Up until now ...

- **Software Process**
  - Waterfall, Spiral, Iterative, XP Programming
  - Productivity

- **Unit Testing**
  - Test for success, for failure and sanity
  - Glass box vs Black Box

- **What does it mean to be O.O.**
  - Encapsulated, State Retention, Implementation / Information Hiding, Object Identity, Messages, Classes, Inheritance, Polymorphism, Generacity
1) What are my office hours?
2) What are some of the activities in the software process? (name as many as you remember)
3) What are some of the characteristics of XP Programming?
4) What is the different between validation and verification?
5) In Unit testing, what is a Unit?
6) What kind of testing is done using only specifications?
7) When observing an encapsulation, what are the two possible point of view?
8) Why is it a bad idea to make attributes public?
9) How can two objects with the same state be differentiated?
10) If I want class B to be a subclass of class A, what characteristic must B have?
Company XYZ is a manufacturing company that produces cartoon action figures for big entertainment companies.

This company needs an inventory and tracking system.

The inventory system keeps track of how many of each figurines is stored in each warehouse.

Figurines are stored in cases.

Clients order the figurines and the cases are eventually shipped to clients.

Build a quick design for this. How many classes do you need?
Company XYZ is a manufacturing company that produces cartoon figurines for big entertainment companies.

This company needs an inventory and tracking system.

The inventory system keeps track of how many of each figurines is stored in each warehouse.

Figurines are stored in cases.

Clients order the figurines and the cases are eventually shipped to clients.
Company XYZ is a manufacturing company that produces cartoon **figurines** for big entertainment companies.

This company needs an **inventory** and **tracking** system.

The inventory system keeps track of how many of each figurines is stored in each warehouse.

Figurines are stored in cases.

**Clients** order the figurines and the cases are eventually shipped to clients.
This time, in UML
Unified Modeling Language (UML)

- A language, both graphical and textual, used throughout the entire process of project design (from requirements analysis to deployment).
- Semi-formal specification that captures structure of O.O.D.
- A standard tool for communicating a design.
UML is a language

Syntax
  Text
  Visual

Semantic
What is it not?

■ programming language, but it can be used to generate code in some programming language

■ development process, but it can be used as part of a development process
Unified Modelling Language (UML)

many notations

COLLECTION

OCL

Constraints

use case diagrams

use cases

structural

class diagrams

package diagrams

(object) interaction

sequence diagrams

collaboration diagrams

behaviour

Activity diagrams

state charts

implementation

deployment
We will focus on three branches

- structural (i.e. class diagrams)
- interaction (i.e. sequence diagrams)
- behaviour (i.e. activity diagrams)
Mid- to late- 1990's

“3 AMIGOS” (Grady Booch, Jim Rumbaugh and Ivar Jacobson)

Joined together to unify their individual ventures

Each was working on some sort of O.O. modeling technique
Merging Techniques

UML (Booch, Rumbaugh, Jacobson)

Use-Case (Jacobson)

OOA (Coad)

OOA - OODLE (Schlaer & Mellor)

Class-Relationship (Desfray)

OOA - OOD (Booch)

OMT (Rumbaugh)

Data-Flow (De Marco)

State-Transition Diagrams (Harel)

Entity-Relationship (Chen)
- Practice of writing compound words or phrases where the words are joined without spaces, and each word is capitalized within the compound.
  - CsGames, LinkedList, eBay, PowerBook, WestJet
- Used by programmers and marketing.
- The terms UpperCamelCase and lowerCamelCase are used to distinguish two types of CamelCase.
  - UpperCamelCase
  - lowerCamelCase
Class Diagram

- Classes consist of
  - the class name ➔ written in BOLD
  - it's features ➔ attributes and methods
  - user-defined constraints

- Note that class diagrams contain only classes, not objects.
Here is a concrete example of a class called Point, which depicts a 2D point.

There are no constraints (yet...)

A class name is written in UpperCamelCase

<table>
<thead>
<tr>
<th>2DPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>x:int</td>
</tr>
<tr>
<td>y:int</td>
</tr>
<tr>
<td>getx():int {return x}</td>
</tr>
<tr>
<td>setx(a:int):void {x = a}</td>
</tr>
<tr>
<td>gety():int {return y}</td>
</tr>
</tbody>
</table>