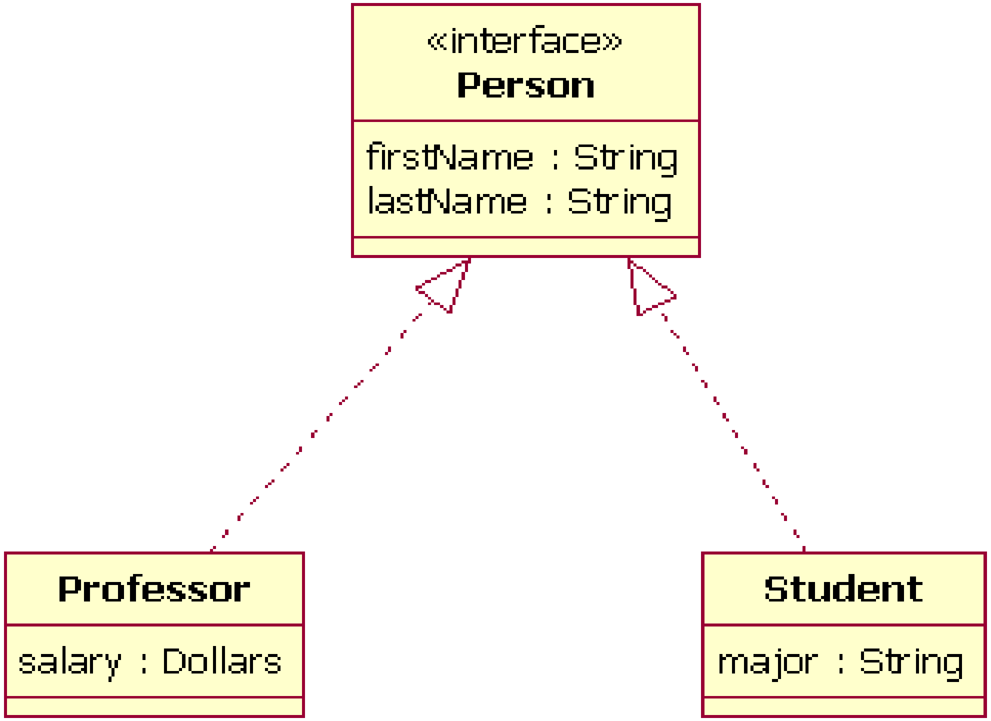
attribute-3 : data-type-3 = default-value-3

operation-1(argument-list-1) : result-type-1

operation-2(argument-list-2) : result-type-2

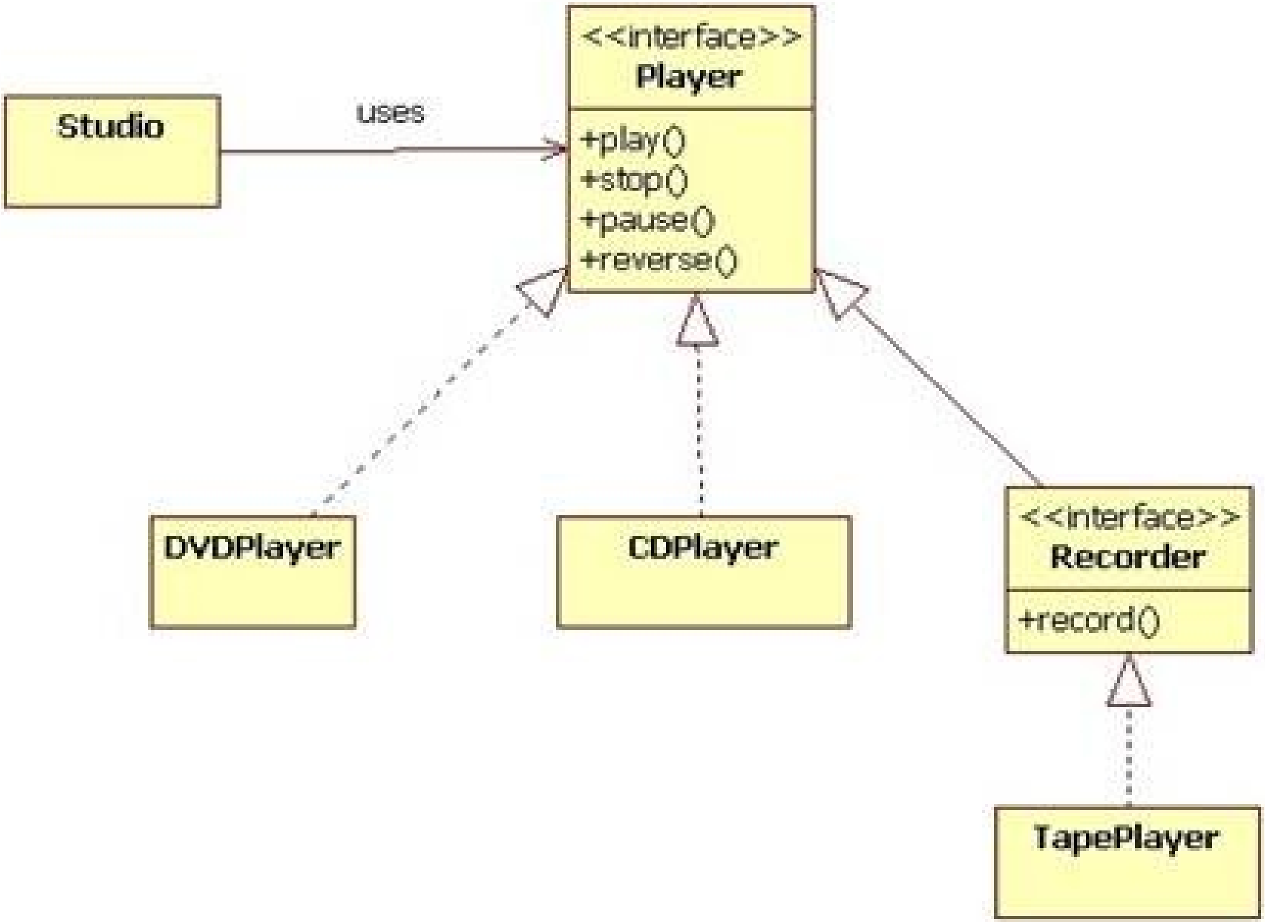
operation-3(argument-list-3) : result-type-3

Interface Classes



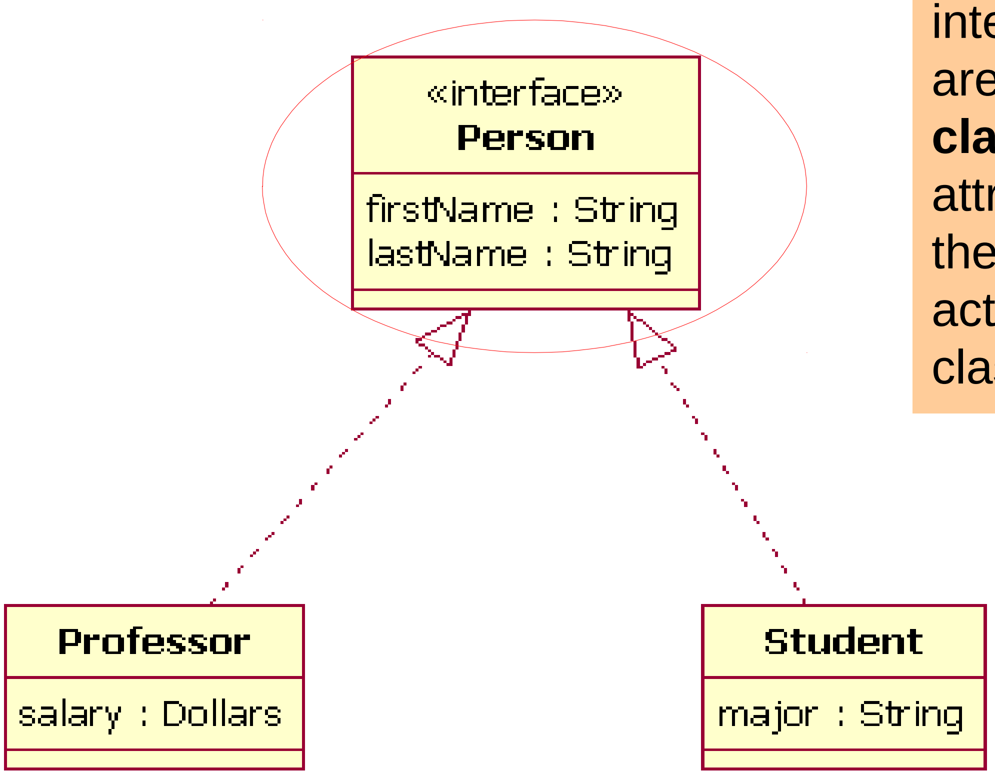
A class and an interface **differ**:
A class can have an actual instance of its type, whereas an interface must have at least one class to implement it. In UML 2, an interface is considered to be a specialization of a class modelling element. Therefore, an interface is drawn just like a class, but the top compartment of the rectangle also has the text "«interface»", as shown in Figure.

Interface Classes



http://www.cs.sjsu.edu/~pearce/modules/lectures/oop/basics/interfaces.htm

Interface Classes



For sensors and actuators that the Cruise Control system interacts with, draw interface classes. For your homework, you are only required to show the **interface class** for each sensor/actuator with attribute and operations that are useful to the CCS. Details on the sensor and actuator class implementing the interface class is not necessary.

Class Diagram for CCS

- The class diagram should have a class for the CCS
- The class diagram should have interface classes for the different sensors, timer, and actuator.
- You should also include class(es) for the buttons on the dashboard.

Overview

- The object model describes the structure of the system (objects, attributes, and operations)
- The dynamic model describes how the objects change state (how the attributes change) and in which order the state changes can take place
- Several models used to find the appropriate dynamic behavior
  - Interaction diagrams
  - Activity diagrams
  - State Diagrams
    - Uses finite state machines and expresses the changes in terms of events and states

Fowler Chapters 4 and 11

Interaction Diagrams

Objectives for Today

- Learn why we use interaction diagrams
- Discuss sequence diagrams
  - Capturing use-cases
  - Dealing with concurrency
- Describe collaboration diagrams
- Describe Activity diagrams
- Clarify when to use what
- Discuss when to use interaction diagrams

Different Types of Interaction Diagrams

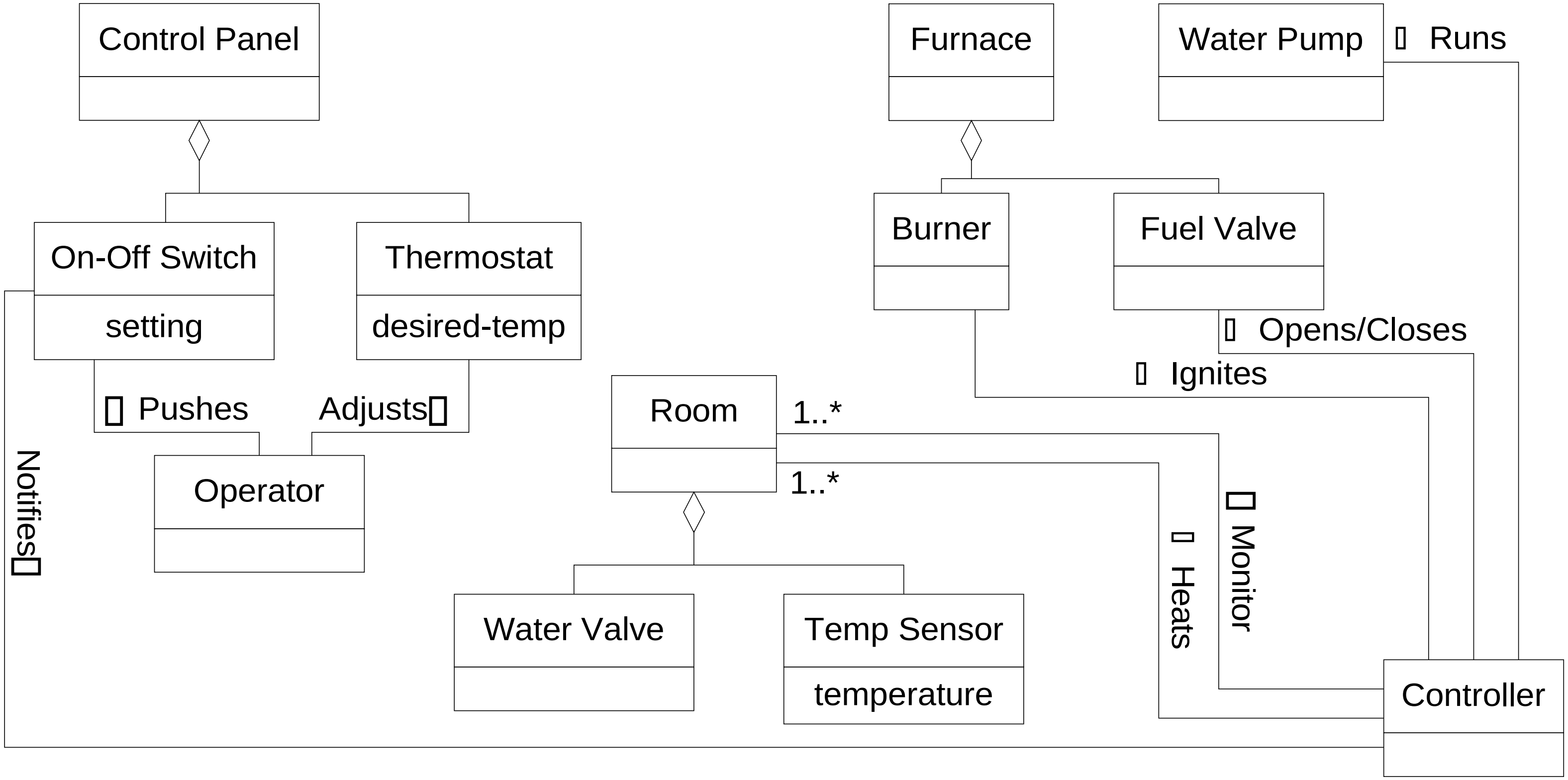
- An Interaction Diagram typically captures a use-case
  - A sequence of user interactions

- Sequence diagrams
  - Highlight the sequencing of the interactions between objects
- Collaboration diagrams
  - Highlight the structure of the components (objects) involved in the interaction

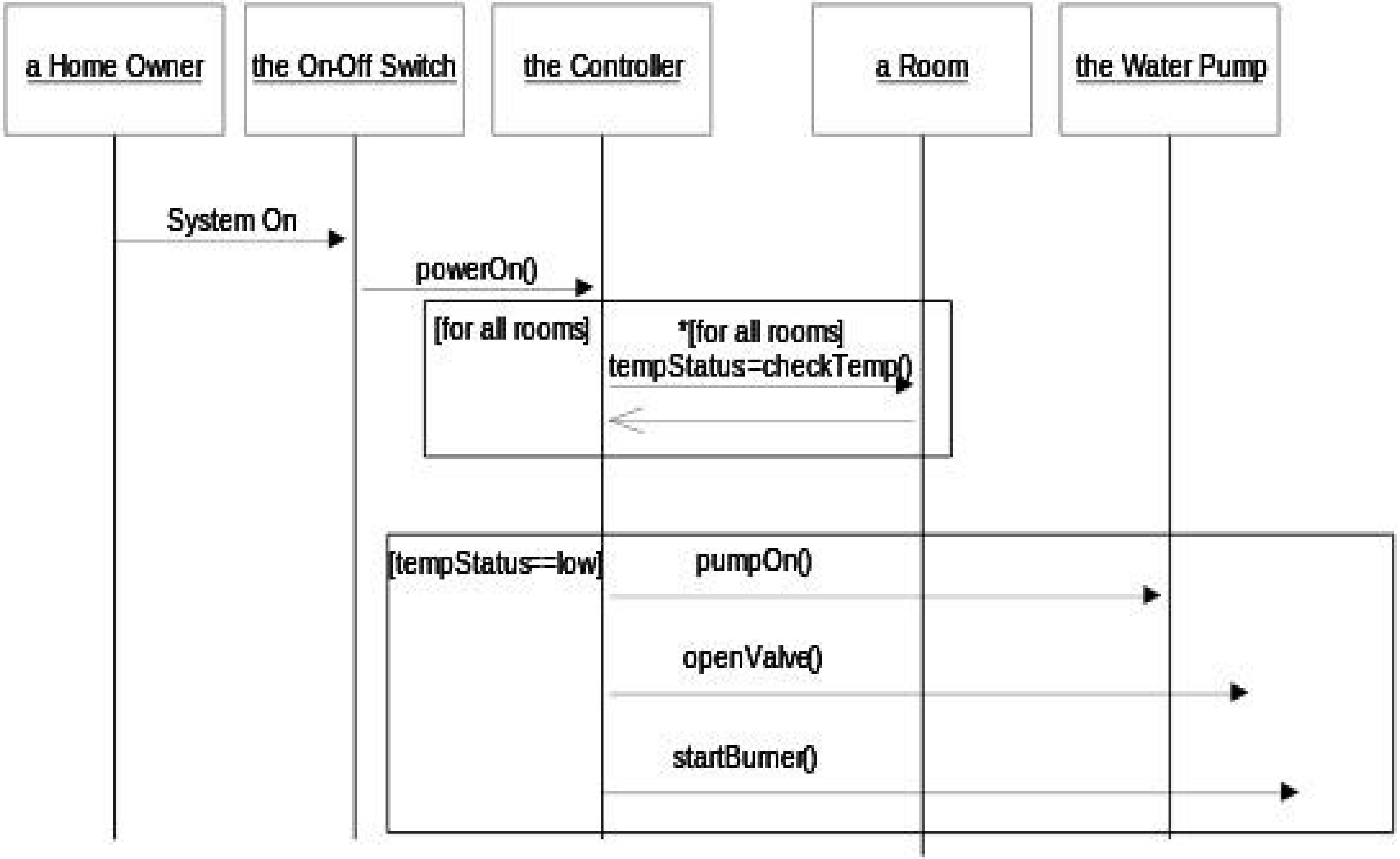
Home Heating Use-Case

Use case: **Power Up**
Actors: Home Owner (initiator)
Type: Primary and essential
Description: The Home Owner turns the power on. Each room is temperature checked. If a room is below the the desired temperature the valve for the room is opened, the water pump started, the fuel valve opened, and the burner ignited.
If the temperature in all rooms is above the desired temperature, no actions are taken.
Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: None

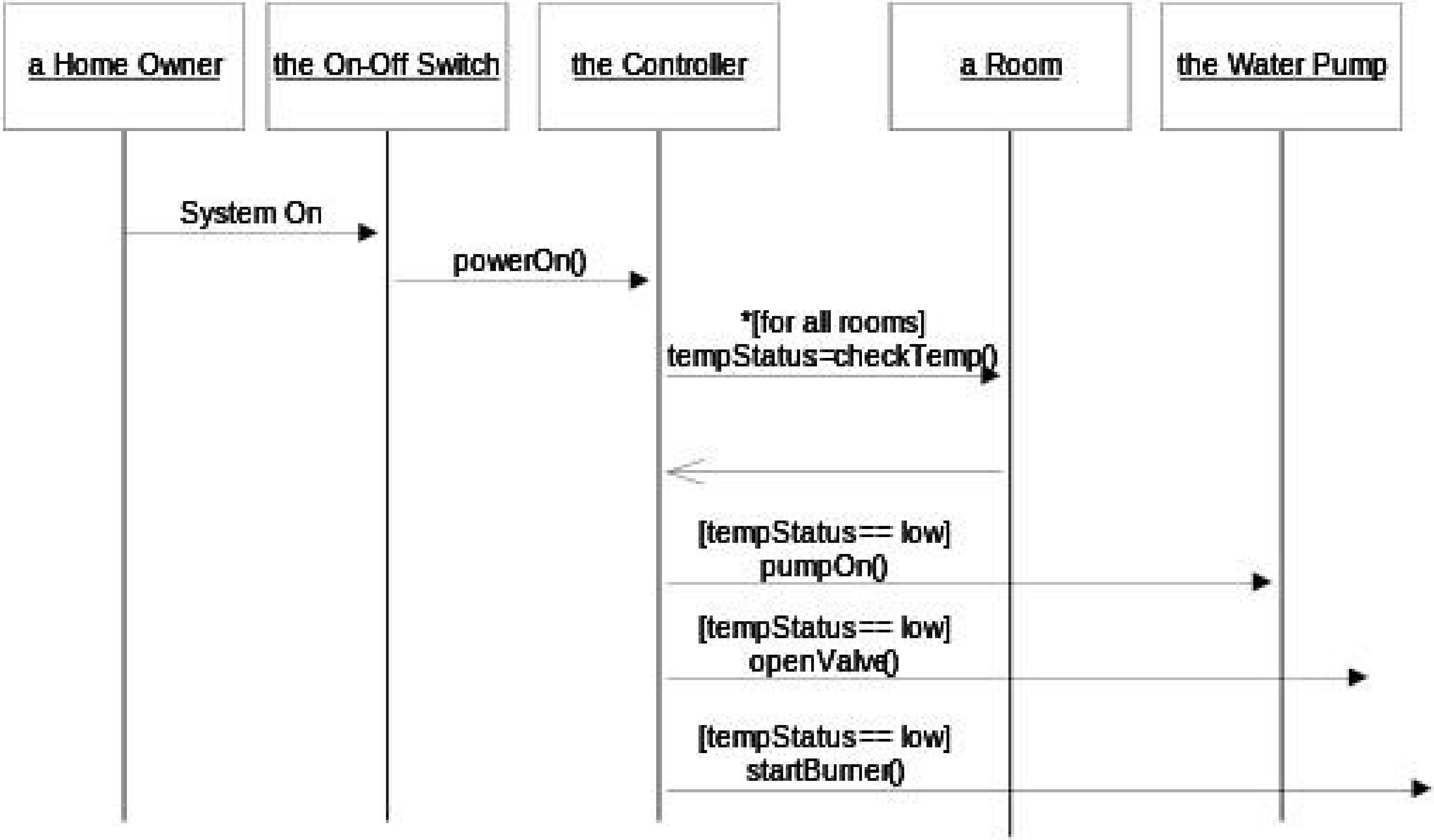
Class Diagram—v1



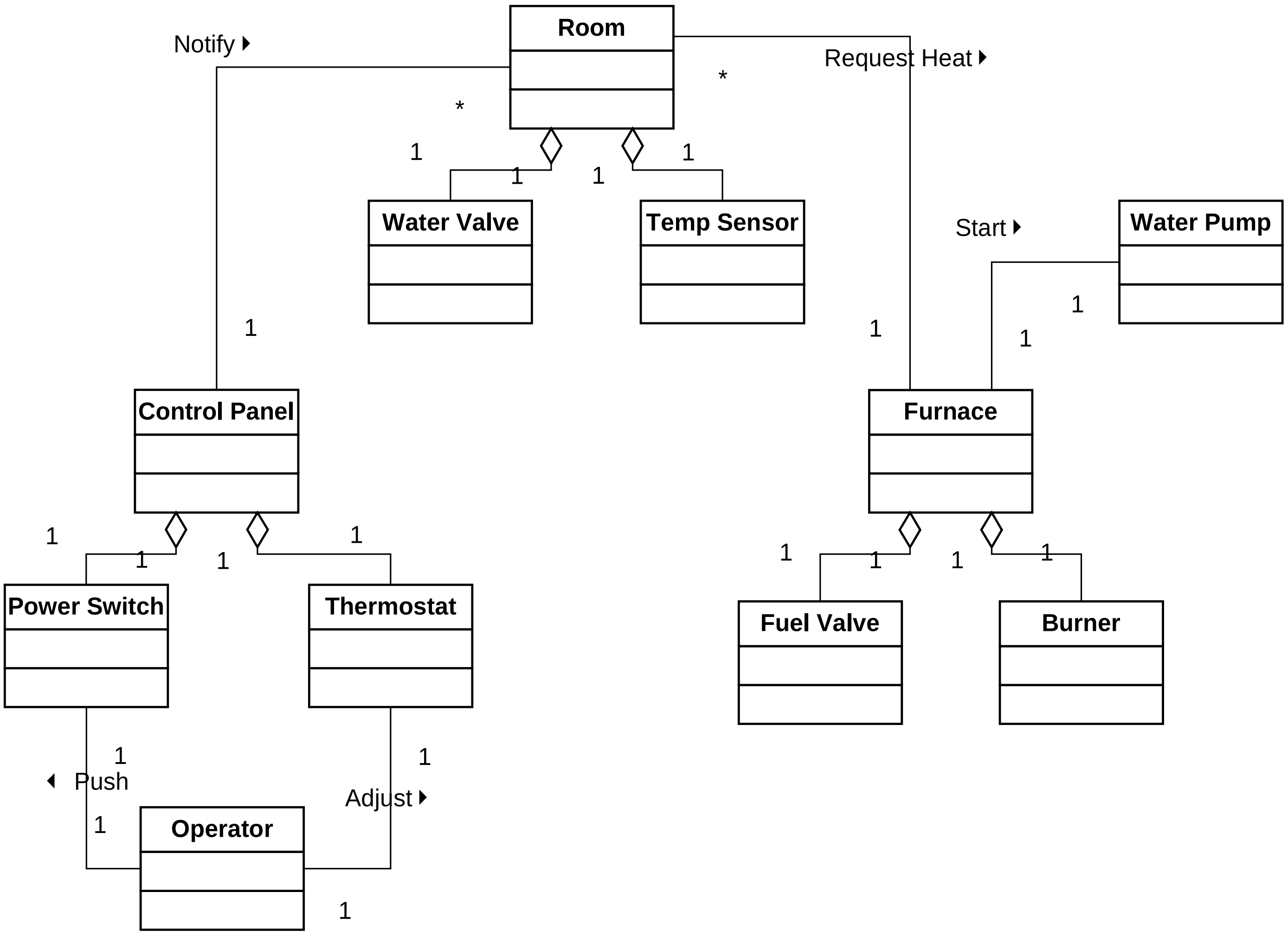
Sequence Diagrams (for cd—v1)



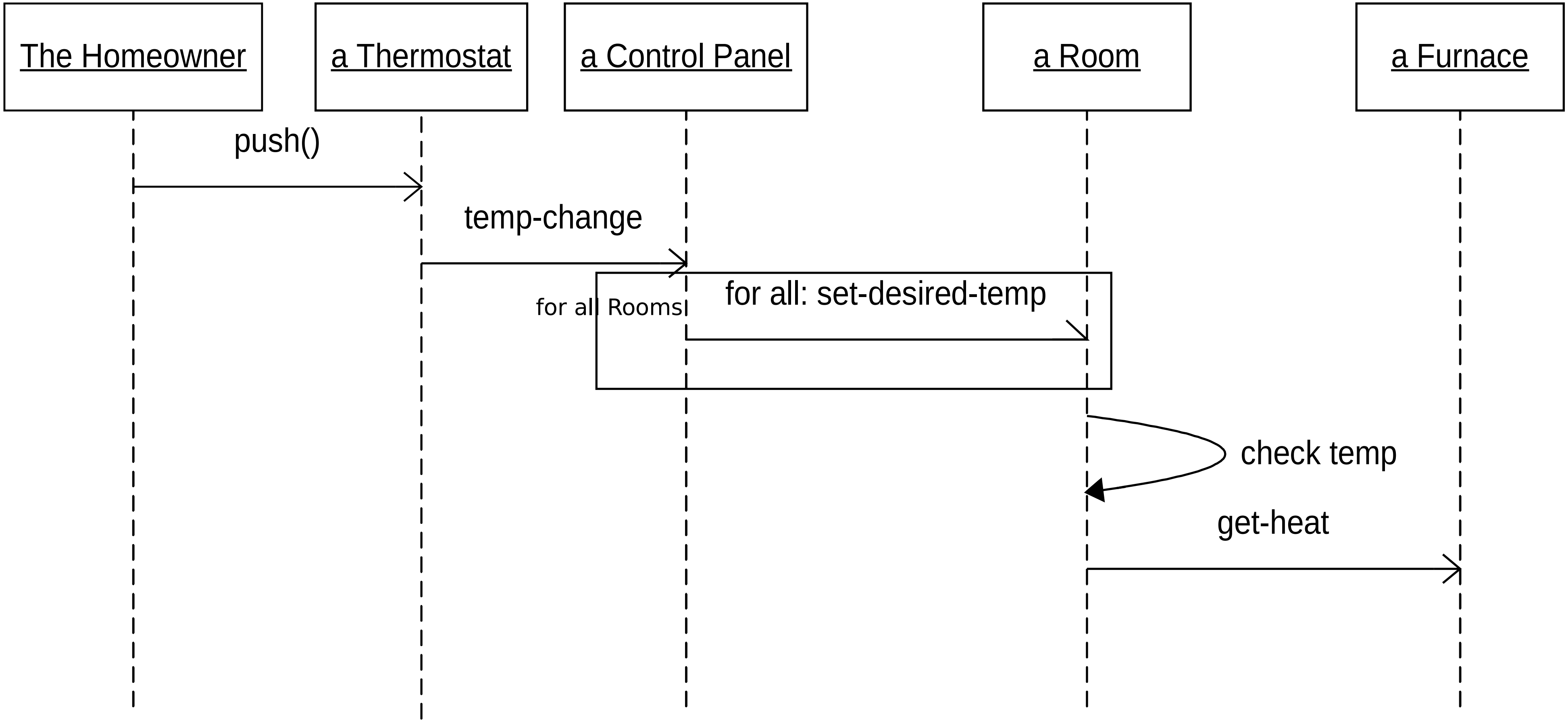
Sequence Diagrams (old) (for cd—v1)



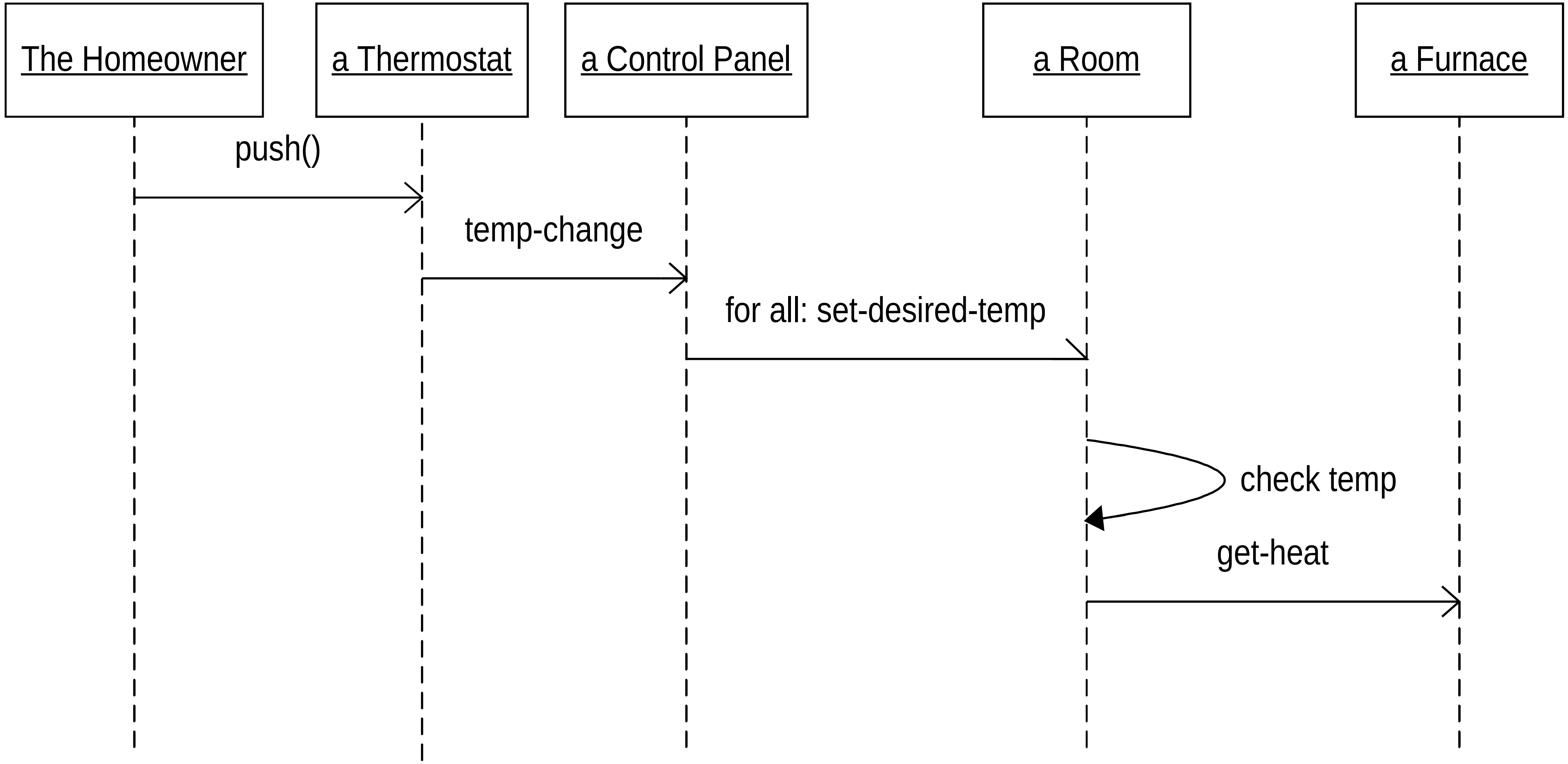
Class Diagram—v2



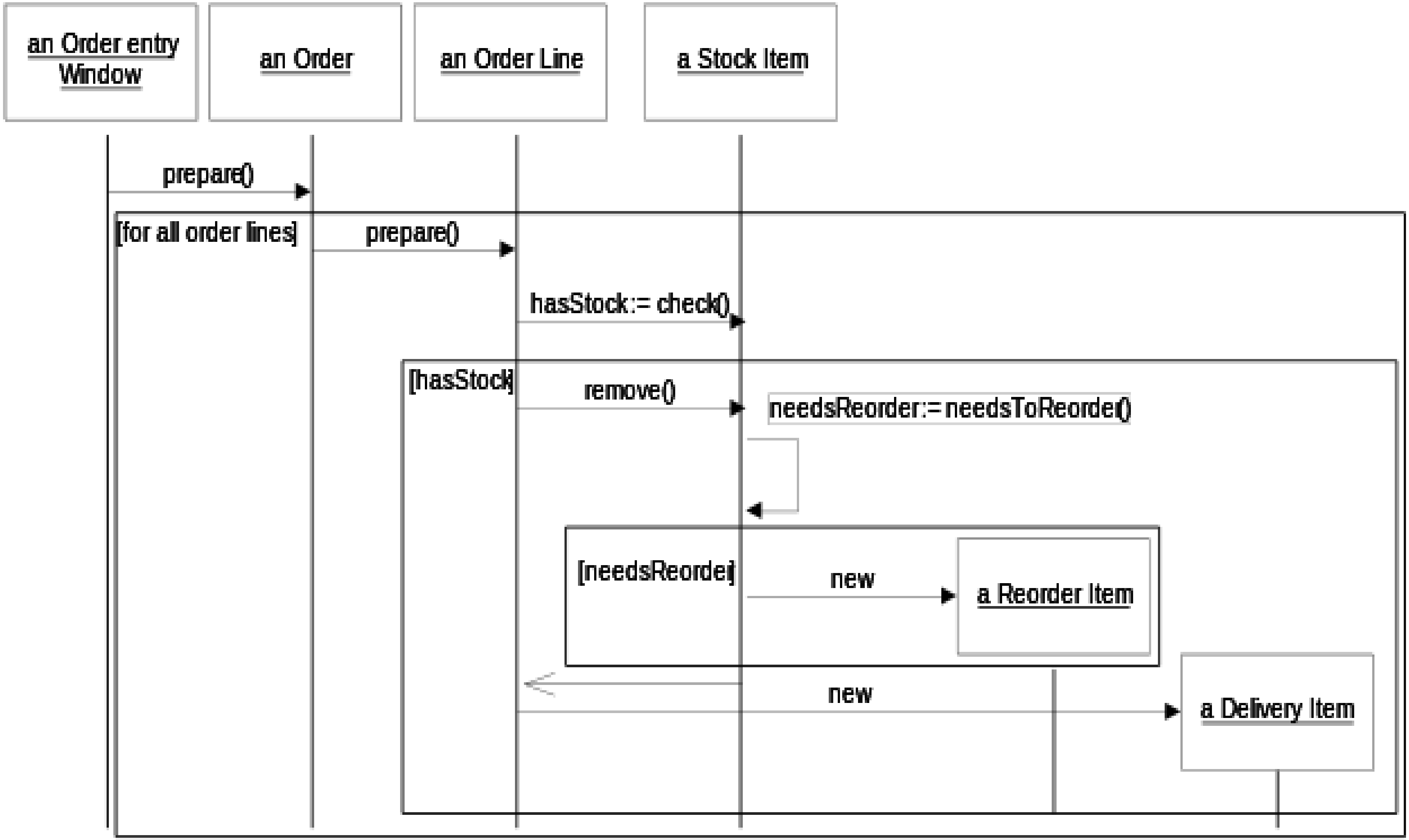
Sequence Diagrams (for cd—v2)



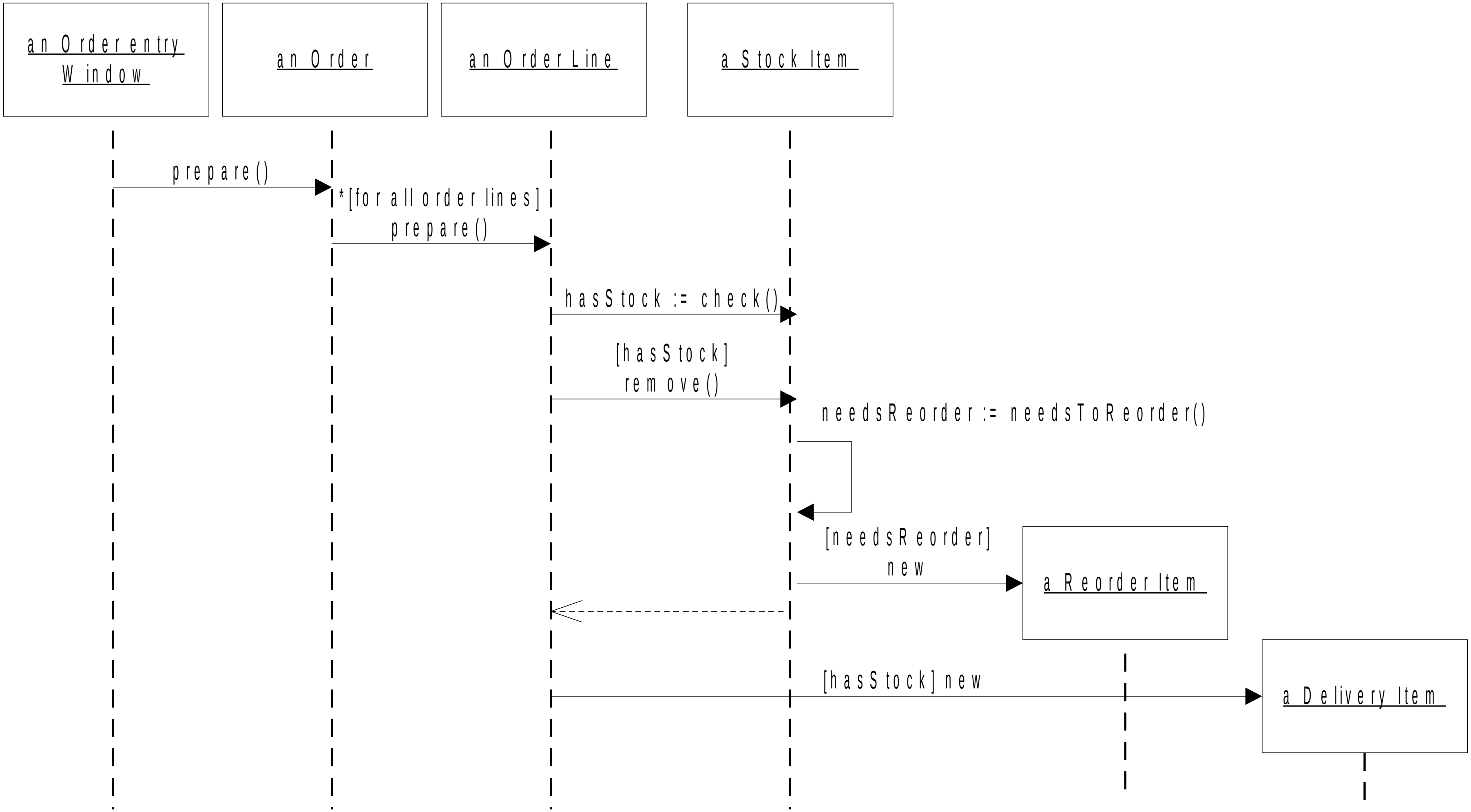
Sequence Diagrams (old) (for cd—v2)



Example from Order Processing

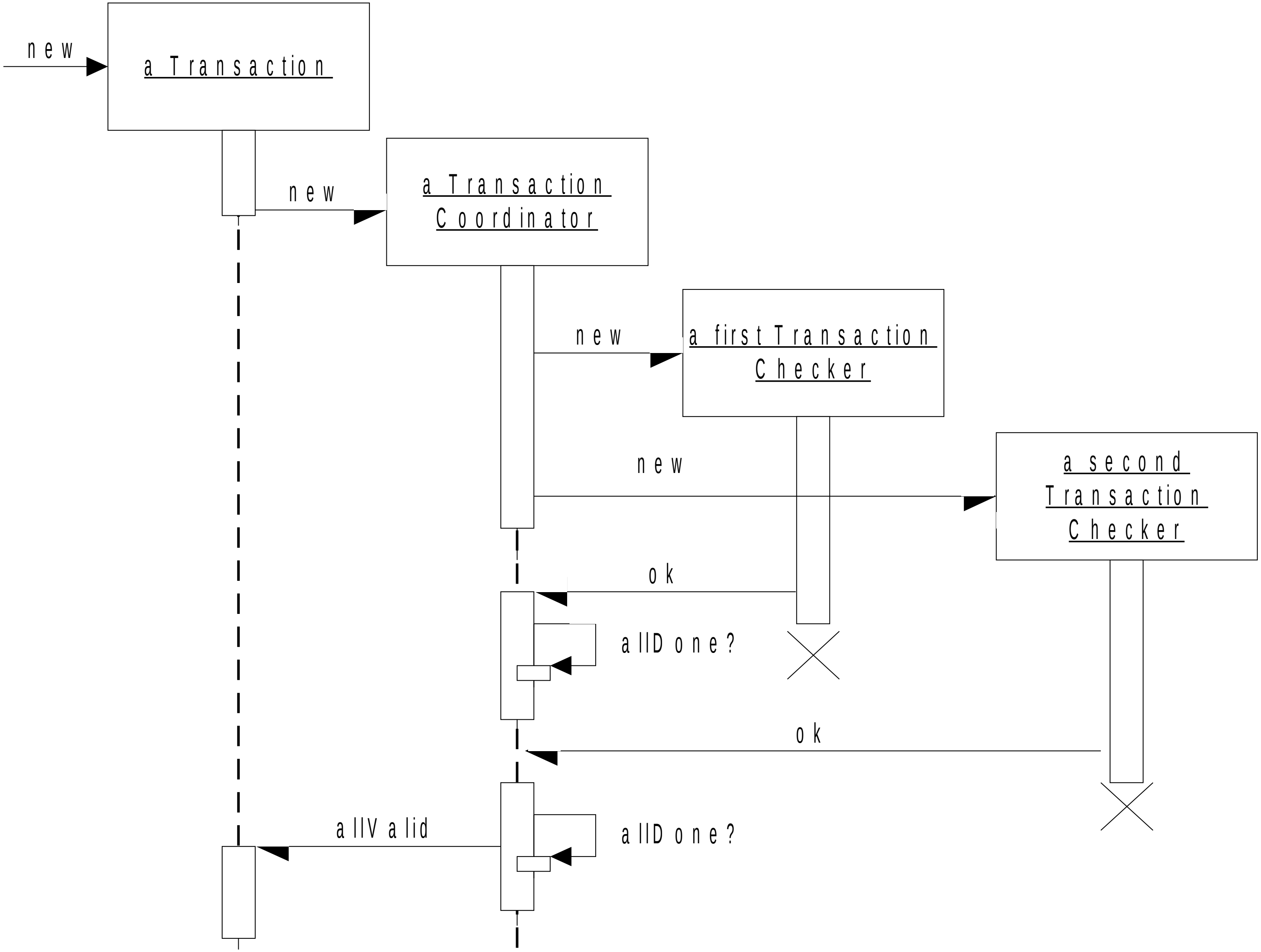


Example from Order Processing (old)

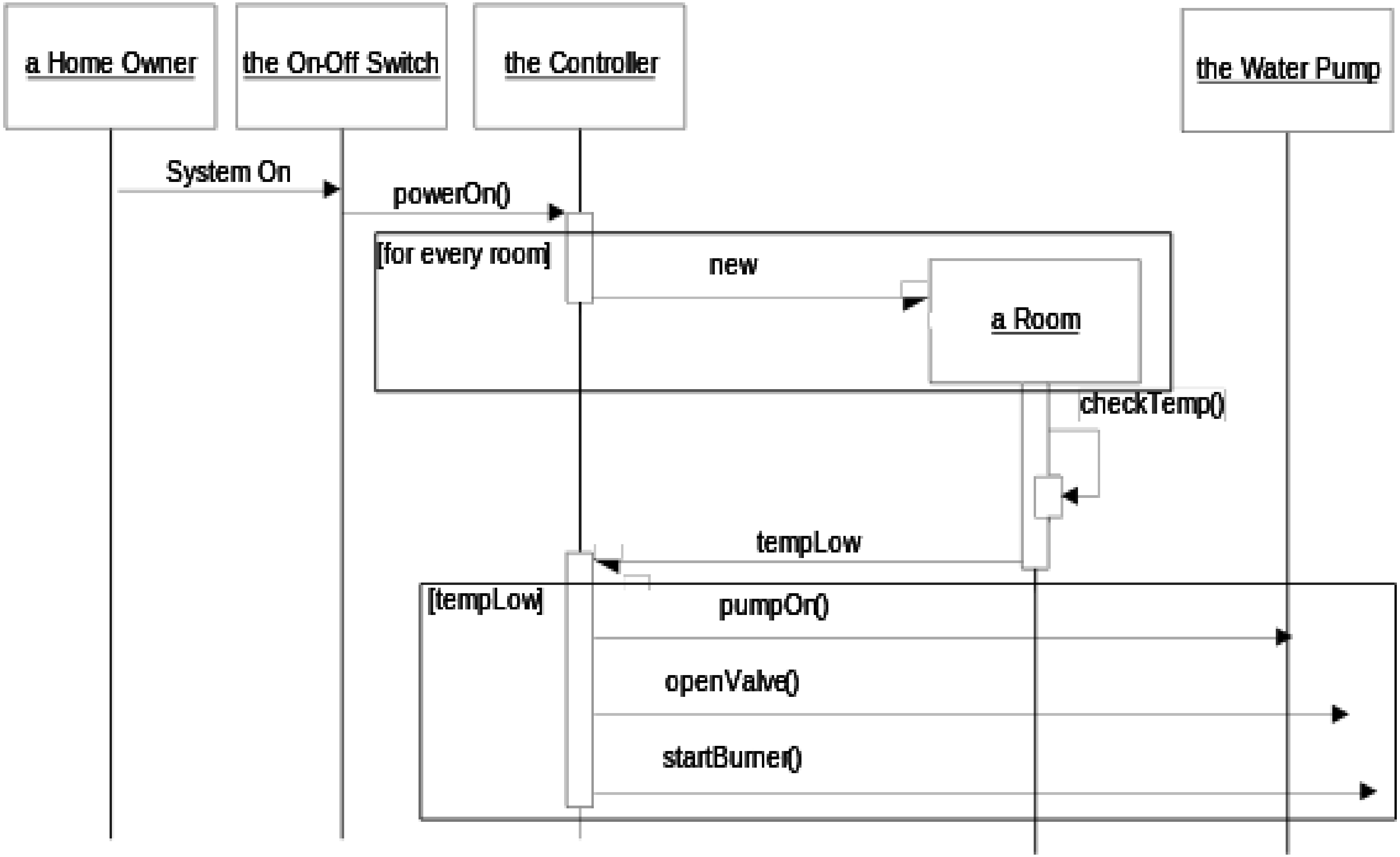


M H

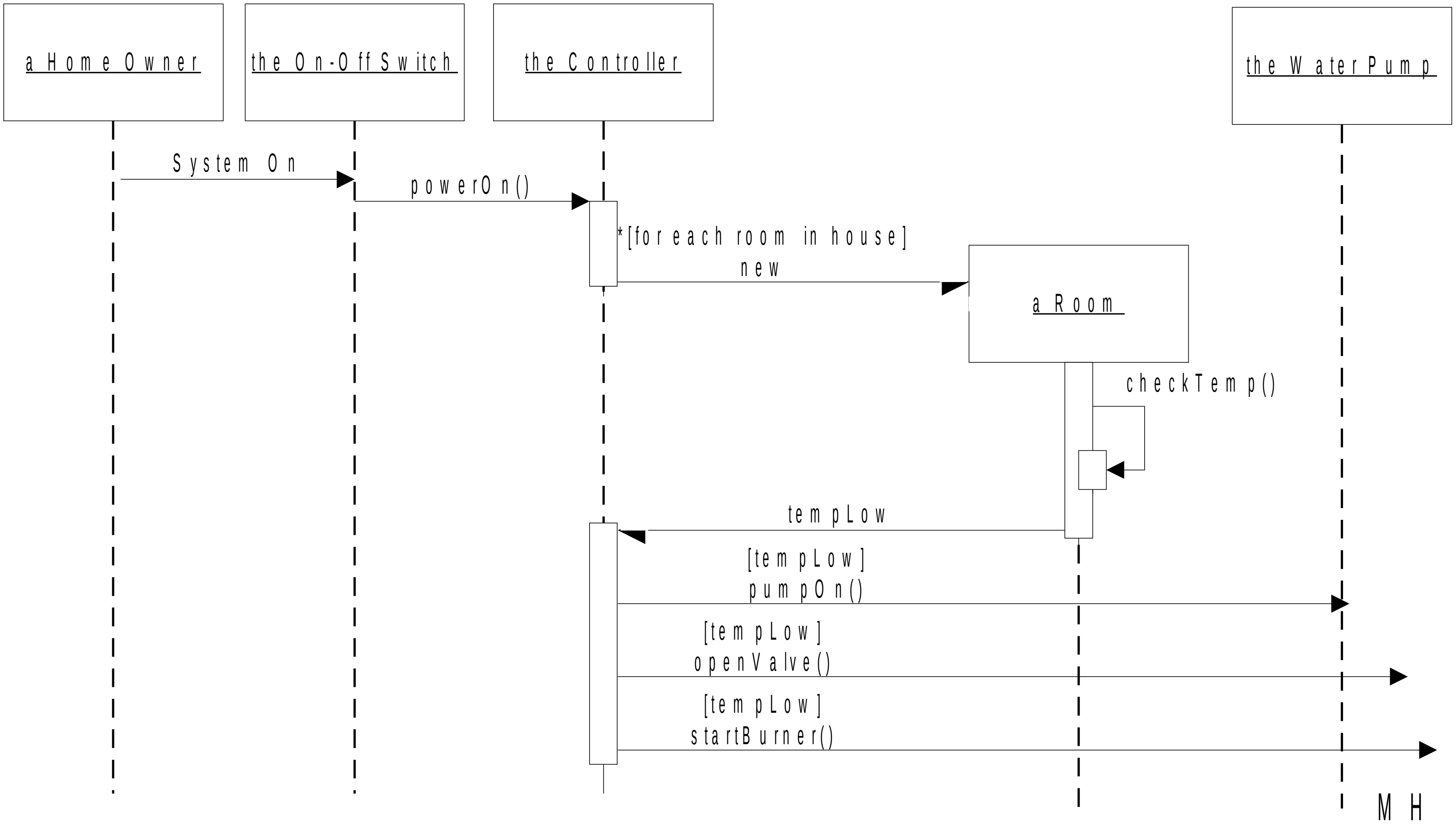
Concurrency


new
a Transaction
new
a Transaction Coordinator
new
a first Transaction Checker
new
a second Transaction Checker
ok
allDone?
ok
allValid
allDone?

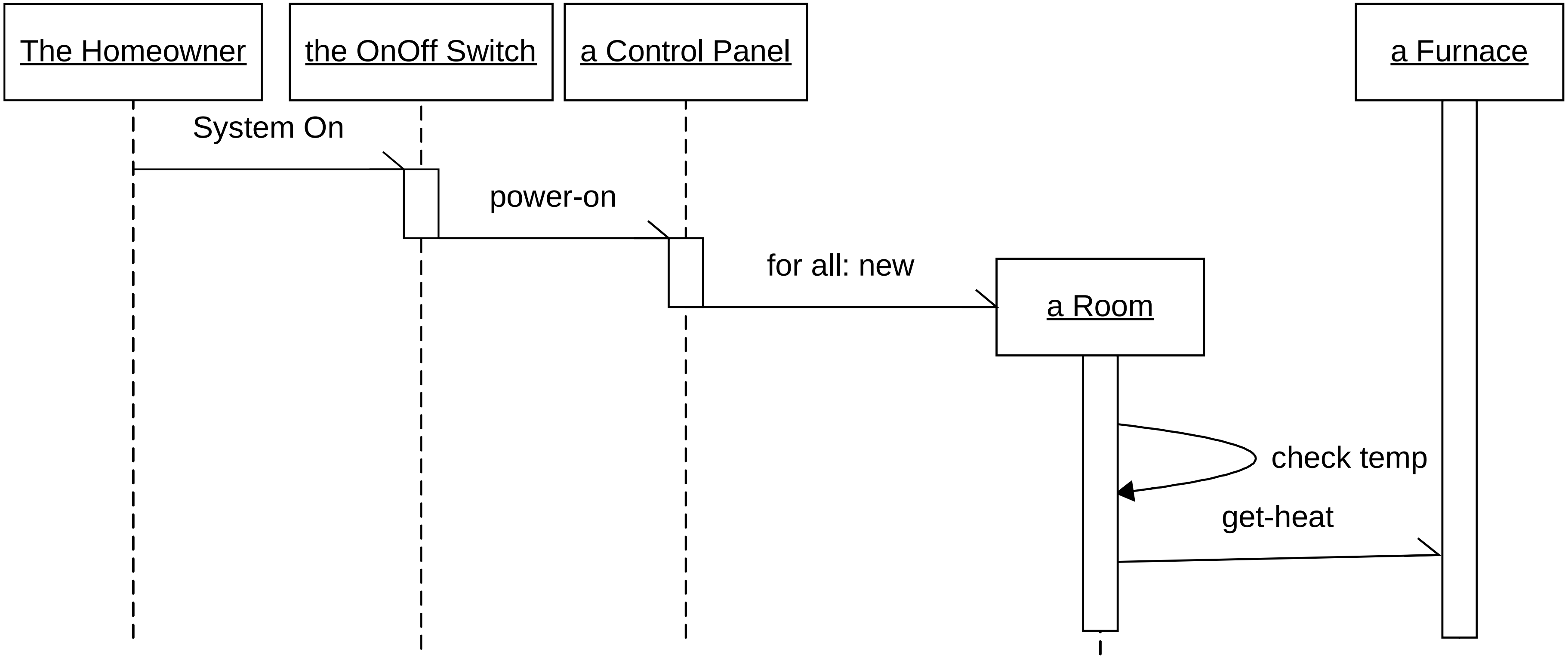
Another Example (for cd—v1)



Another Example (old) (for cd—v1)



Another Example (old) (for cd—v2)



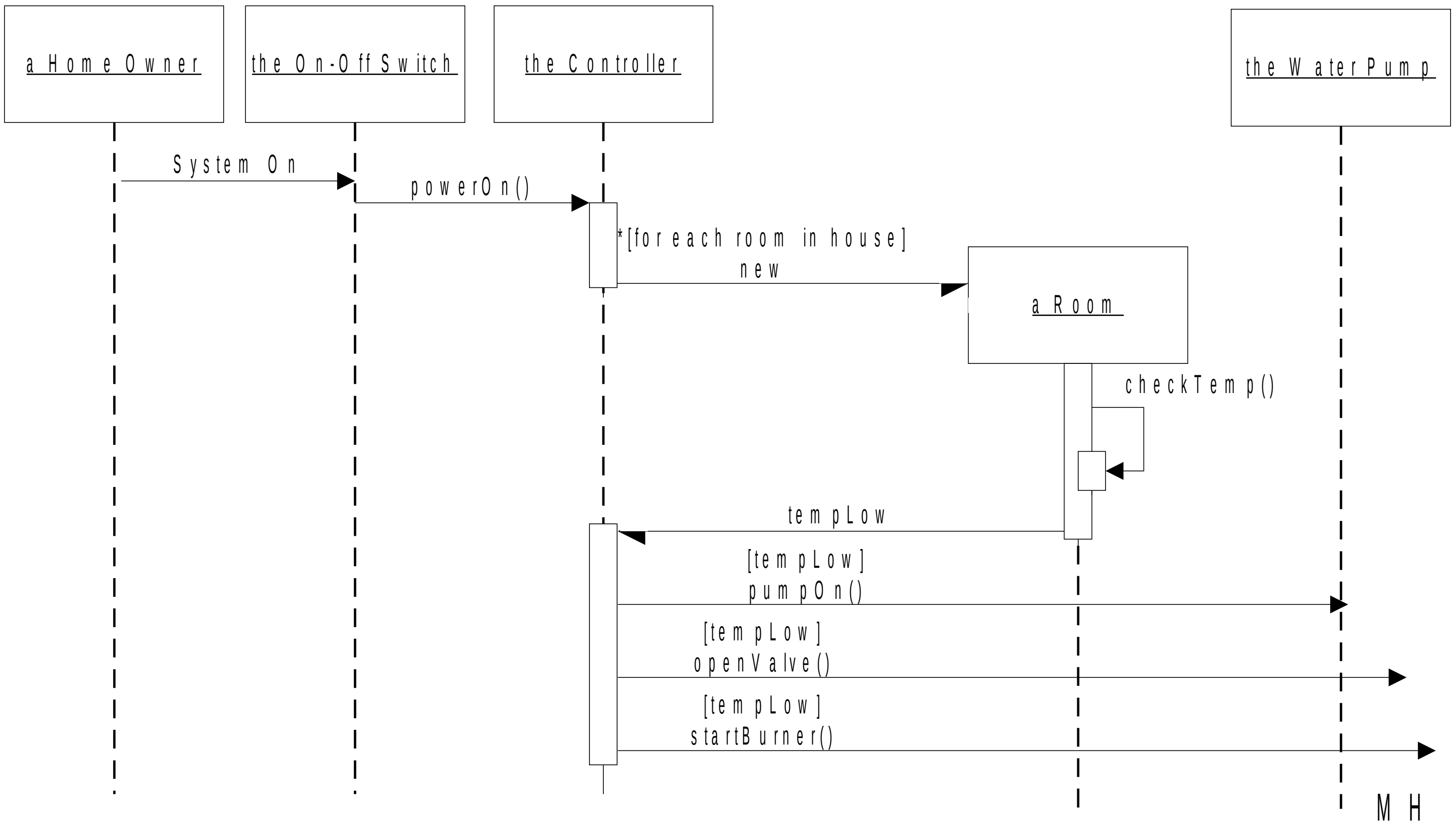
Comment the Diagram (for cd —v1)

When the owner turns the system on

the on switch notifies the controller

The controller creates a room object for each room in the building

The rooms sample the temperature in the toom every 5 s. When a low temp is detected the room notifies the controller.

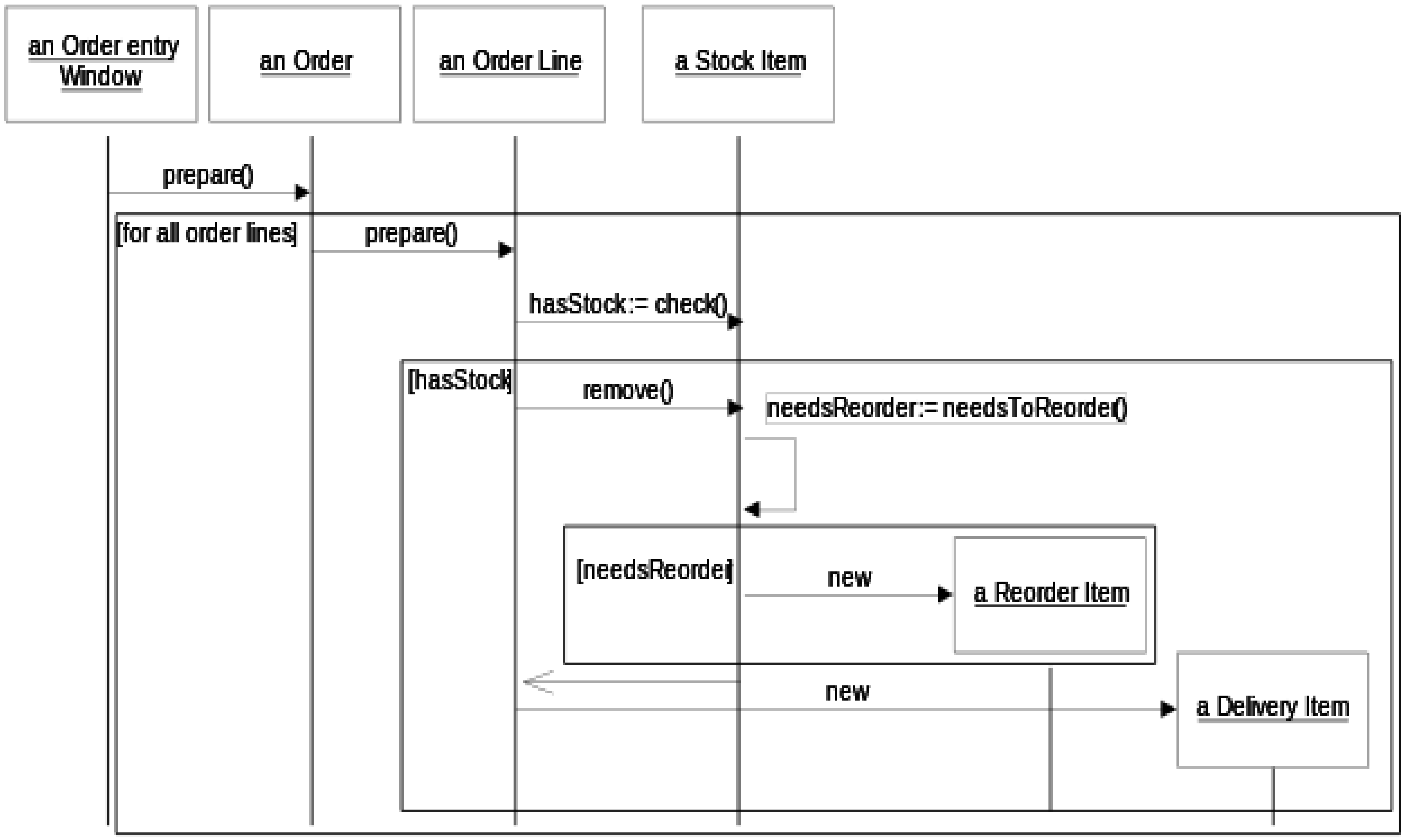


Sequence Diagram Summary

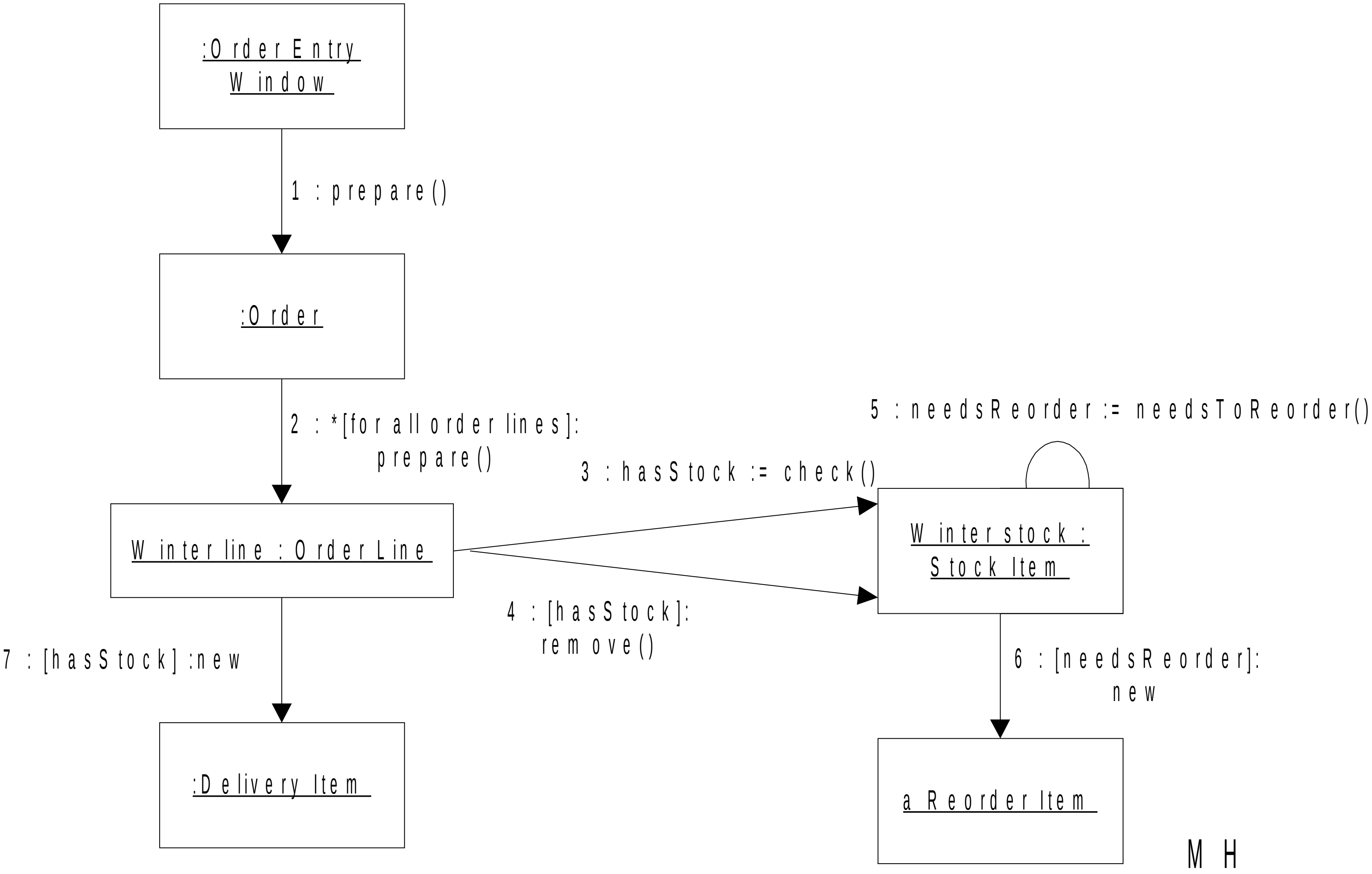
- Highlights the sequencing of of the interactions between objects
  - Shows what messages (information) is passed between objects
  - Shows in which order the messages are sent
  - This is the main emphasis of the diagram
- Allows for concurrency, process creation, and process destruction
- Clarity is the goal – use comments

Demo of Papyrus and draw.io tools

Example from Order Processing



Collaboration Diagrams



:Order Entry Window

1 : prepare()

:Order

2 : *[for all order lines]: prepare()

3 : hasStock := check()

5 : needsReorder := needsToReorder()

Winter line : Order Line

Winter stock : Stock Item

4 : [hasStock]: remove()

7 : [hasStock] :new

6 : [needsReorder]: new

:Delivery Item

a Reorder Item

M H

Collaboration Diagrams Summary

- Highlights the structure of the components (objects) involved in the interaction
  - Better shows how the various objects are related to each other
  - Can help you identify which classes to put in a larger module
- Does the same thing as a sequence diagram, but with a different focus
- Again, clarity is the goal – use comments

Conditional Behavior

- Something you will encounter trying to capture complex use-cases
  - The user does something. If this something is X do this… If this something is Y do something else… If this something is Z…
- Split the diagram into several
  - Split the use-case also
- Use the conditional message
  - Could become messy
- **Remember, clarity is the goal!**

Comparison

- Both diagrams capture the same information
  - People just have different preferences

- I like sequence diagrams
  - They clearly highlight the order of things
  - Invaluable when reasoning about multi-tasking
- Others like collaboration diagrams
  - Shows the static structure
  - Very useful when organizing classes into packages
- I get the structure from the Class Diagrams

When to Use Interaction Diagrams

- When you want to clarify and explore single use-cases involving several objects
  - Quickly becomes unruly if you do not watch it

- If you are interested in one object over many use-cases — **state transition diagrams**

- If you are interested in many objects over many use cases — **activity diagrams**

How to use CRC cards in modeling

CRC Modeling

We Will Cover

- What is CRC modeling?
- What are CRC cards?
- How do we do this?
- This is silly
  - Does it really work?

- Let’s give it a try!

Problems

- In an OO model, no object stands by itself
  - Only operates on its own data
  - Every object has some responsibilities
  - But
  - to get the job done they need to collaborate
- This can be a tough shift in the way of thinking about software

- Immerse the analyst in the object world
  - CRC modeling (Kent Beck and Ward Cunninham)

Class-Responsibility-Collaborator Modeling

- Aids in finding attributes, operations, and associations
- Provides guidelines for what a class *"should be able to do"* and what other classes it will work with and work for
- Based on standard index cards and group meetings
- Three steps
  - Identify classes
  - Define responsibilities
  - Identify collaborators

Identify Classes

- Use guidelines defined earlier
- Capture the class on a CRC card
  - Software support is available

Class Name:	
Class Type:(e.g. device, property, etc.)	
Class Characteristics: (e.g., tangible, atomic, aggregate, etc.)	
Responsibilities:	Collaborators

Order	
Class Type: Property	
Class Characteristics: tangible	
Responsibilities:	Collaborators
Check if items in stock	Order Line
Determine price	Customer
Check for valid payment	
Dispatch to delivery address	

Define Responsibilities

- Follow the guidelines
  - Responsibilities should be evenly distributed
  - State responsibilities in general terms
  - Information should be localized
  - Information related to the responsibility in the same place

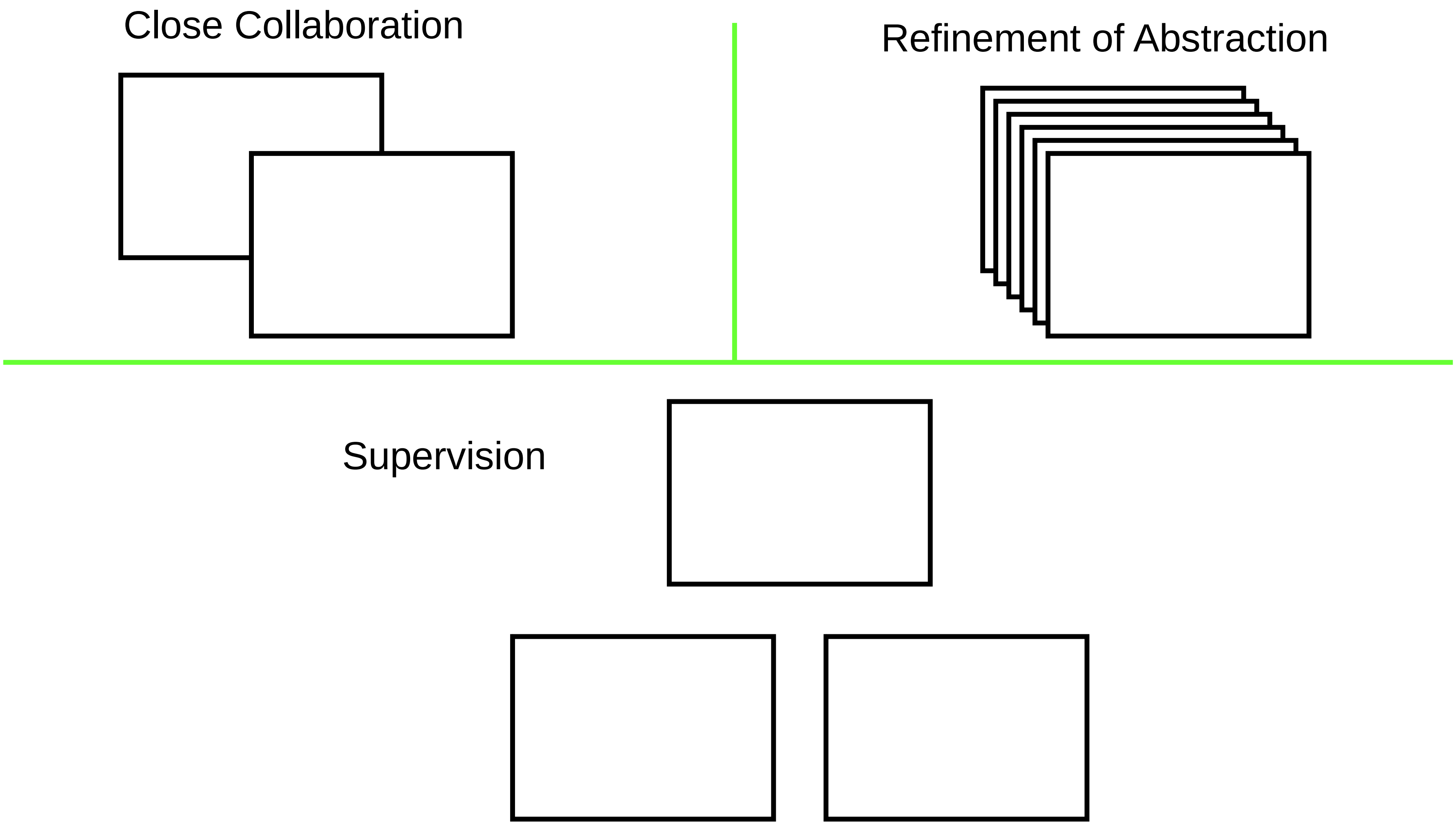
Identify Collaborators

- Classes fulfill their responsibilities in one of two ways
  - It can use its own methods to modify its own attributes
  - It can collaborate with other classes
- If a class cannot fulfill its responsibilities alone, identify and document the collaborators
  - is-part-of
  - has-knowledge-of
  - depends-upon

Perform CRC Review

- Manual review technique with role playing
- Give each participant a collection of CRC cards
- Use scenarios organized into categories
- Review leader reads the scenario
  - Manually execute scenarios
  - Pass token to the “executing” class
  - Call upon collaborators

Organizing the Cards



Close Collaboration

Refinement of Abstraction

Supervision

We Have Learned

- Interaction diagrams show how objects “talk” to each other
  - Sequence diagrams
  - Collaboration diagrams
- Sequence diagrams
  - highlight the ordering of the interactions
- Collaboration diagrams
  - highlight the structure of the collaborators
- Pick the one that fits your needs well
  - Remember, clarity is the goal!!

Next Time

- Design Patterns
  - Highly relevant for the design assignment
  - Design homework due on **October 16th**

- Reading
  - Web resources
  - Fowler Chapter 10 (or equivalent)