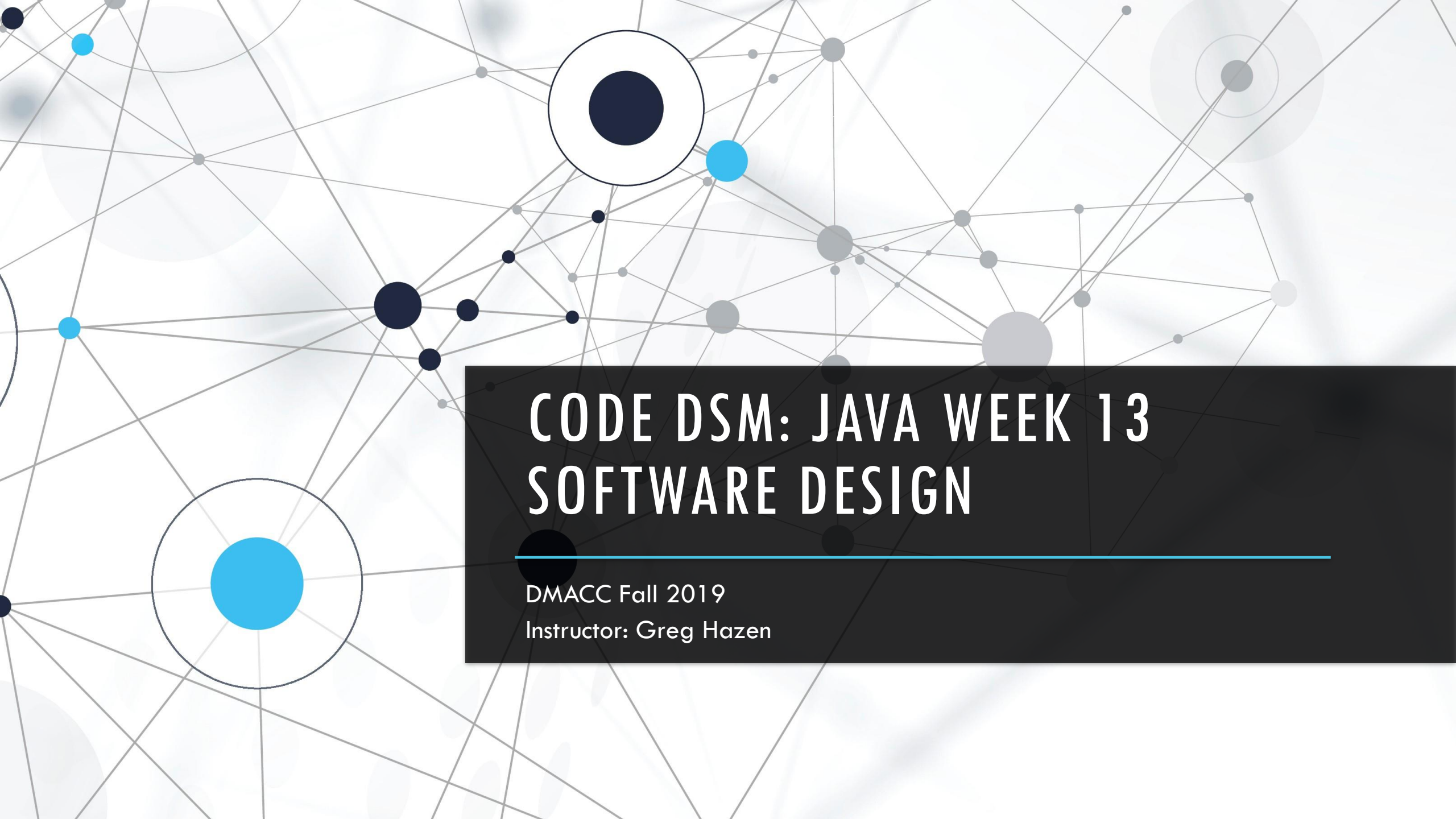


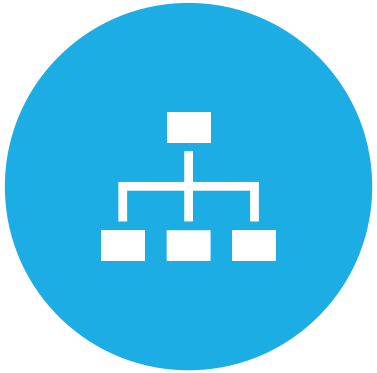
CODE DSM: JAVA WEEK 13 SOFTWARE DESIGN

DMACC Fall 2019

Instructor: Greg Hazen

AGENDA

STORIES



DIAGRAMS & PLANNING

DESIGN HANGMAN

STORIES

Capture the value to be delivered to the user

Putting a login button on a website isn't the value added

The value added is the ability to log into the website

STORIES

As a [role]

I need to [do some activity]

so that [I get some valuable outcome]

STORIES: EXAMPLE

As a customer
I need to order from a menu
so that I can get my food

STORIES: EXAMPLE

As a waiter
I need to get the menu from the restaurant
so that I can take orders from the customers

As a waiter
I need to total order prices
so that customers can pay

STORIES: EXAMPLE

As a manager
I need to add items to the menu
so that customers have items to order

As a manager
I need to update items in the menu
so that customers have accurate prices

STORIES

A tool to help us

- Discover the requirements
- Keep the focus on the value added

UML

Unified Modeling Language

A way to model the structure and behavior of code independent of any language

Serves as a blueprint for the code

If you jump straight into code, you're just guessing how the classes will fit together

UML

Many companies use UML for planning out large projects with many teams

Small companies and teams can benefit too, but in a less formal way (e.g. whiteboarding)

Diagrams also serve to present plans to the stakeholders

UML

Two types of diagrams

- Structural
- Behavioral

UML: STRUCTURAL

Represent the static parts of the code, like classes

Some of the diagrams are

- **Class Diagram**
- Package Diagram
- Deployment Diagram
- and others…

UML: BEHAVIORAL

Represent the dynamic parts of the code, like stored state, activities and interactions

Some of the diagrams are

- **Activity Diagram**
- **Use Case Diagram**
- **Sequence Diagram**
- and others…

UML: USE AS NEEDED

The purpose of the diagrams is to communicate about code

If it's easier to represent in code, don't waste time creating diagrams

UML: USE AS NEEDED

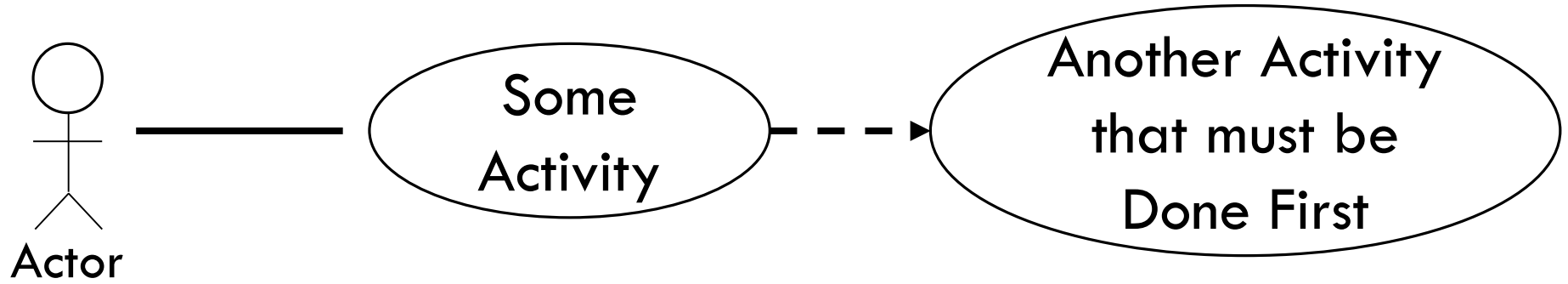
Once the design is implemented in code, don't keep the planning diagrams

By keeping logic in multiple places, all except the code WILL become outdated and MISLEADING

UML: USE CASE DIAGRAM

Visualizes activities and the role (or actor) doing them

Dashed arrows visualize that an activity depends on another activity to be done first



LET'S TRY IT ON THE WHITEBOARD

Using the Restaurant stories from before

UML: ACTIVITY DIAGRAM

Visualize the order in which the activities take place

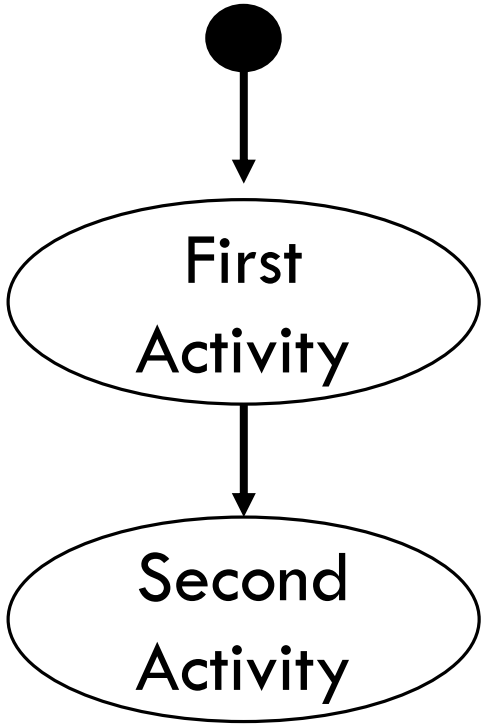
Good for listing out activities with the users

Has the activities, but not who's doing them

UML: ACTIVITY DIAGRAM

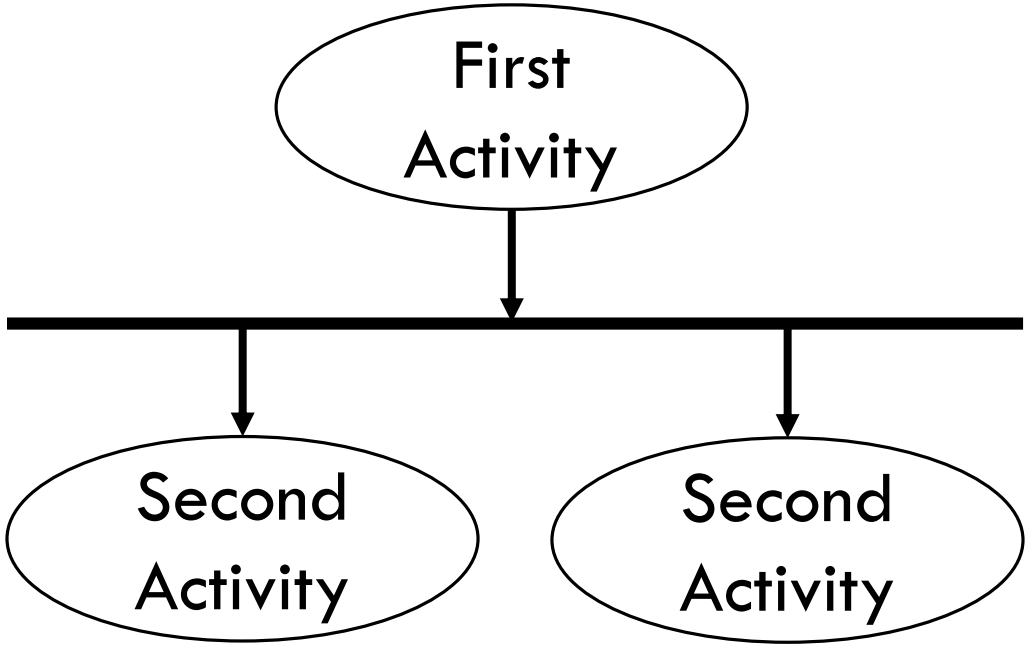
Starts with a black filled in circle

Arrow to the first and subsequent activities



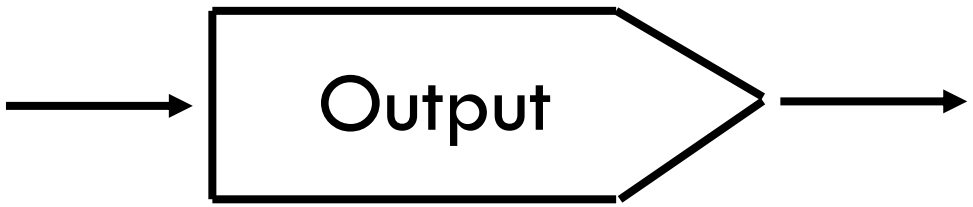
UML: ACTIVITY DIAGRAM

If the activity calls multiple activities next, use a fork

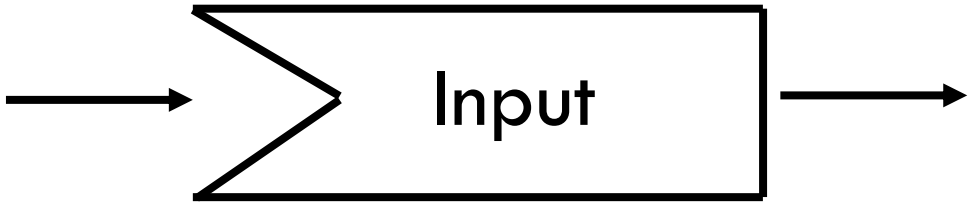


UML: ACTIVITY DIAGRAM

You can also represent outputs (e.g. save to file or asking user a question)



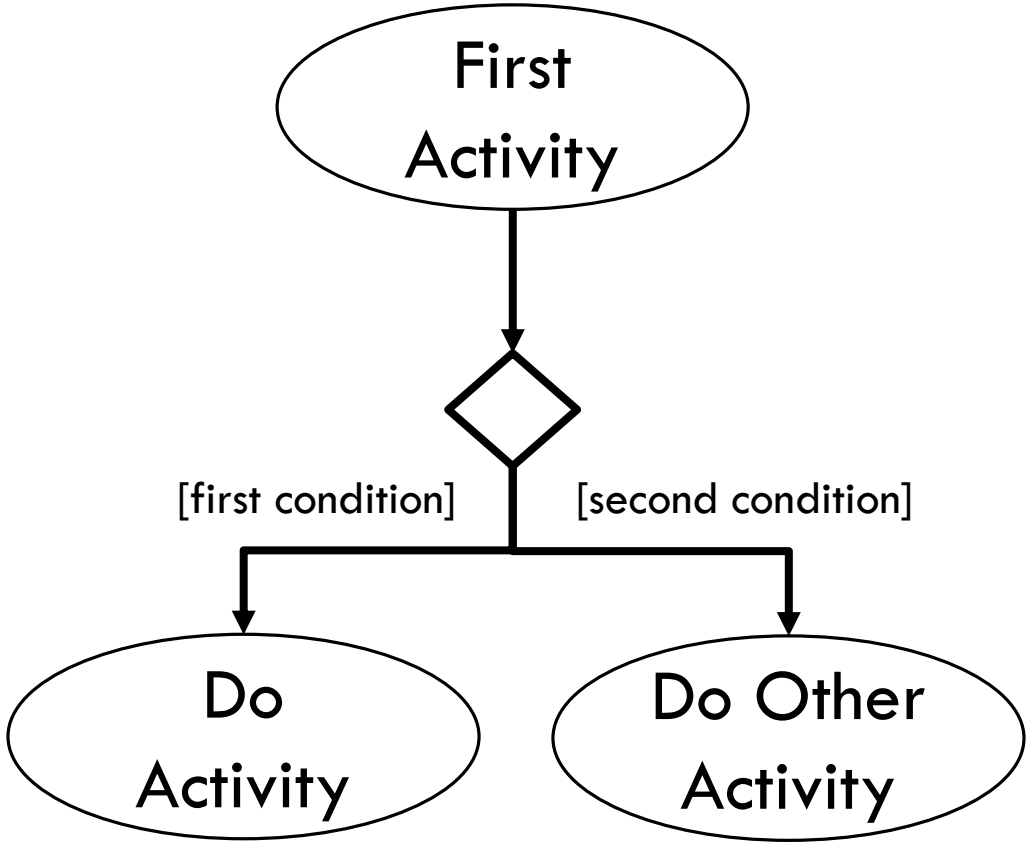
And inputs (e.g. reading a file or getting a response)



UML: ACTIVITY DIAGRAM

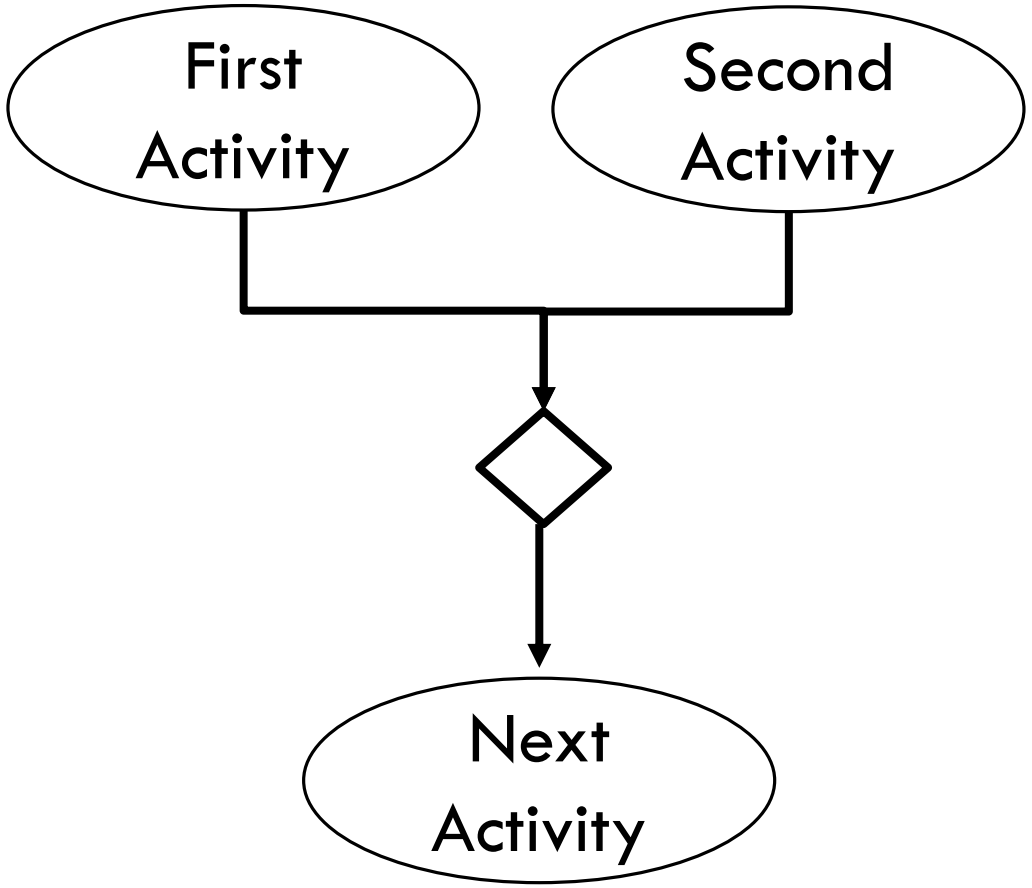
Use a split to represent a decision

You can use text in square brackets to label each decision



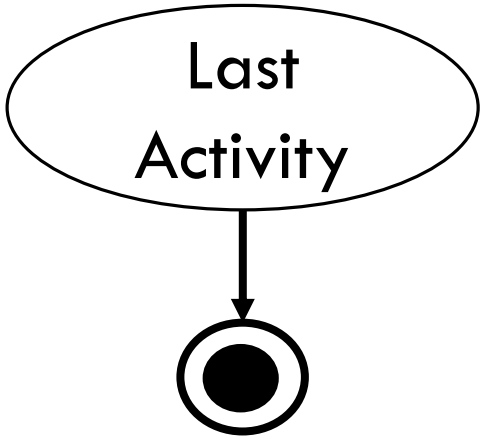
UML: ACTIVITY DIAGRAM

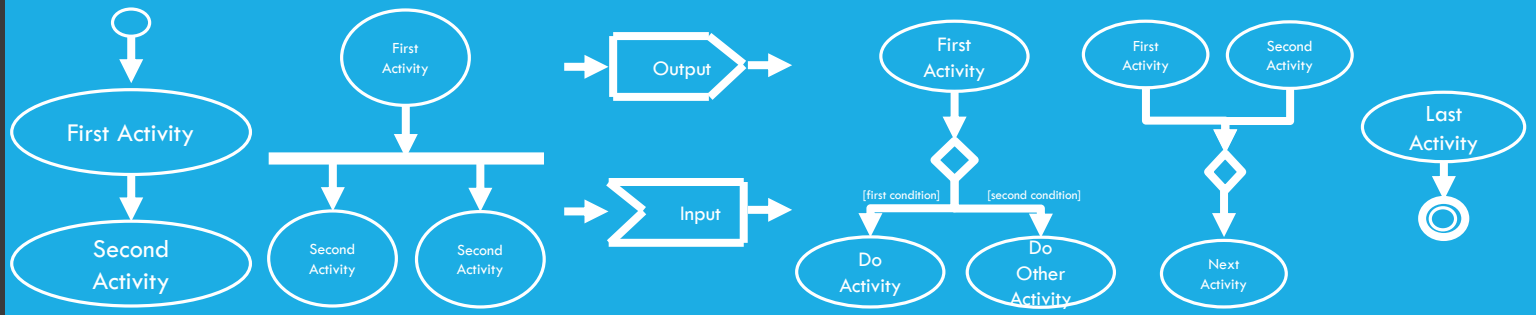
A join can bring multiple activities back together



UML: ACTIVITY DIAGRAM

Finally, terminate the last activity





LET'S TRY IT ON THE WHITEBOARD

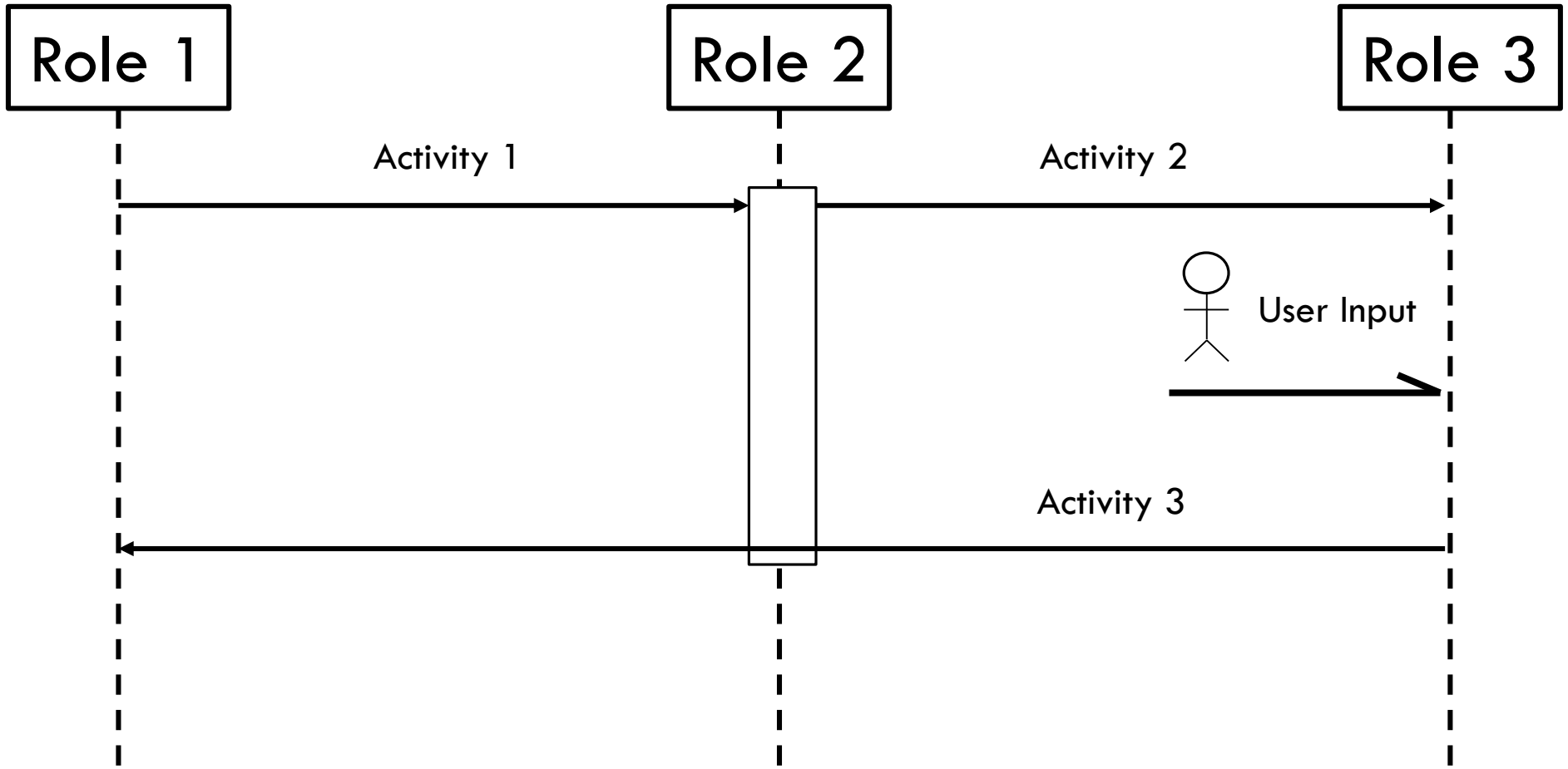
Using the Restaurant stories from before

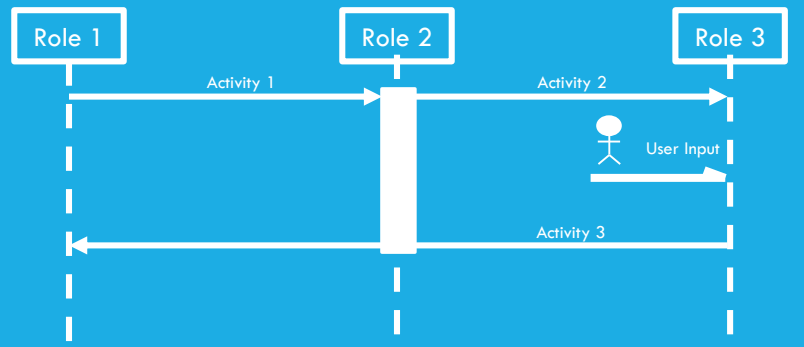
UML: SEQUENCE DIAGRAM

Similar to Activity Diagrams, but introduces who is doing the activities

Vertical dashed lines labeled with role names separate the activities organized from top to bottom

UML: SEQUENCE DIAGRAM





LET'S TRY IT ON THE WHITEBOARD

Using the Restaurant stories from before

UML: CLASS DIAGRAM

Finally, we're to the point of thinking in terms of classes

If you jump straight to class design, you may be missing requirements or use case scenarios you hadn't considered

UML: CLASS DIAGRAM

Good at representing Object Oriented concepts

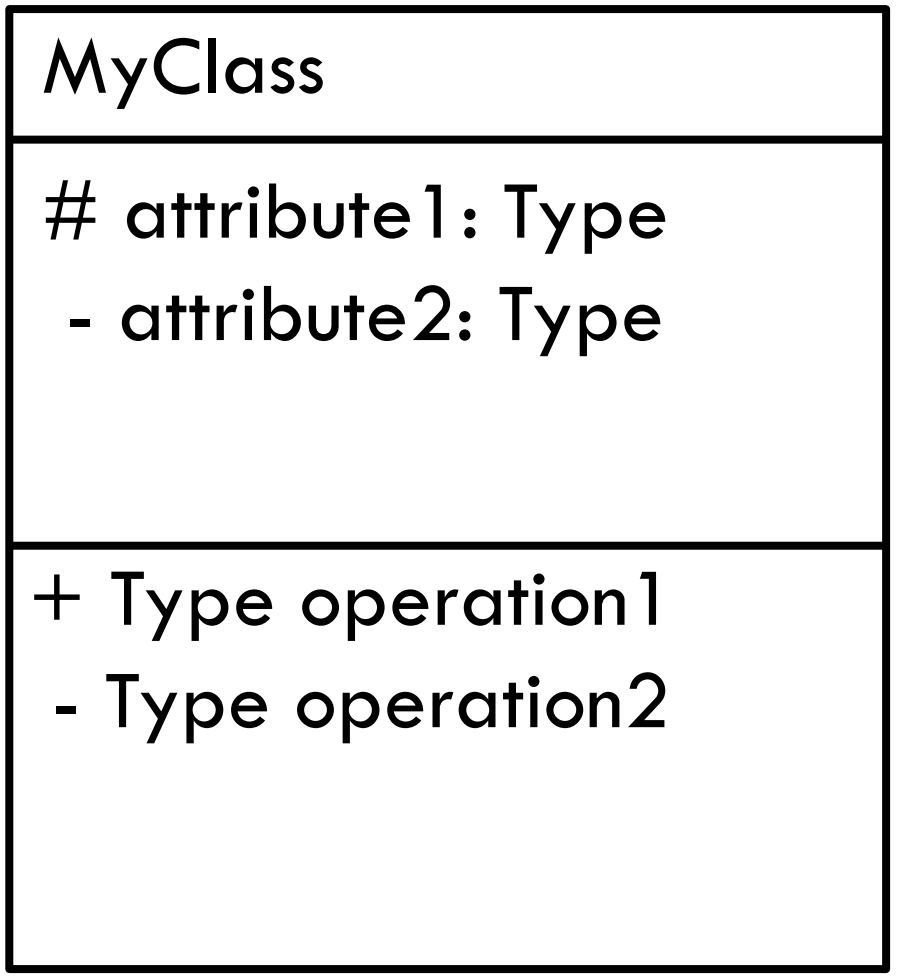
Remember Classes, Objects, Inheritance, Abstraction, Encapsulation and Polymorphism?

UML: CLASS DIAGRAM

Classes are represented by boxes with three parts

1. Class Name
2. Attributes/Variables
3. Operations/Methods

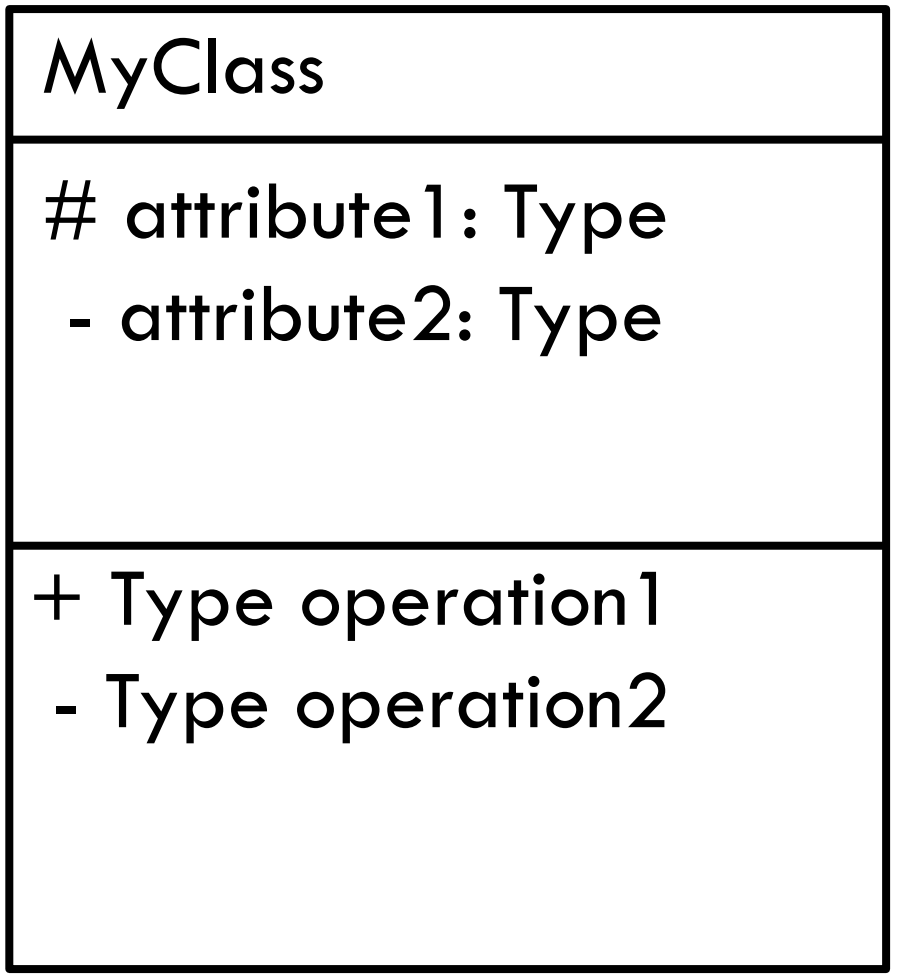
The “Type” is the object type (e.g. String, int…)



UML: CLASS DIAGRAM

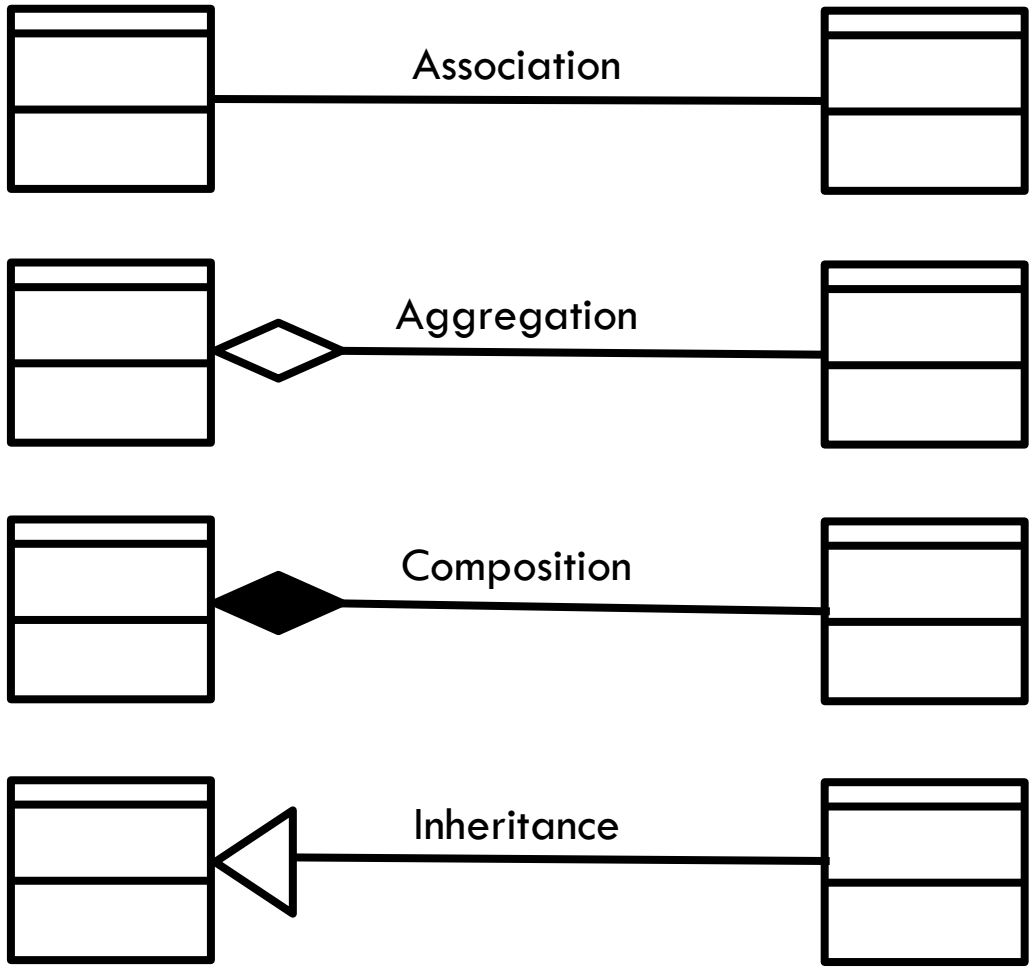
Attributes and Operations can be

+ Public
- Private
Protected



UML: CLASS DIAGRAM

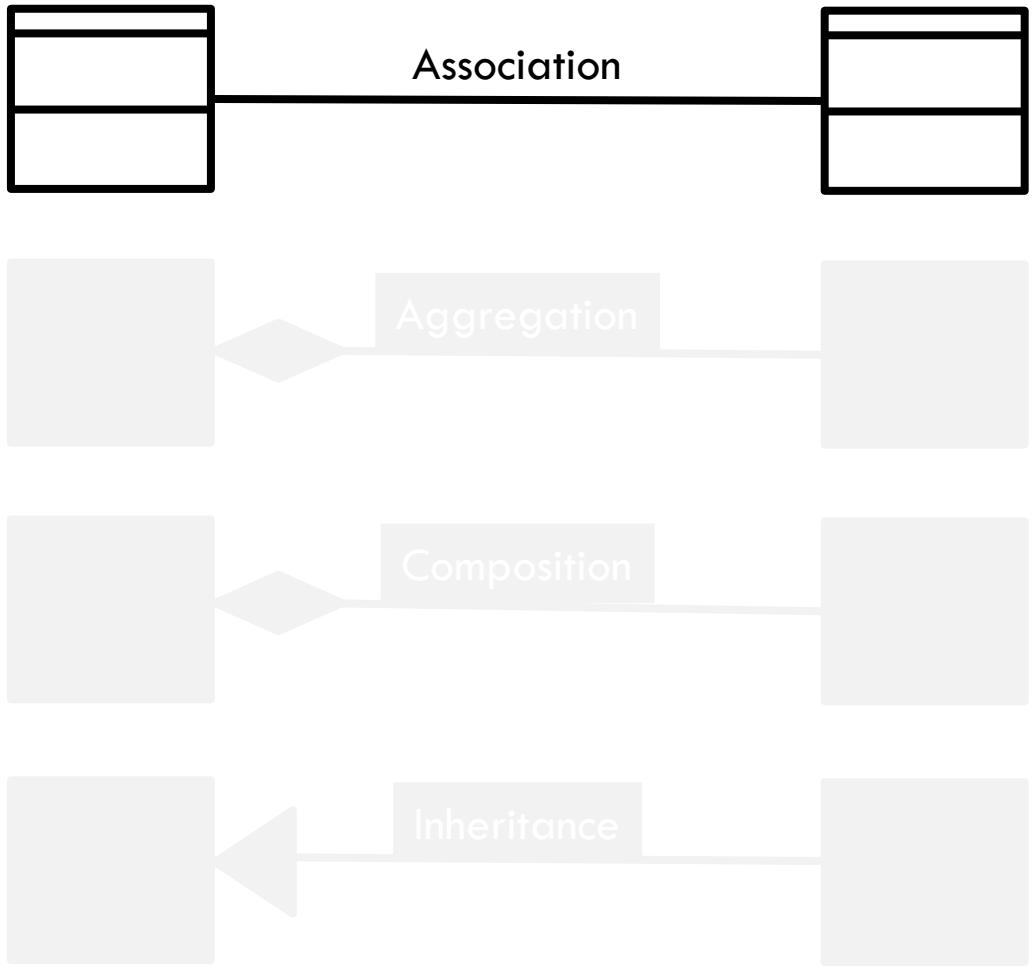
The relation between two classes is represented by different types of lines or arrows



UML: CLASS DIAGRAM

Association

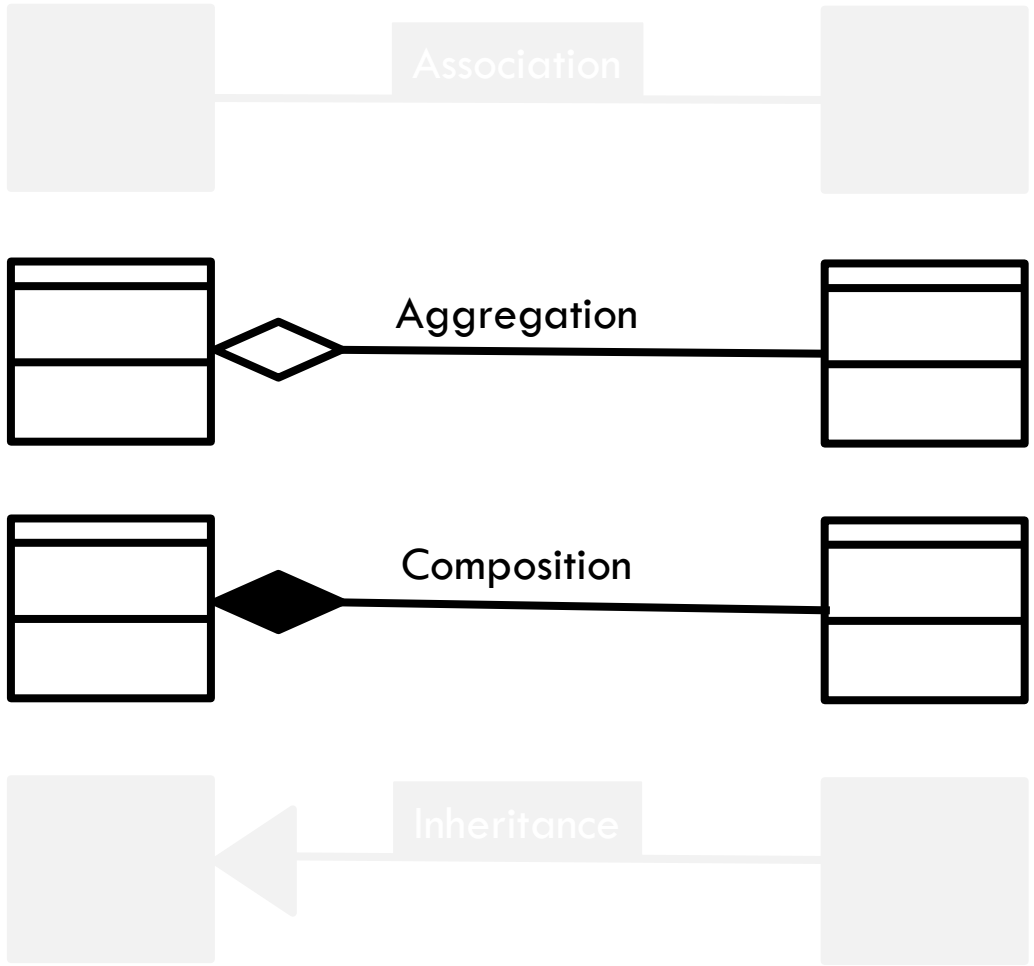
Two classes interact, but don't store each other



UML: CLASS DIAGRAM

Aggregation and Composition

The class on the right is part of the class on the left

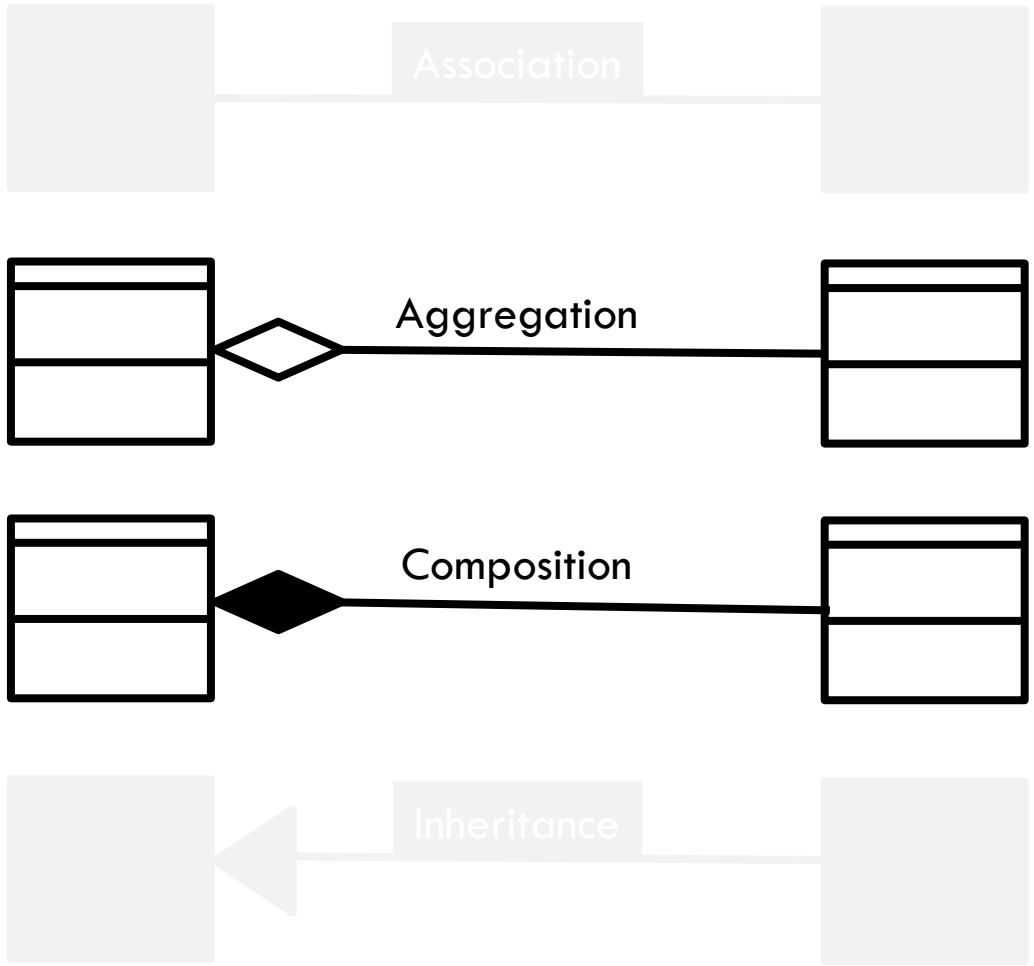


UML: CLASS DIAGRAM

Aggregation

The class on the right **can** exist independently of the class on the left

Like a student can exist without their parent present

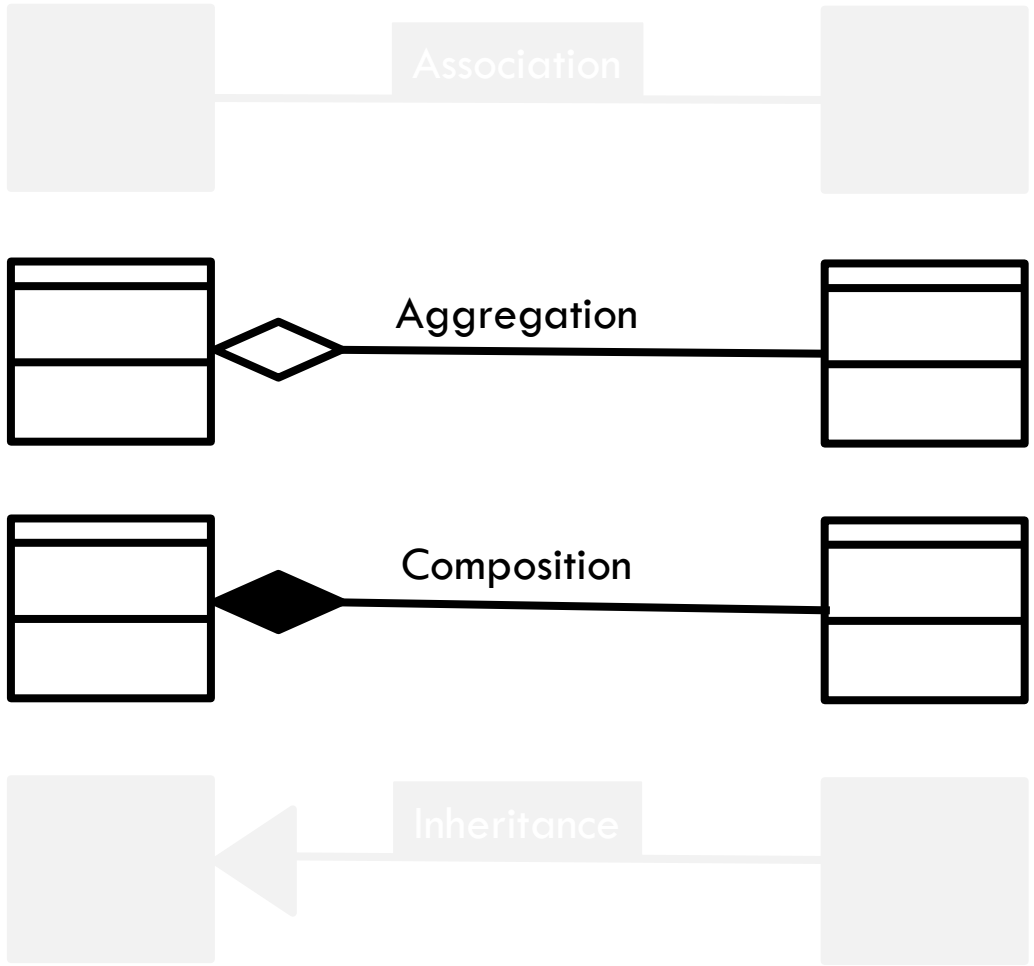


UML: CLASS DIAGRAM

Composition

The class on the right **cannot** exist independently of the class on the left

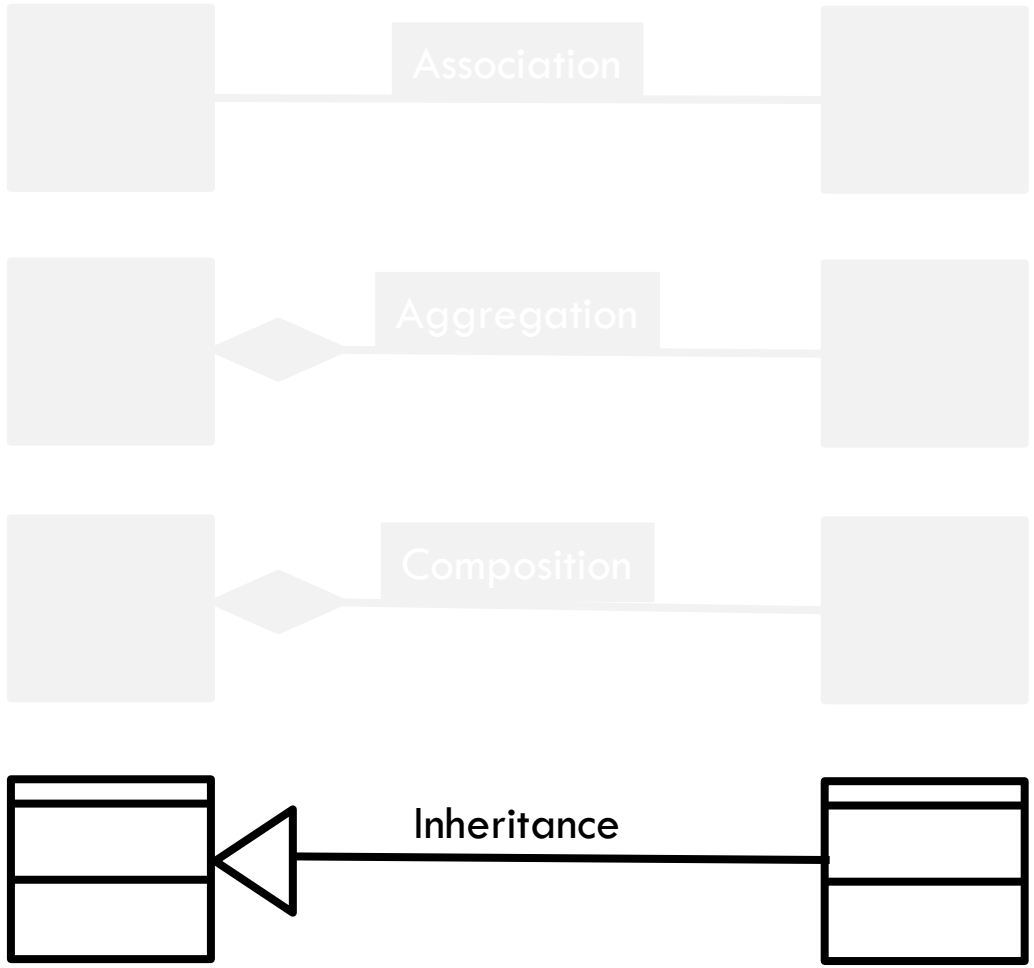
Like a room in a house cannot exist without the house



UML: CLASS DIAGRAM

Inheritance

The class on the right inherits from the class on the left

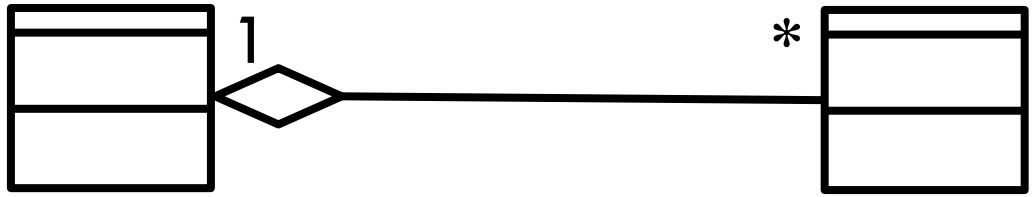


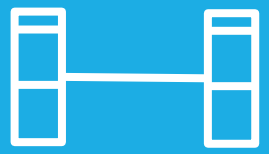
UML: CLASS DIAGRAM

Each arrow/line can represent one or more of the classes

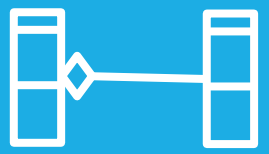
1	only one
*	zero or more
1..3	one to three
2..*	two or more

Example, one class has zero or more students

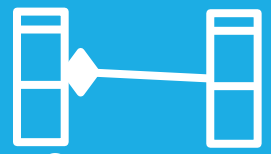




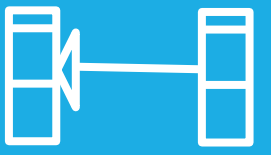
Association



Aggregation



Composition



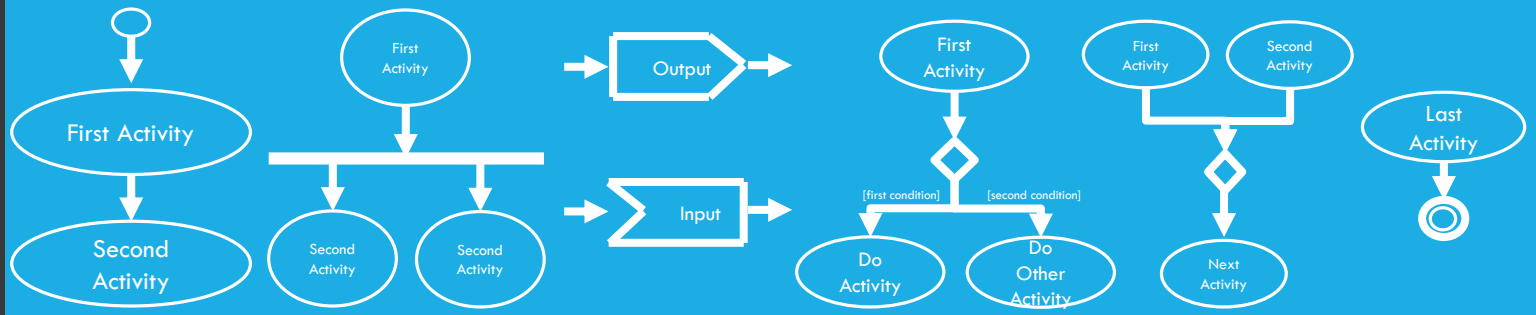
Inheritance

LET'S TRY IT ON THE WHITEBOARD

Using the Restaurant stories from before

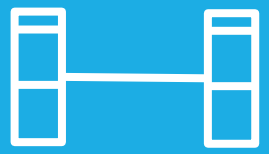
TOGETHER ON THE WHITEBOARD

Let's create a Use Case Diagram for Hangman

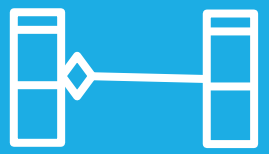


TOGETHER ON THE WHITEBOARD

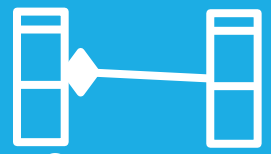
Let's create an Activity Diagram for Hangman



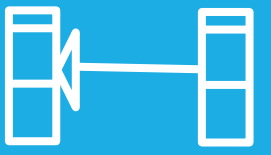
Association



Aggregation



Composition



Inheritance

EXERCISE

Create a Class Diagram of YOUR plan for Hangman

FINAL PROJECT

In-class time now to work on your final project

I'm available for questions or design help

ASSIGNMENT

No Homework or quiz!

SEE YOU JANUARY 17TH!

Don’t forget to ask question early in the week

REFERENCES

- Visual-Paradigm.com. (2019). What is Activity Diagram? [online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ [Accessed 7 Jan. 2020]
- Visual-Paradigm.com. (2019). UML Aggregation vs Composition. [online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/ [Accessed 7 Jan. 2020]
- Pluralsight.com. (17 Mar 2016). Picturing Architecture: UML (The Good Bits) and More. [online] Available at https://app.pluralsight.com/library/courses/picturing-architecture-uml [Accessed 1 Jan. 2020]