# CS225L Lab 6: UML Diagrams

# Learning Outcomes

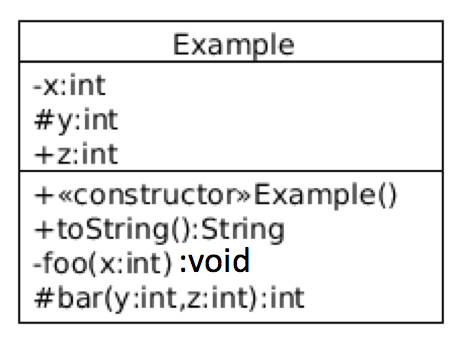
* What is UML
* How to create a UML diagram

# Pre-lab: Intro to UML Diagrams

**Overview**

Unified Modeling Language (UML) is a way of visualizing a software program using a collection of diagrams. The notation has evolved to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

(ref: https://www.smartdraw.com/uml-diagram/)

**A Single Class Example**

Say we have the below class:

public class Example {

private int x;

protected int y;

public int z;

public Example() { ... }

public String toString() { ... }

private void foo(int x) { ... }

protected int bar(int y, int z) { ... }

}

*Rules:*

|  |  |
| --- | --- |
| private | - |
| public | + |
| protected | # |
| package-private (default) | ~ |
| static | \_\_\_\_\_\_ (underline) |

fields: <visibility> <field name> <data type>

methods: <visibility> <method name> <parameter type> <return type>

Example: public static void main(String[] args) ---> + main(String[]): void

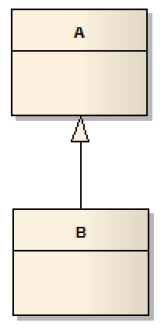
**Class Relationships**

There are four basic categories of class relationships:

1. Inheritance (Generalization / Realization)
2. Dependency
3. Association
4. Aggregation / Composition

*Inheritance - Generalization:*

public class B extends A { …… } public interface B extends A, C { …… }

*Inheritance - Realization:*

public class B implements A, C { …… }



*Dependency:*

The Dependency relationship is a simple one used to denote that a class relies (temporarily) on another class in its implementation.



Here the computeInterest() method in BankAccount class need to use the Math.pow()method. That’s dependency between BankAccount class and Math class.

*Association:*

Association denotes a general relationship between classes which may change between consecutive executions of an application.

Example 1:

public class House {

// attributes (such as address) and code for class House go here...

}

public class Person {

private House home; // Person includes a reference to an instance of House

}

// other attributes (such as name) and code for class Person go here, which may make use of the 'home' reference }



Example 2:

public class House {

// attributes (such as address) and code for class House go here...

}

public class Person {

private List<House> homes; // Person includes a reference to an collection of Houses }

}



*Aggregation:*

Aggregation relationships are used to indicate that instances of one class *contain* or *are composed of* instances of other classes.

public class Course {

private List<Student> students; // reference to a collection of Students,

private Professor instructor; // a reference to an single instance of a Professor,

private Schedule sched; // and of a single Schedule

// other attributes and code for the Course class go here...

}



If the diamond is left empty, it signifies it is an aggregation. This relation is stronger than a simple association. In this case a Course aggregates a Professor and a list of Students.

If the diamond is black, this means it is a composition, which is even stronger than an aggregation because the aggregated class cannot be aggregated by other classes. Its "life" depends on the container. (A schedule itself without a course does not make any sense while a professor or a student without a course still makes sense)

(ref: https://stackoverflow.com/questions/15141678/what-does-a-diamond-sign-signify-in-uml-class-diagrams)

Hints about Association and Aggregation/Composition:

Association: A person ***can*** live in / build / own / rent / destroy a house.

Aggregation/Composition: A course ***must*** have an instructor, several enrolled students and a schedule.

**Reading Materials**

1. From Ball State University: <http://www.cs.bsu.edu/homepages/pvg/misc/uml/>
2. From tutorialspoint: <http://www.tutorialspoint.com/uml/>

**Lab Activities**

For this lab, pair up in groups of no more than 4 people or work solo. Then select one of the following games:

* Chess
* Monopoly
* Risk
* The Settlers of Catan

1. **If in a group otherwise skip this:** write down every group member’s full name. Everyone in the group needs to submit the same diagram.
2. For the selected game, you are to develop at least 10 requirements needed to produce a software application to play the game.
3. Once the requirements are done, next develop the UML Diagram to represent the software application of the game you selected.