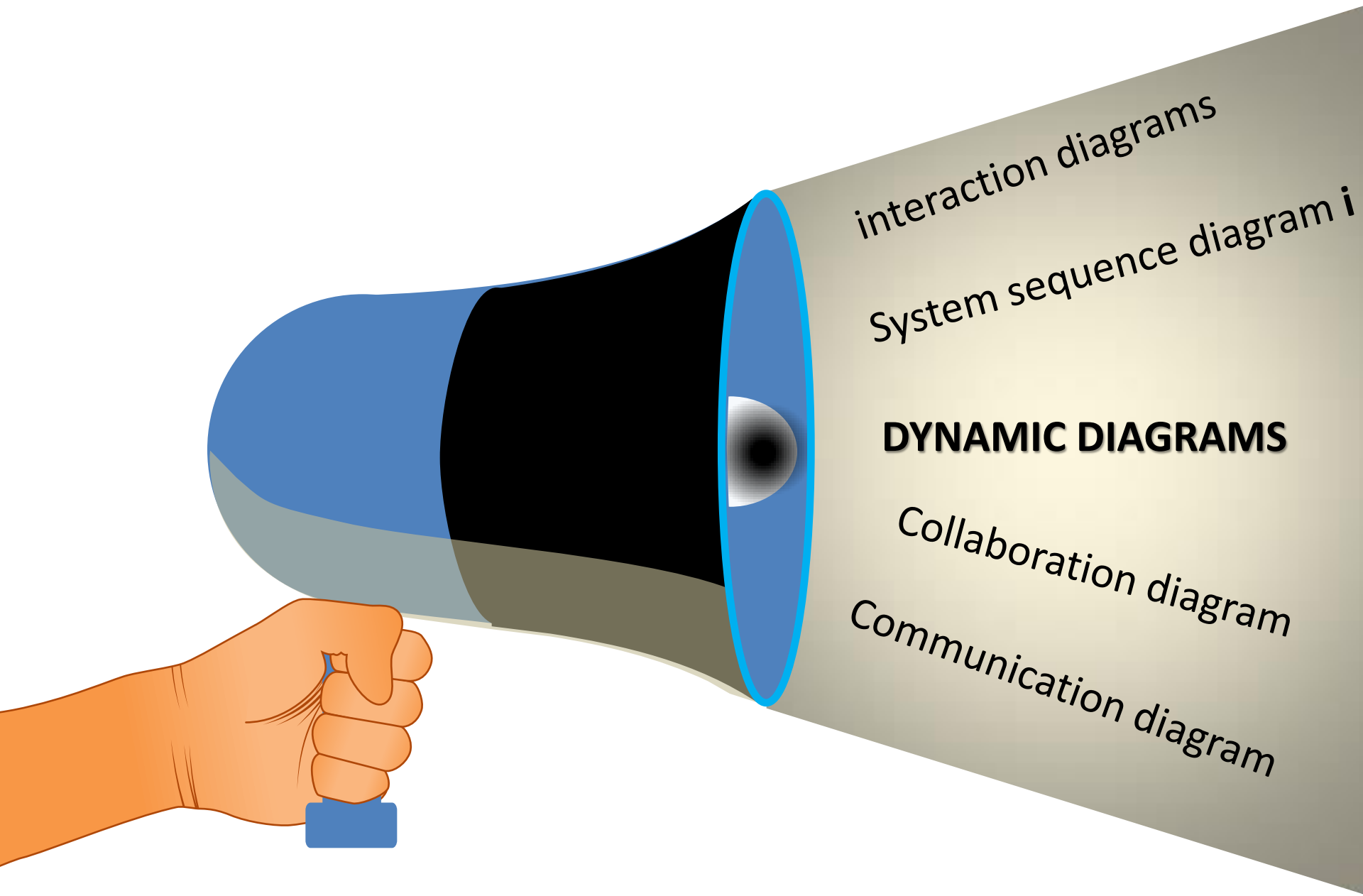
Module -5 Dynamic Diagrams

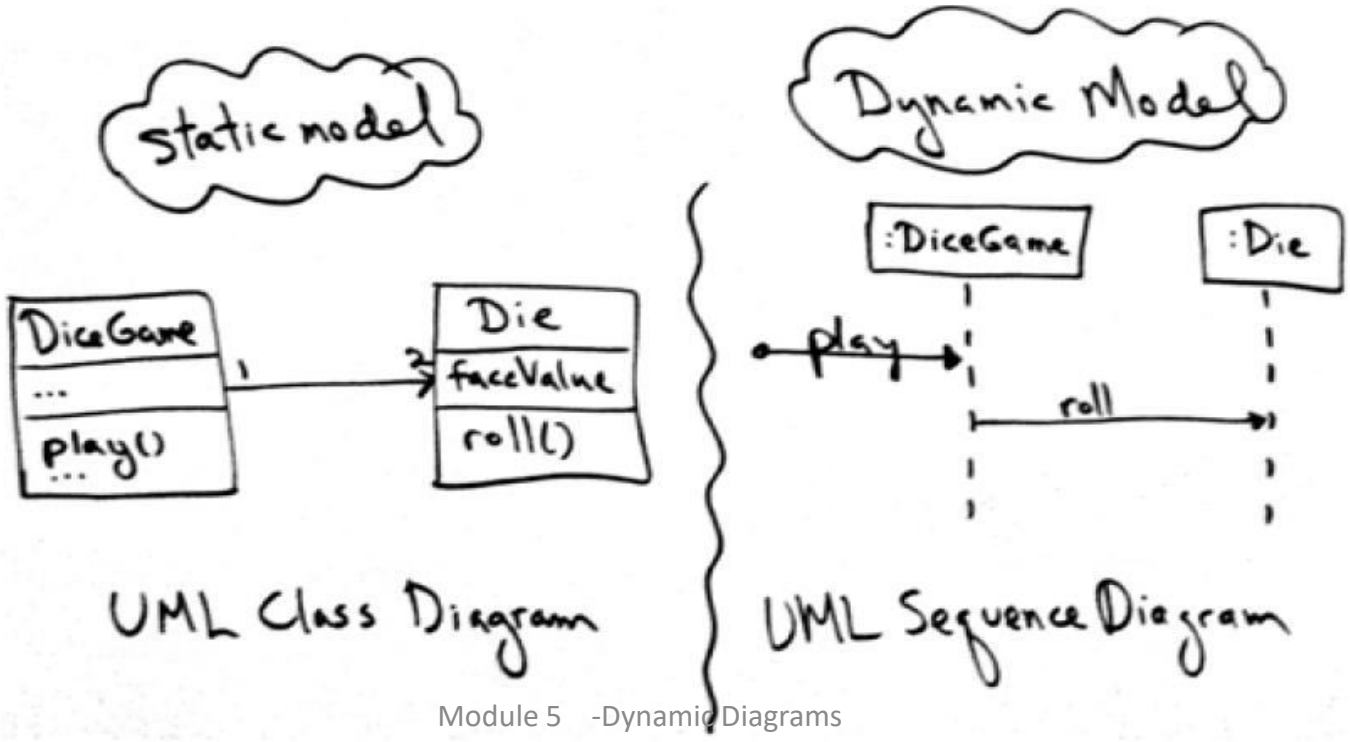
- interaction diagrams
-System sequence diagram
–Collaboration diagram
-Communication diagram

Dr.S.Neelavathy Pari
Assistant Professor (Sr. Grade)
Department of Computer Technology
Anna University, MIT Campus


interaction diagrams
System sequence diagram i
DYNAMIC DIAGRAMS
Collaboration diagram
Communication diagram

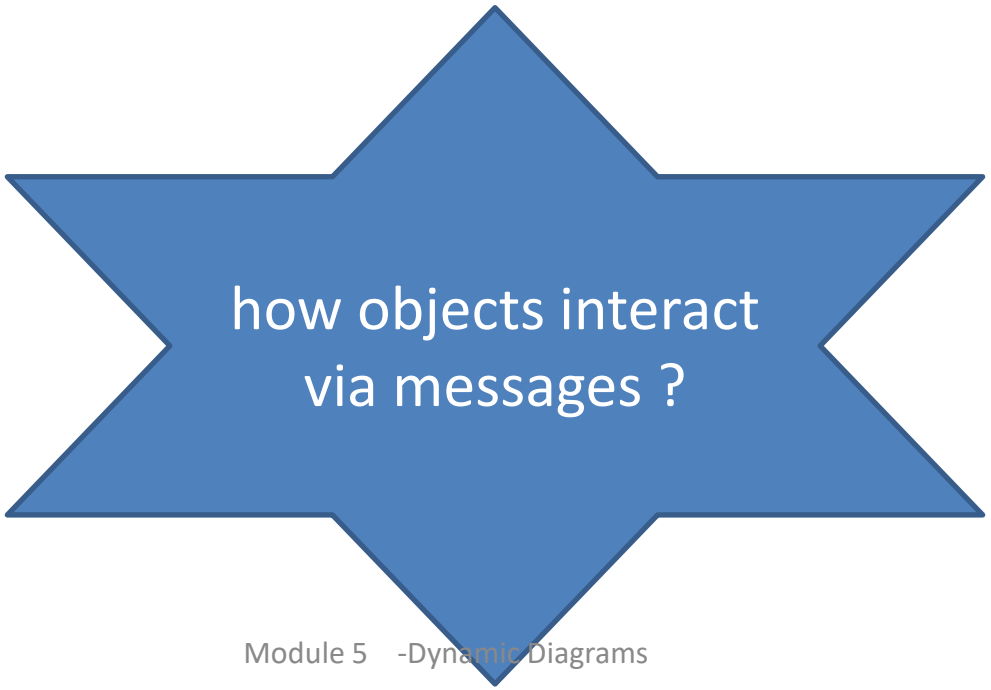
Dynamic Diagrams

- Design the logic,
- Behavior of the code
- Method bodies.



UML Interaction Diagrams

- Provide a reference for frequently used UML interaction diagram notation sequence and communication diagrams.

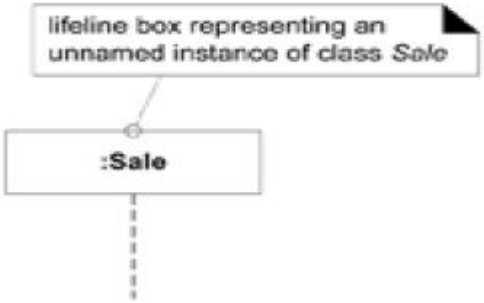


interaction diagram

- The term **interaction diagram is a generalization of two more specialized UML diagram types:**
- sequence diagrams
- communication diagrams

Different Types of Interaction Diagrams

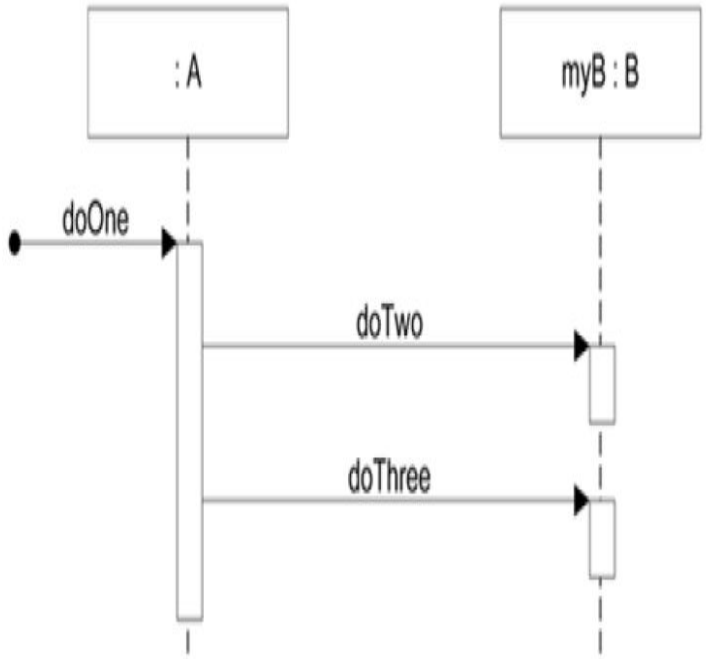
- An Interaction Diagram typically captures a use-case
  - A sequence of user interactions
- **Sequence diagrams**
  - Highlight the sequencing of the intera objects
- Collaboration diagrams
  - Highlight the structure of the components (objects) involved in the interaction



Sequence Diagram

class *A has a method named doOne and an attribute of* type *B. Also, that class B has methods named doTwo and doThree*

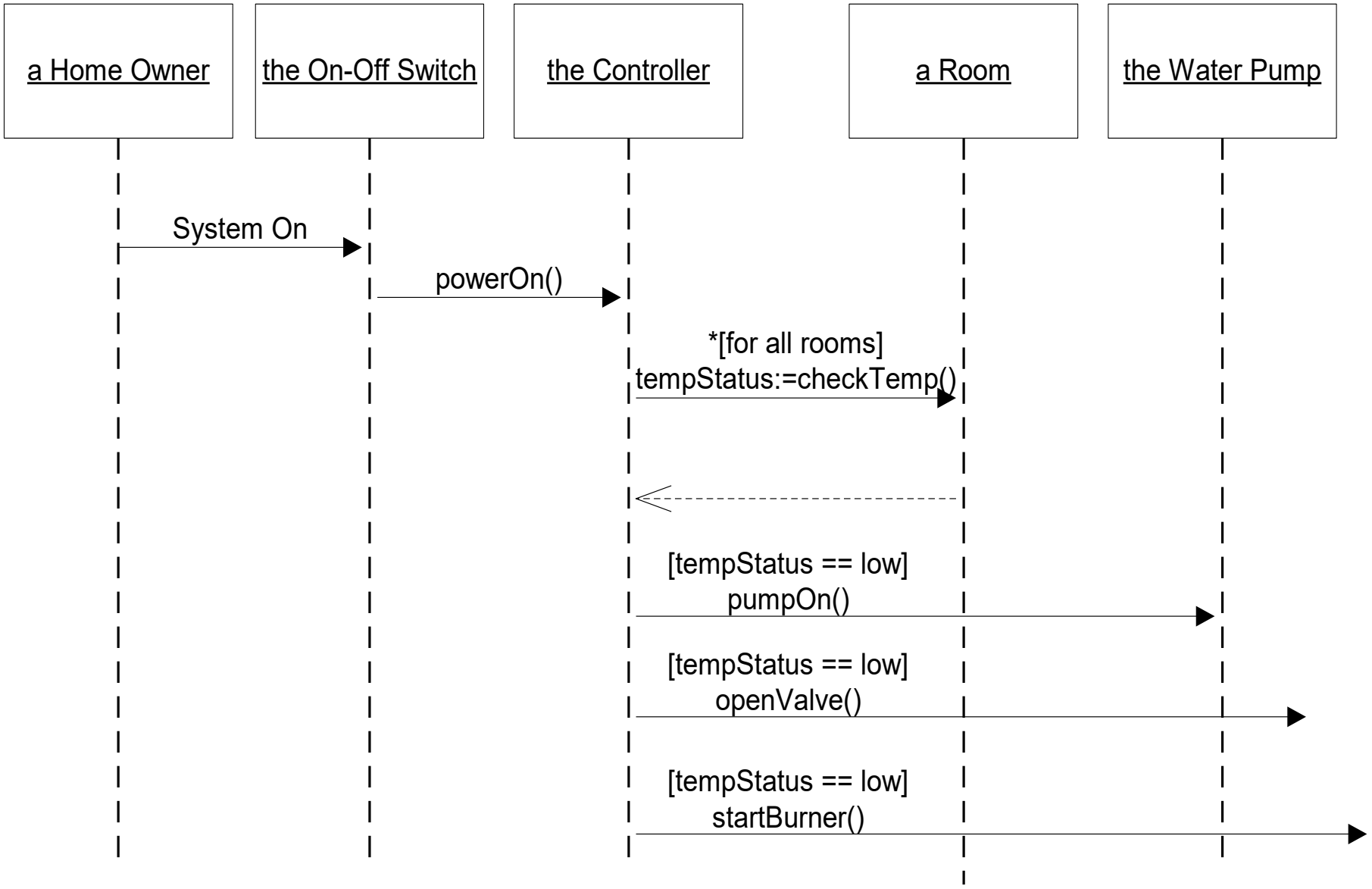
```
public class A
{
private B myB = new B();

public void doOne()
{
   myB.doTwo();
   myB.doThree();
}
// …
```



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

Sequence Diagrams



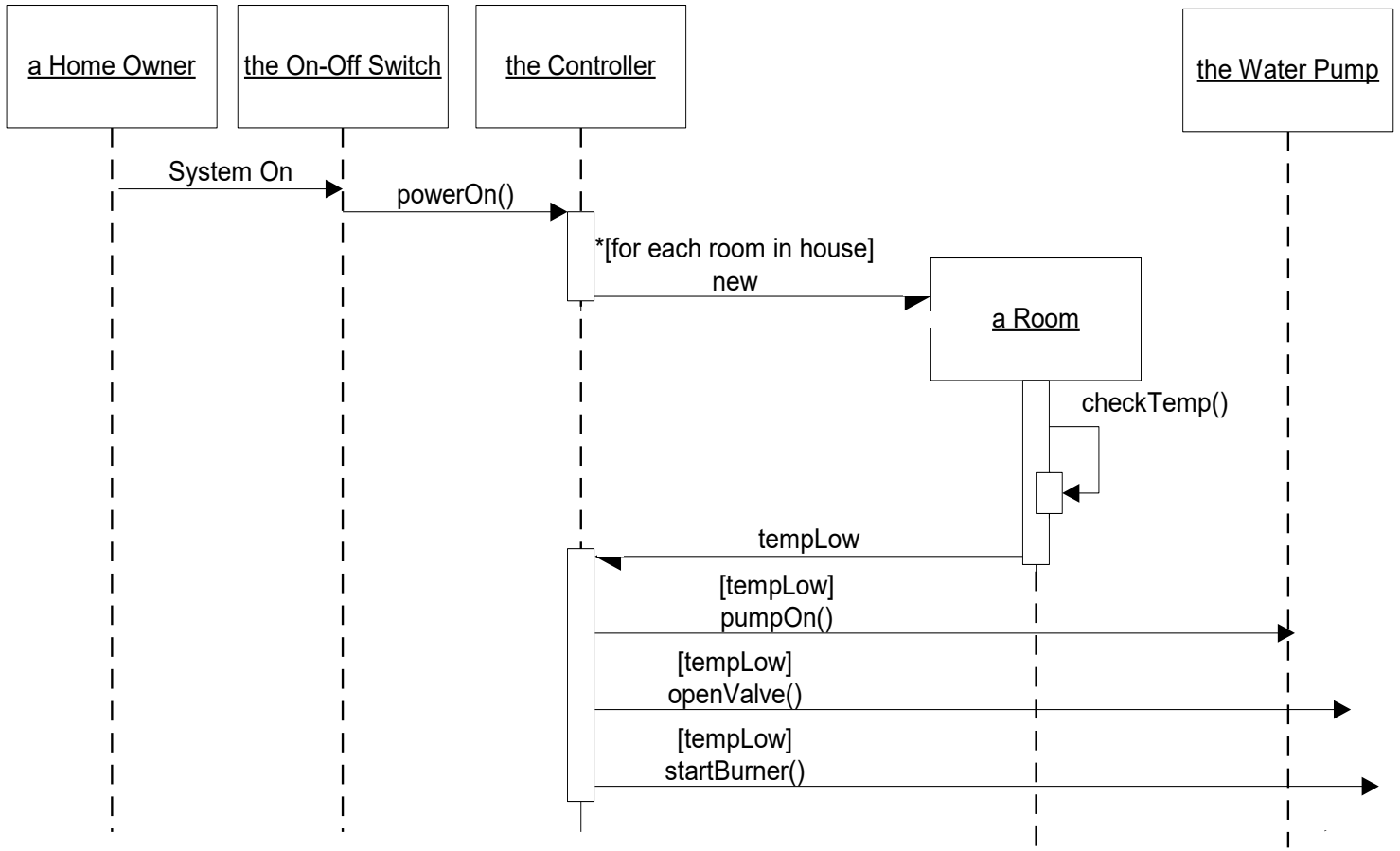
Comment the Diagram

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.



What is UML Collaboration Diagram?

•A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task.

•A Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes

•A Collaboration between objects working together provides emergent desirable functionalities in Object-Oriented systems

•Each object (responsibility) partially supports emergent functionalities

•Objects are able to produce (usable) high-level functionalities by working together

•Objects collaborate by communicating (passing messages) with one another in order to work together

- Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways.

- So, here are some use cases that we want to create a collaboration diagram for:
- Model collaborations between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the messages passing between objects and roles within the collaboration
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes) that participate in use cases
- Each message in a collaboration diagram has a sequence number.
- The top-level message is numbered 1. Messages sent during the same call have the same decimal prefix but suffixes of 1, 2, etc. according to when they occur.

Notations of Collaboration Diagram

- **Objects**
- An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon:
- Object name : class name
- You can use objects in collaboration diagrams in the following ways:
- Each object in the collaboration is named and has its class specified
- Not all classes need to appear
- There may be more than one object of a class
- An object's class can be unspecified. Normally you create a collaboration diagram with objects first and specify their classes later.
- The objects can be unnamed, but you should name them if you want to discriminate different objects of the same class.

- **Actors**
- Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction. If you have several actor instances in the same diagram, try keeping them in the periphery of the diagram.
- Each Actor is named and has a role
- One actor will be the initiator of the use case

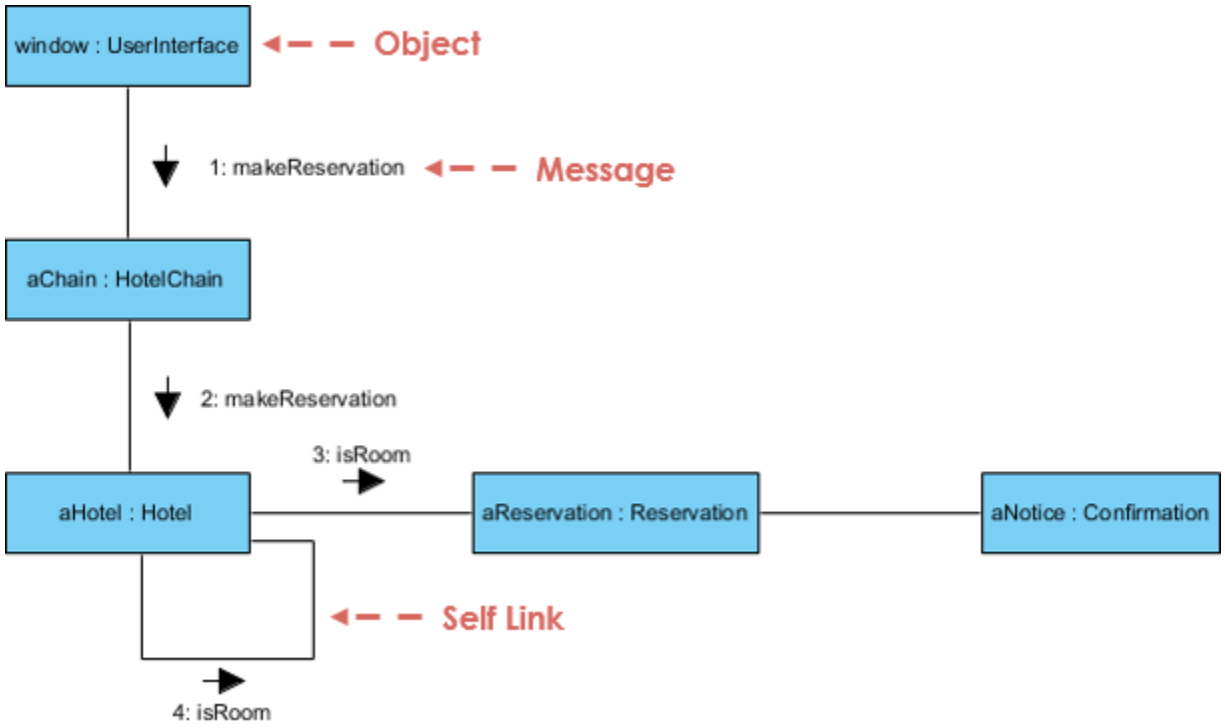
- **Links**
- Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram
- Links are defined as follows:
- A link is a relationship among objects across which messages can be sent. In collaboration diagrams, a link is shown as a solid line between two objects.
- An object interacts with, or navigates to, other objects through its links to these objects.
- A link can be an instance of an association, or it can be anonymous, meaning that its association is unspecified.
- Message flows are attached to links, see Messages.

- **Messages**
- A message is a communication between objects that conveys information with the expectation that activity will ensue. In collaboration diagrams, a message is shown as a labeled arrow placed near a link.
- The message is directed from sender to receiver
- The receiver must understand the message
- The association must be navigable in that direction

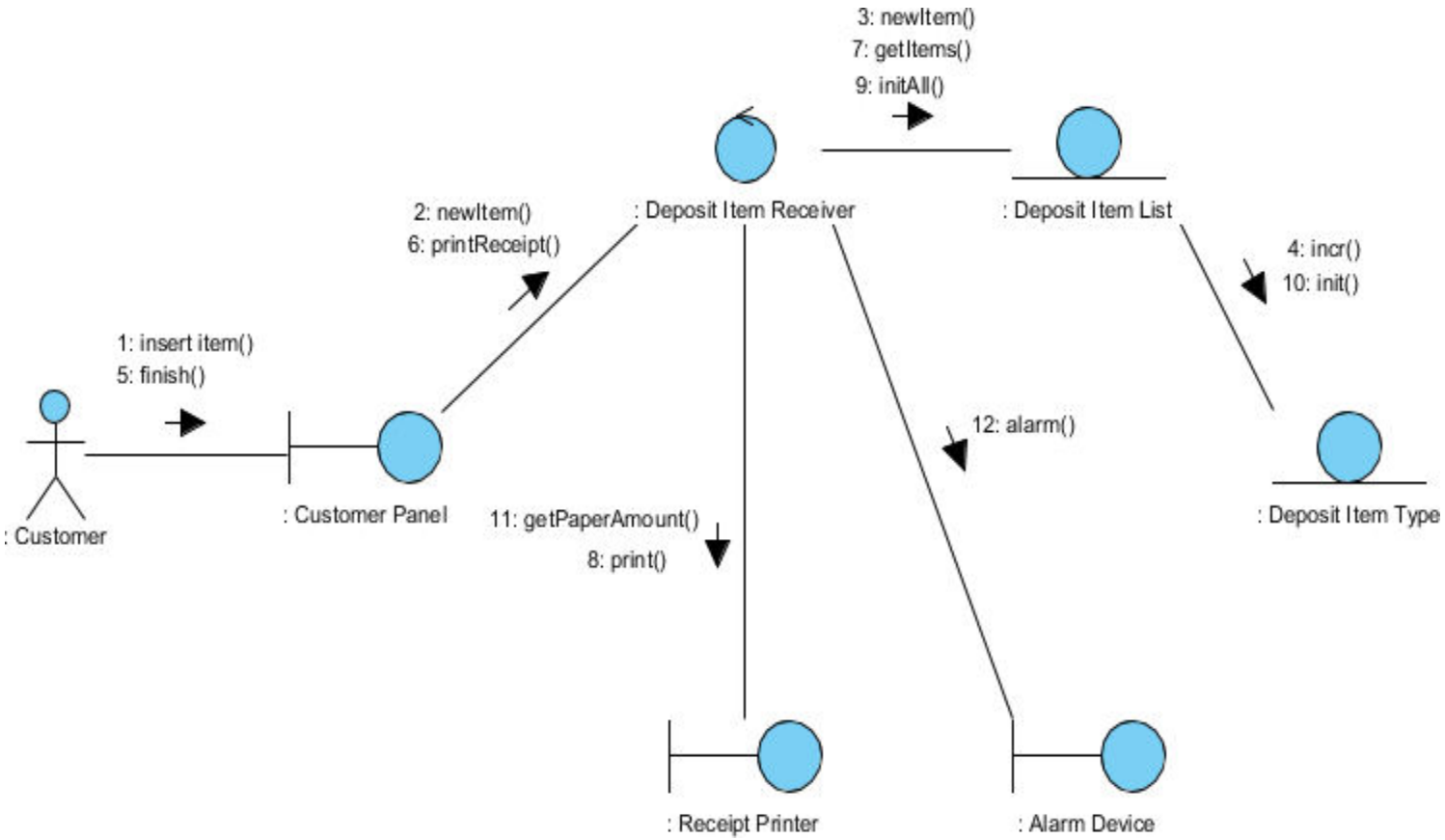
Steps for Creating Collaboration Diagrams

- Identify behavior whose realization and implementation is specified
- Identify the structural elements (class roles, objects, subsystems) necessary to carry out the functionality of the collaboration
  - Decide on the context of interaction: system, subsystem, use case and operation
- Model structural relationships between those elements to produce a diagram showing the context of the interaction
- Consider the alternative scenarios that may be required
  - Draw instance level collaboration diagrams, if required.
  - Optionally draw a specification level collaboration diagram to summarize the alternative scenarios in the instance level sequence diagrams

Collaboration Diagram Example



Collaboration Diagram in Robustness Diagram Format



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
- We prefer 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
- We get the structure from the Class Diagrams

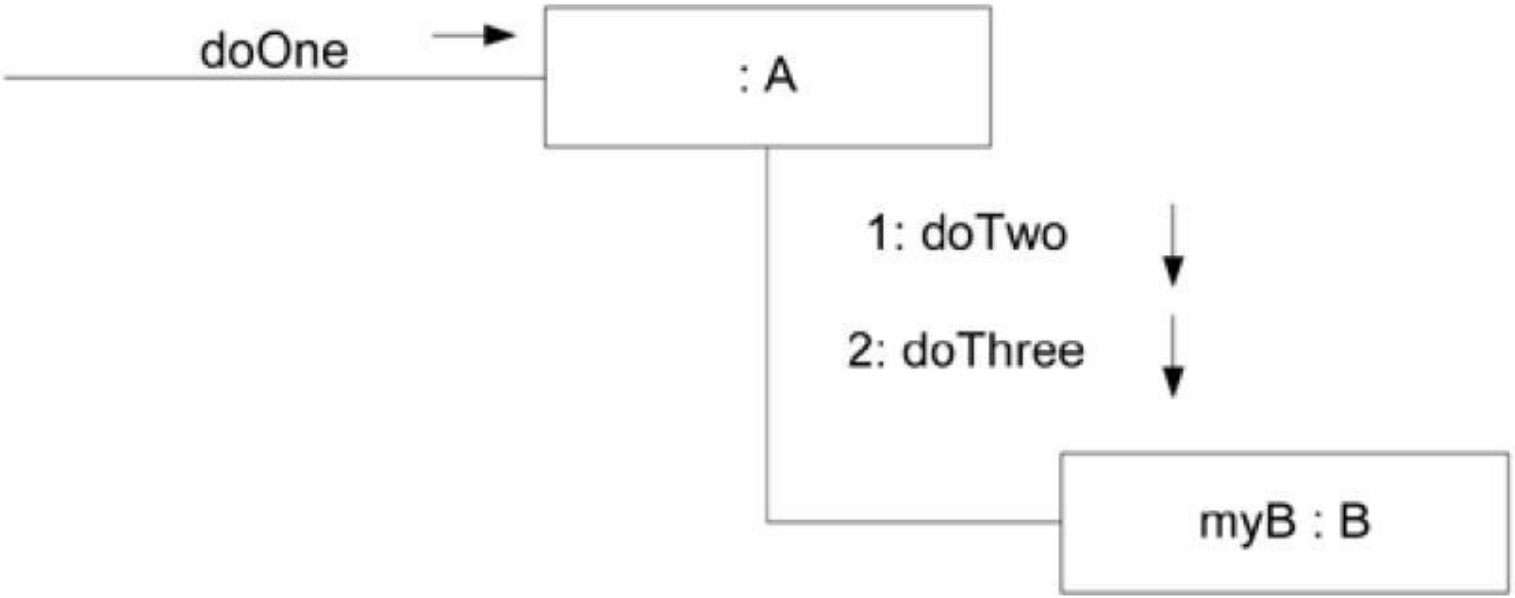
When to Use Interaction Diagrams

- 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**

Communication diagrams

- **Communication diagrams illustrate object interactions in a graph or network format, in which objects can be** placed anywhere on the diagram (the essence of their wall sketching advantage),

Communication diagrams



Identify the Strength and weakness of Sequence and Communication diagram

some related code for the *Sale class and its makePayment method*

```
public class Sale
{
private Payment payment;

public void makePayment( Money cashTendered )
{
    payment = new Payment( cashTendered );
    //…
}
// …
}
```

Construct sequence and communication diagram