

Notation Part 2

Object Orientated Analysis and Design

Benjamin Kenwright

Outline

- Review
- What do we mean by Notation and UML?
- Types of UML View
- **Continue UML Diagram Types**
- Conclusion and Discussion
- Summary

Revision Question

- Write down the UML Class Diagram visibility attributes:

Public _____

Private _____

Protected _____

Package _____

Answer

- Public (+) Visible to any element that can see the class
- Protected (#) Visible to other elements within the class and to subclasses
- Private (-) Visible to other elements within the class
- Package (~) Visible to elements within the same package

Revision Question

- Draw the notations for the different types of relationships

Dependency



Association







Direct Association

Inheritance

Realization

Aggregation

Answer

Dependency	
Association	
Direct Association	
Inheritance	
Realization	
Aggregation	

Question

■ Is an “Activity Diagram” a static or dynamic system model?

a) Static (Structural)

b) Dynamic (Behavioral)

Answer

b) Dynamic (Behavioral)

Revision Question

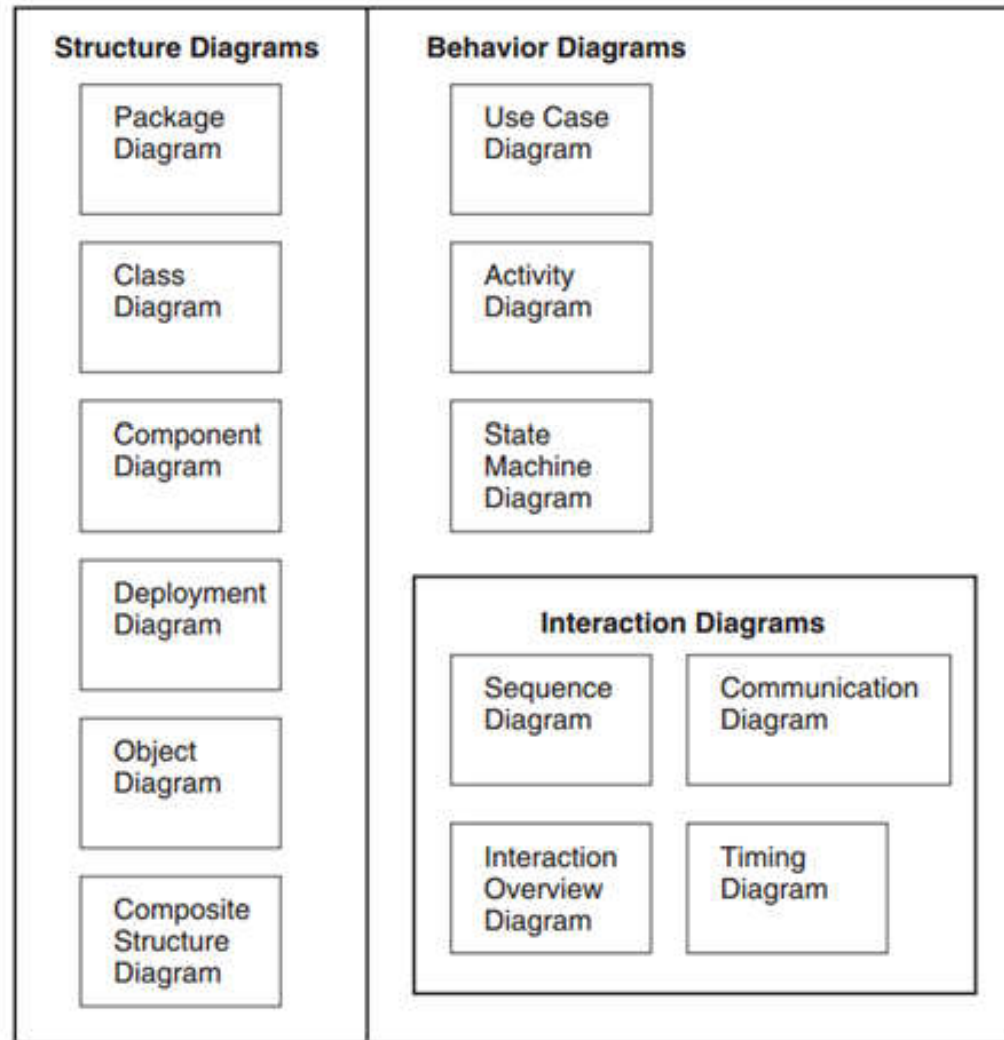
- List the various UML Diagram Types

Structure Diagrams	Behavior Diagrams
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	
<input type="text"/>	
<input type="text"/>	
<input type="text"/>	

Interaction Diagrams

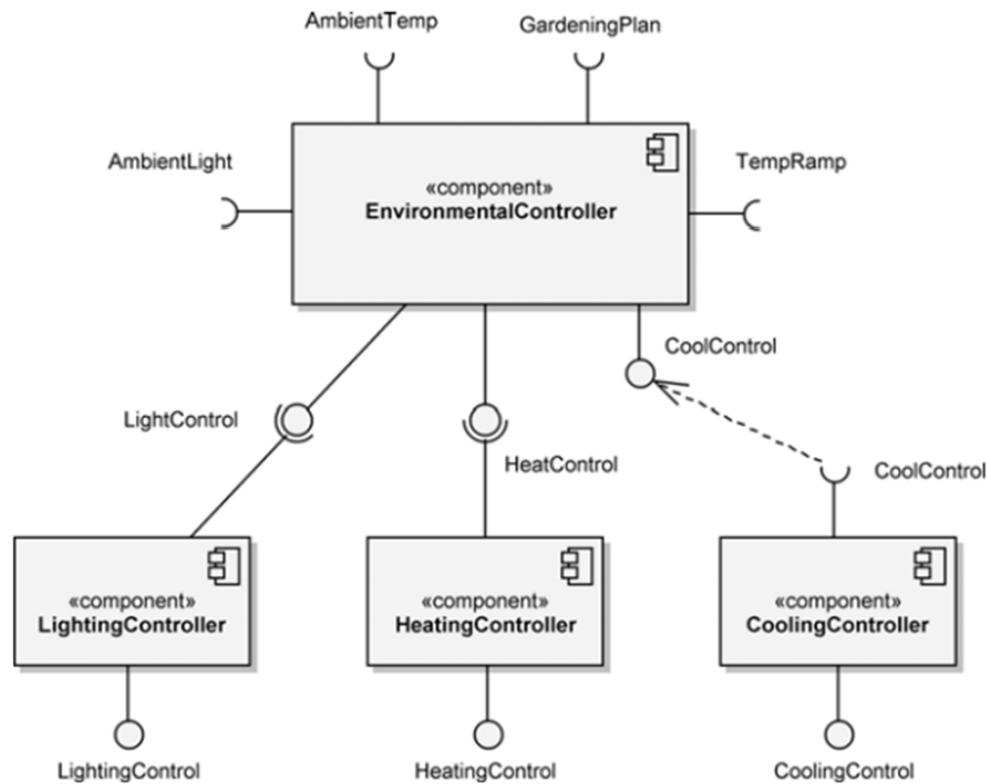
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

Answer



Question

■ What type of Diagram is this:



Answer

■ Component Diagram

A component diagram shows the internal structure of components and their dependencies with other components

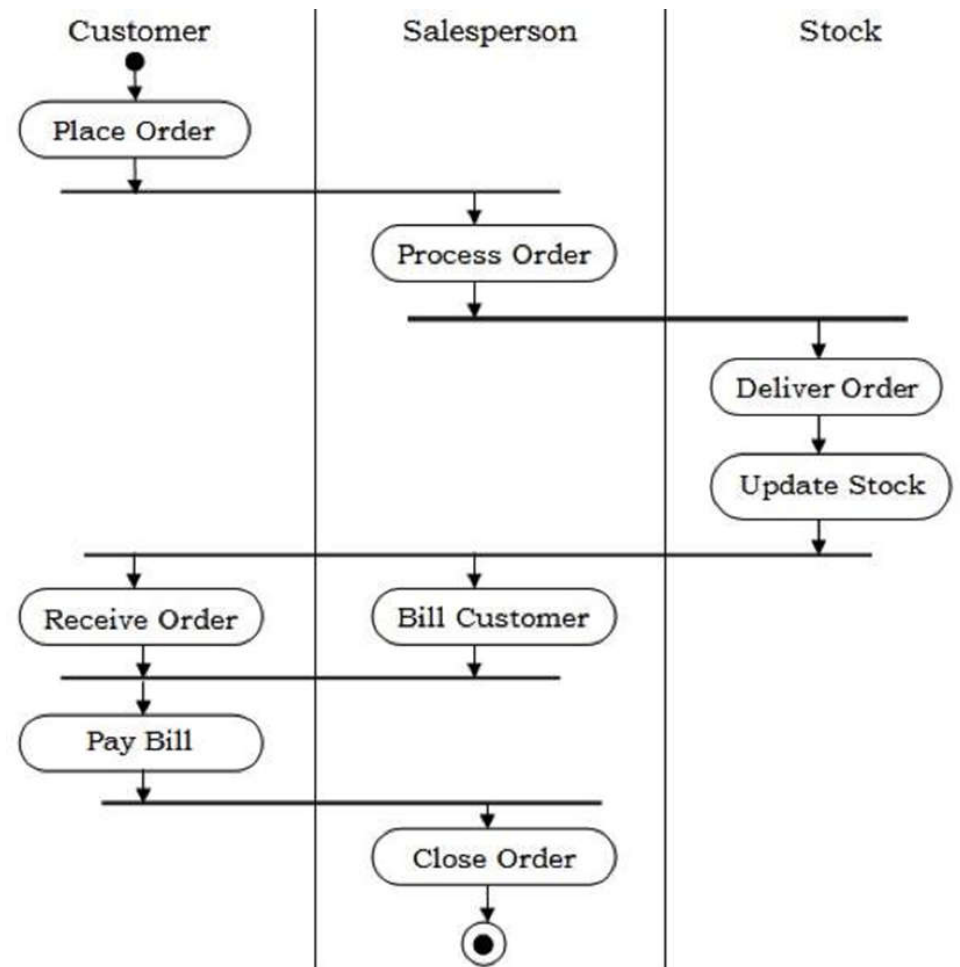
Question

- Draw a simple Activity Diagram?

Answer

- Activity diagrams provide visual depictions of the flow of activities, whether in a system, business, workflow, or other process

Example



Review

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams

Last Week

- **Sequence Diagrams**
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

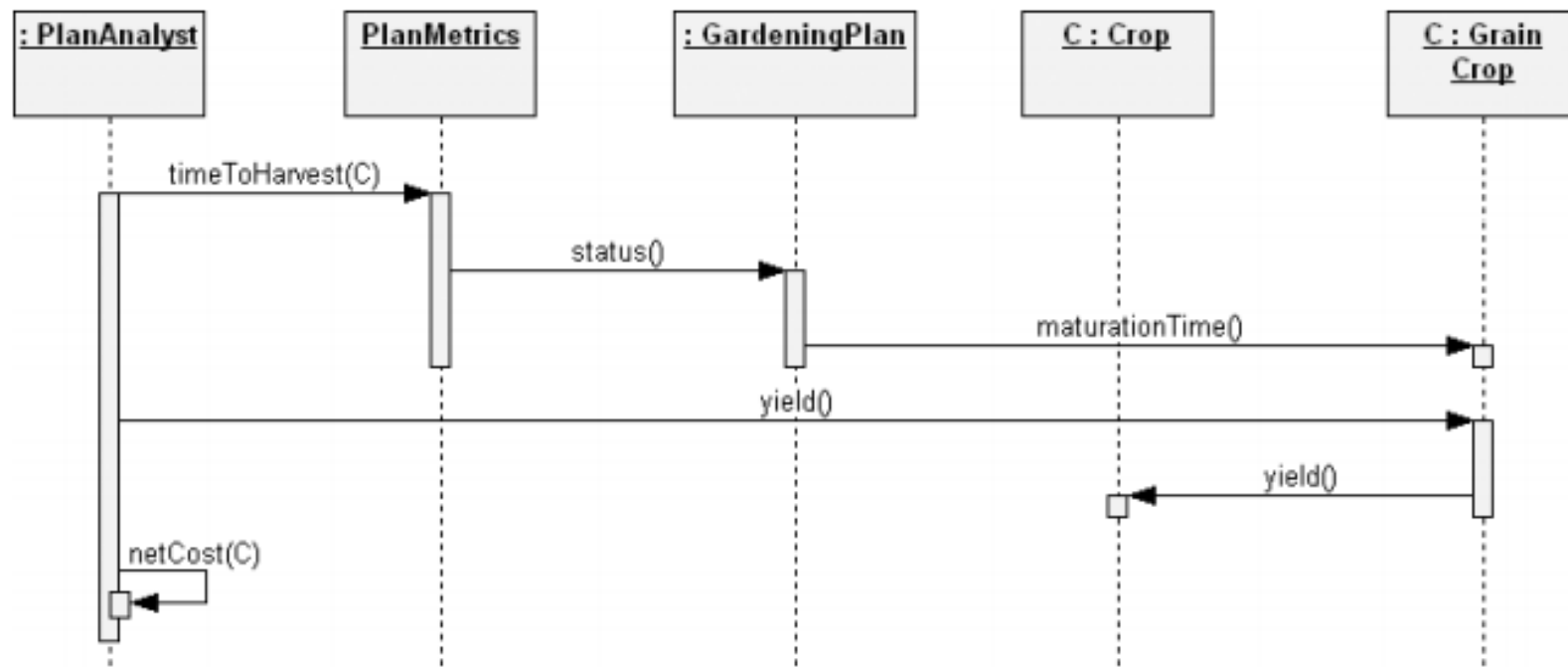
Sequence Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- **Sequence Diagrams**
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Sequence Diagram

- A sequence diagram **traces** the **execution** of a scenario in the same context as an object diagram. To a large degree, a sequence diagram is simply another way to represent an object diagram

Example



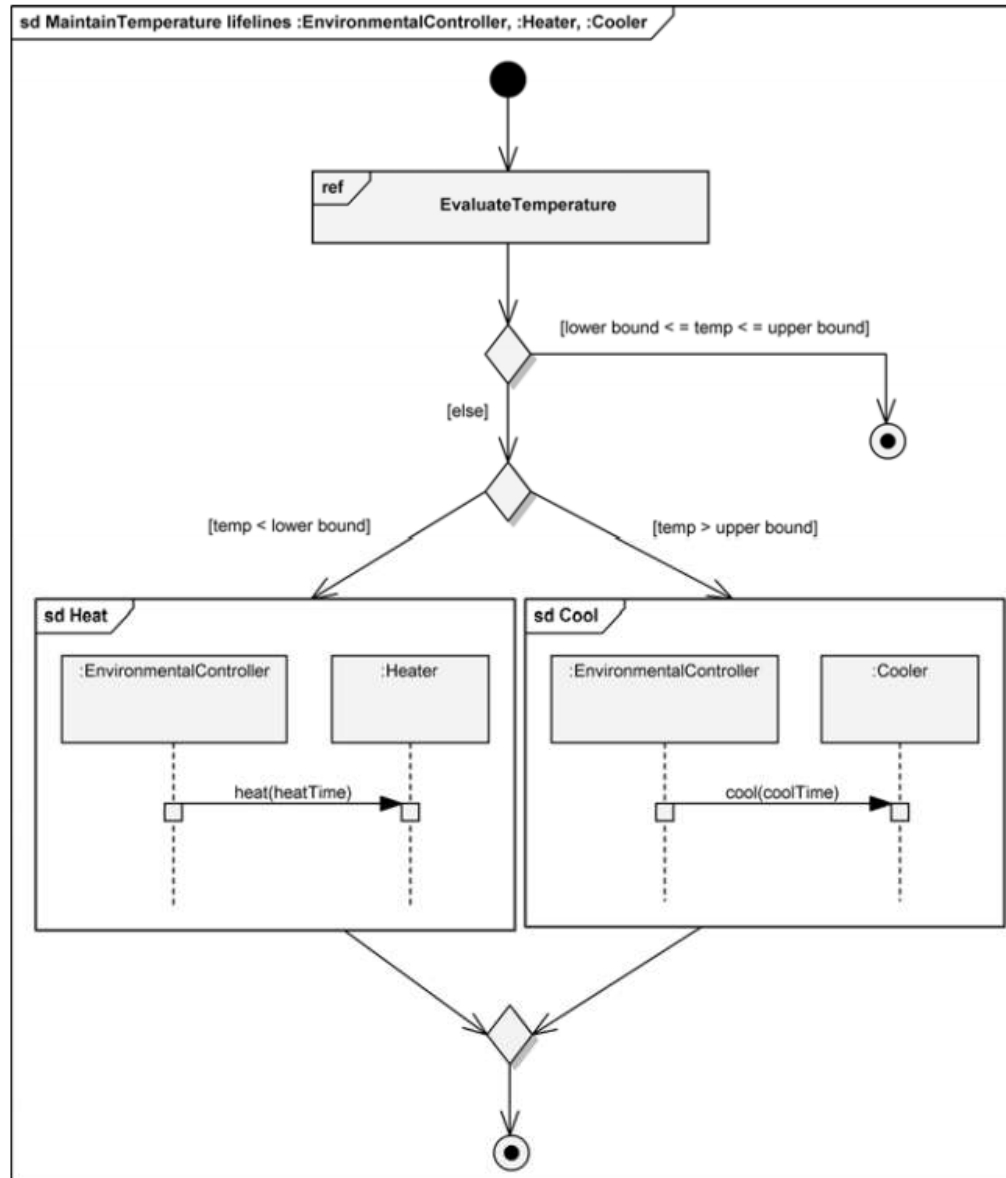
Interaction Overview Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- **Interaction Overview Diagrams**
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Interaction Overview Diagram

- Interaction overview diagrams are a combination of **activity diagrams** and **interaction diagrams**
- Intended to provide an overview of the flow of control between interaction diagram elements

Example



The Interaction Overview Diagram for MaintainTemperature

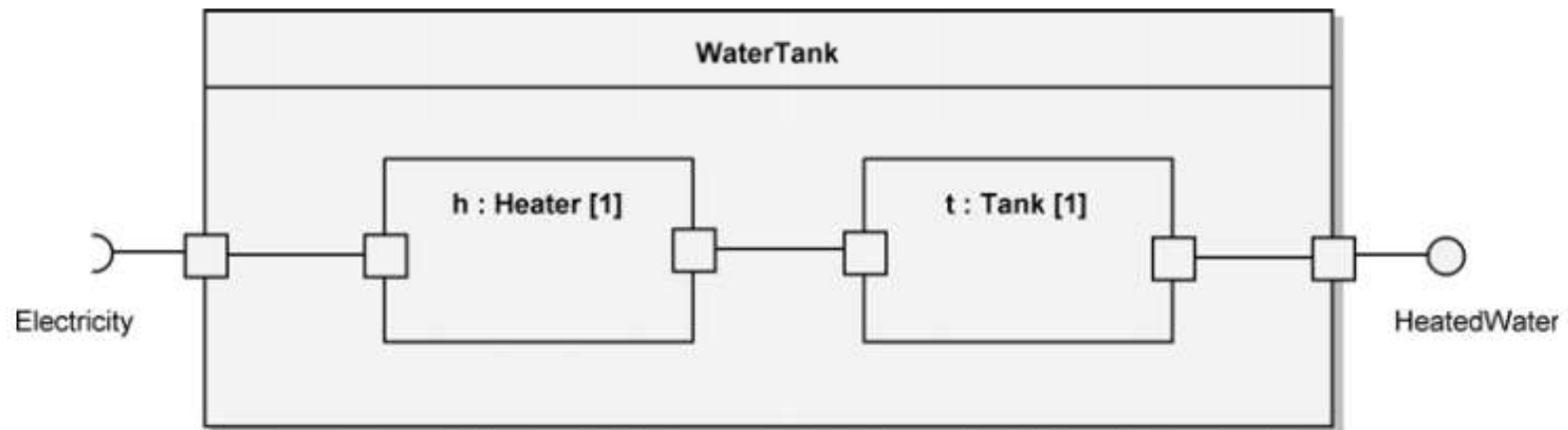
Composite Structure Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- **Composite Structure Diagrams**
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Composite Structure Diagram

- Composite structure diagrams provide a way to depict a structured classifier with the definition of its internal structure.
- This internal structure is comprised of parts and their interconnections, all within the namespace of the composite structure.

Example



The Composite Structure Diagram for `WaterTank`

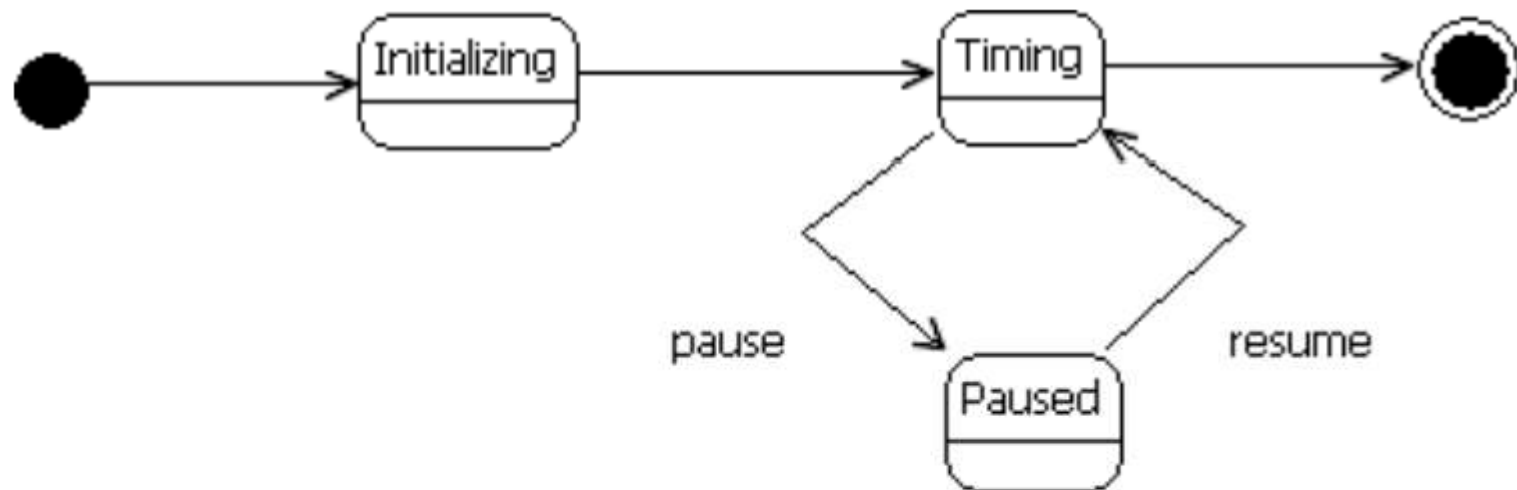
State Machine Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- **State Machine Diagrams**
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

State Machine Diagram

- A state machine diagram is used to design and understand time-critical systems
- A state machine diagram expresses behavior as a progression through a series of states, triggered by events, and the related actions that may occur
- These are also known as behavioral state machines

Example



States and Transition Events for the Duration Timer

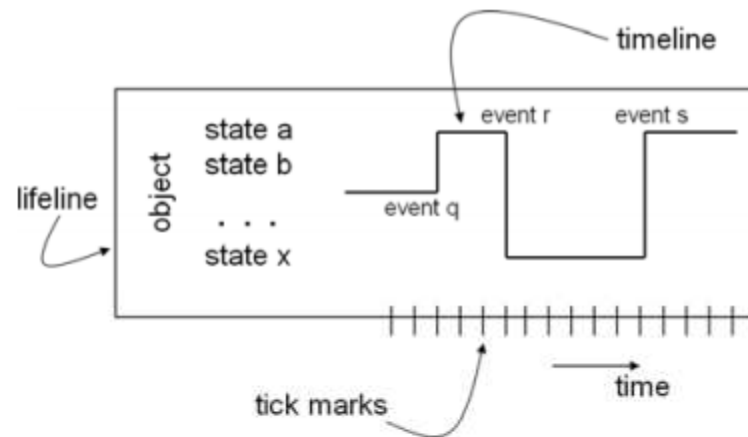
Timing Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- **Timing Diagrams**
- Object Diagrams
- Communication Diagrams

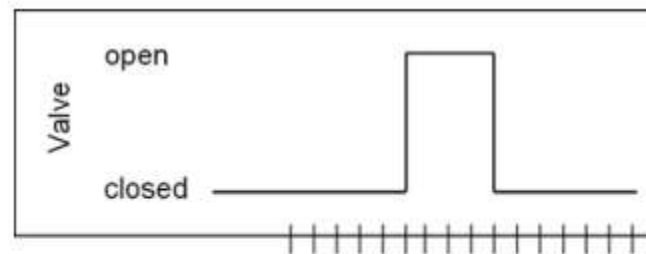
Timing Diagrams

- Timing diagrams are a type of interaction diagram
- Their purpose is to show how the states of an element or elements change over time and how events change those states

Example



A Generic Timing Diagram



A Timing Diagram for the Valve Object

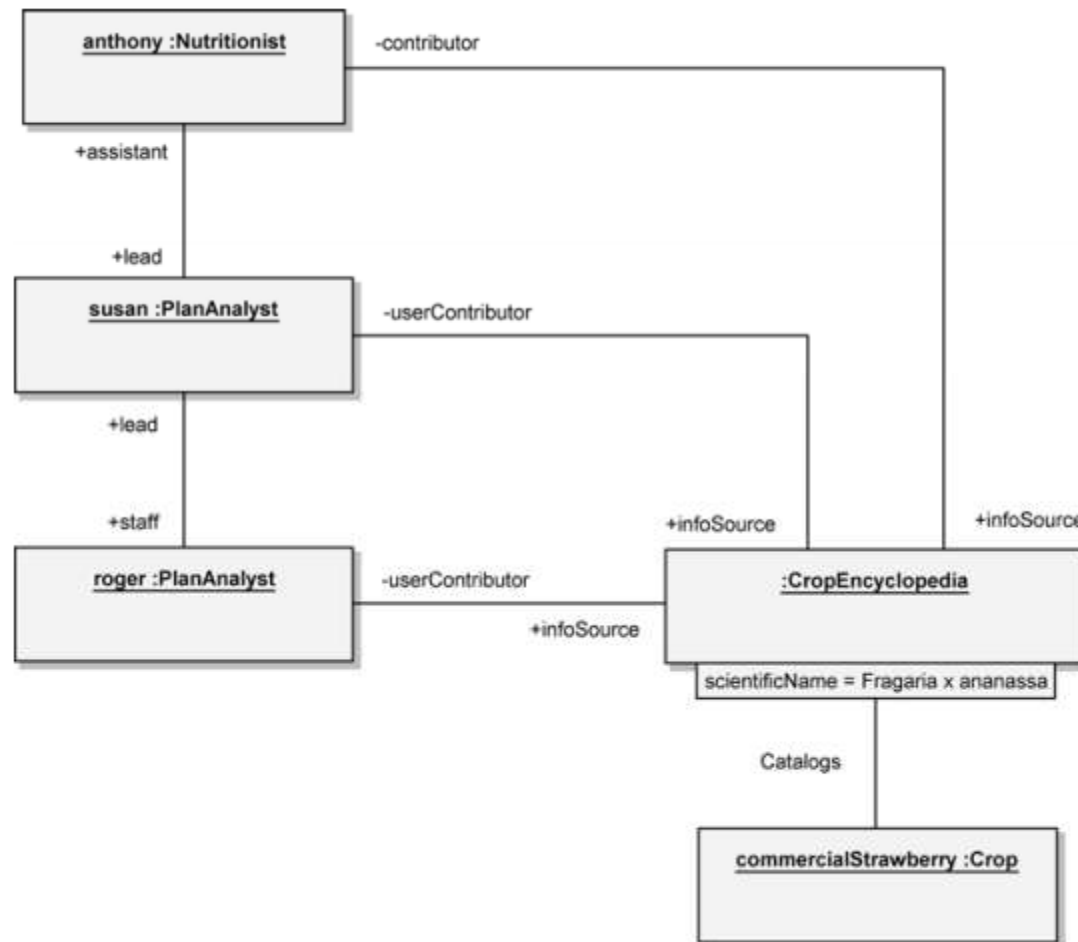
Object Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- **Object Diagrams**
- Communication Diagrams

Object Diagram

- An object diagram is used to show the existence of objects and their relationships in the logical design of a system
- Stated another way, an object diagram represents a snapshot in time of an otherwise transitory stream of events over a certain configuration of objects.

Example



Object Relationships

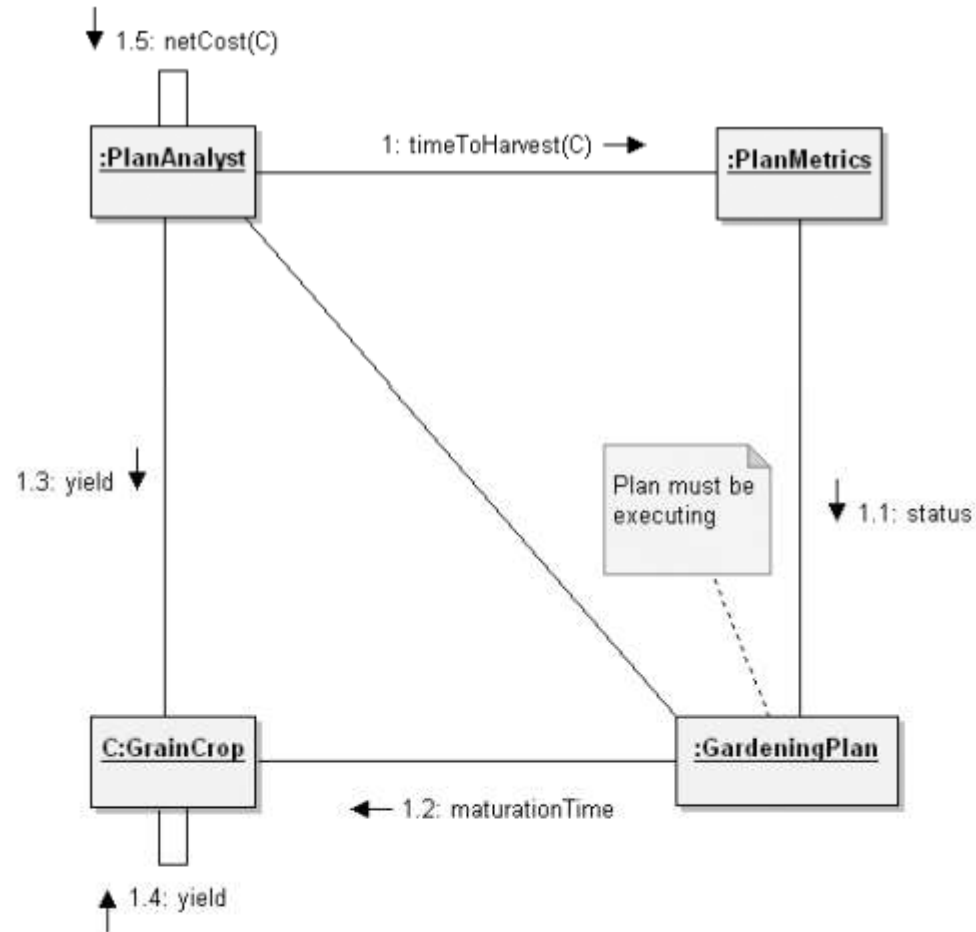
Communication Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- **Communication Diagrams**

Communication Diagram

- A communication diagram is a type of interaction diagram that focuses on how objects are linked and what messages they pass as they participate in a specific interaction

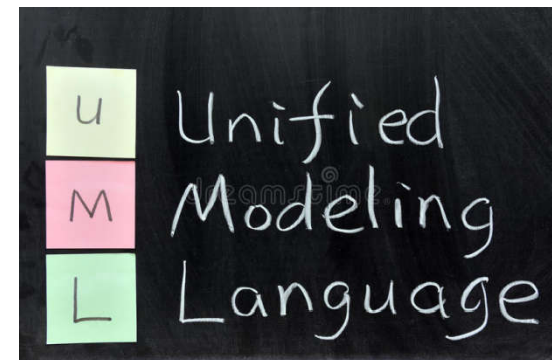
Example



Communication Diagram for the Hydroponics Gardening System

Summary

- Clear idea of Notation in Object Orientated Analysis and Design
- Visualising System
- UML Diagrams (Types)
- **UML Diagrams**



This Week

- Review Slides
- Read Chapter 6
- Online Quizzes
- Version Control (GitHub)

Questions/Discussion

■ Research Task

- ▷ What was the first object-oriented language?

Revision Question

- Name any three object oriented programming languages?

Answer

- Example, C++ , java , small talk and C# are most popular object oriented programming languages.

Question

- What do we mean by data hiding?

Answer

- Data hiding or **encapsulation**, is the mechanism in which implementation **details of a class are kept hidden** from the user (or external world)
- For example, data hiding concept is supported using the public, protected and private keywords which are placed in the declaration of the class

Question

- Briefly summarize the importance of using inheritance

Answer

- Inheritance is one of the most powerful features of object oriented programming. Most important advantages of inheritance are:
 - Reusability
 - Saves times and efforts
 - Closeness with the real world
 - Easy modification
 - Transitive Nature of inheritance

Question

- What do you mean by overloading of a function? When do you use this concept? Give an example of function overloading?

Answer

■ Function overloading is a technique where **several function declarations** are specified with a **same name** that can perform similar tasks, but on different data types (distinguished by their number and type of arguments)

■ Example

```
int add (int a, int b);  
int add (int a, int b, int c);  
float add (float a, float b);
```

Hence, overloaded functions perform different activities depending upon the kind of data sent to them

Question

- List the difference between Polymorphism and Overloading?

Answer

■ Polymorphism

Polymorphism is an important concept of OOPS.

Polymorphism means ability of one object to take many different forms.

Two main types of polymorphism:

Runtime polymorphism

Compile time
polymorphism

■ Overloading

Overloading is the mechanism to implement polymorphism.

Overloading is the mechanism to use the same thing for different purposes.