# 1.204 Lecture 1

## Course introduction
## Data models

## Announcements

- How-to install documents on Web:
  - Java, Eclipse, submit problem sets (1.00 Web site)
  - SQL Server, Visual Paradigm, JDBC (1.204 Web site)
- We will give you access to 1.00 Web site
  - Sign up as 1.00/1.001 listener if you plan to sit in on 1.001
- Software installation help
  - Email me with questions or to set up a time for help
- MySQL: Mac users can use MySQL to avoid Boot Camp, etc.
  - No support for MySQL installation problems
- Homework 0: Software installation, due Mon Feb 8. Ungraded.
- Homework 1: Air schedule, due Tue Feb 16. Graded.
  - Initial modeling/coding exercise, using straightforward solution method
  - Homework 1 only: You may code in a language other than Java
- Lecture notes: Printed notes handed out each lecture
- Readings: On reserve at Barker, and on electronic reserve

## Course outline

- **Staff:**
  - **George Kocur -**
- **Class Monday, Wednesday 1-2:30pm**
- **Office hours**
  - **George Kocur MW4:45-5:45**
- **Prerequisite: 1.00/1.001/6.005.**
  - **Can listen to 1.00 this semester**
- **Grading:**
  - **7 homework sets (70%)**
  - **2 quizzes (30%)**
- **Bring your laptop to office hours for help**
  - **Not needed in class, though you're welcome to bring it**

## Topics

- **Databases**
  - **Data modeling, normalization**
  - **SQL, JDBC**
- **Data structures**
  - **Stacks, queues, trees/dictionaries, heaps, sets, graphs**
- **Divide and conquer, greedy models**
  - **Sorting, selection**
  - **Knapsack, job scheduling, spanning trees, shortest paths**
- **Dynamic programming**
  - **Resource allocation, job scheduling, knapsack**
- **Branch and bound**
  - **Knapsack, facility location**
- **Linear/nonlinear systems, linear programming**
  - **Nonlinear optimization, constrained and unconstrained**
  - **Network equilibrium (convex combinations), choice estimation**
  - **Solution of linear systems, linear programming**
- **Approximate queuing theory**
  - **Time-varying queues, deterministic queues, graphical methods**

## Homework

- **Homework topics**
  1. **Informal algorithm design (warm-up homework)**
  2. **Database**
  3. **Network data structure**
  4. **Greedy algorithm**
  5. **Network algorithm**
  6. **Dynamic programming**
  7. **Branch and bound**
  8. **Nonlinear optimization**
- **Work individually**
  - **You may discuss approach to homework with others**
  - **You must write your own Java and SQL code**
  - **Please read Academic Honesty Guidelines in FAQ**
- **Create one Eclipse project for the whole term**
  - **Create src.xxx packages for each lecture, homework**

## Readings, computer systems

- **Readings (some on electronic reserve) are sections from these books:**
  - **Computer Algorithms in C++ (2nd ed), Horowitz, Sahni, Rajasekaran**
  - ***Introduction to Algorithms, Cormen, Leiserson, Rivest, Stein***
  - ***Big Java, Horstmann***
  - **Murach's SQL Server 2008 for Developers, Syverson, Murach**
  - **JDBC API Tutorial and Reference, Fisher, Ellis, Bruce**
  - **Applications of Queuing Theory, Newell**
  - **Linear Programming, Chvatal**
  - **Numerical Recipes, Press et al (2nd or 3rd edition)**
  - **Urban Transportation Networks, Sheffi**
- **Use your own laptop or desktop computer**
  - **All software available for download**
  - **Either open source or free**
  - **TA will help with installation and initial usage**
- **1.204 site on Stellar (stellar.mit.edu)**
  - **Lecture notes**
  - **Homework**
  - **Electronic reserve, online readings**
  - **Announcements**

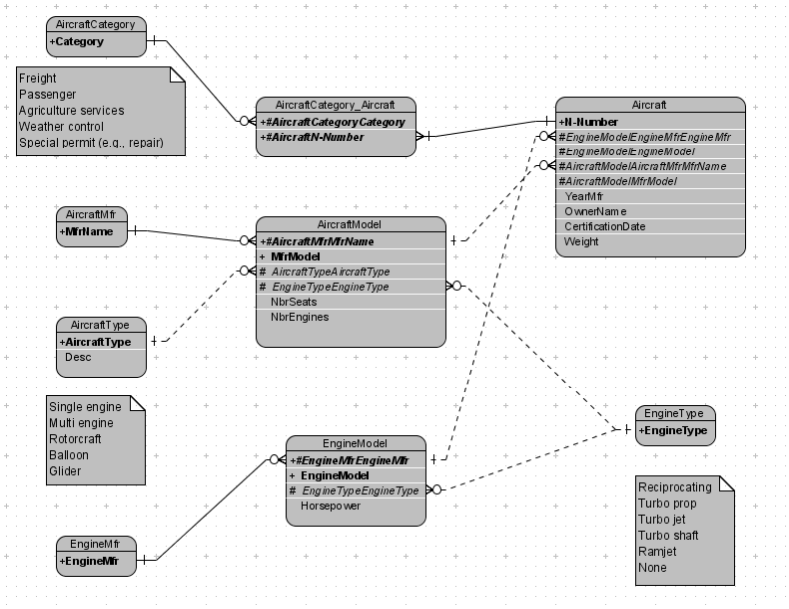
## Data models

- **Data model is representation of**
  - **Things (or entities or classes) of importance to a system**
  - **How the things relate to each other**
- **It is built and modified until it represents the system well enough to support a system model**
- **Data models are extended to become class diagrams in the Unified Modeling Language [UML] by adding the behaviors of each entity to the model**

## Logical data modeling

- **Method to discover the data, relationships and rules of a system, collectively called the system rules**
- **Logical data models are the basis of:**
  - **Physical data models, or actual databases**
  - **Applications, parts of which can be automatically generated from the data model**
- **Small model for aircraft**
  - **Says a lot about system structure**
  - **Gives good picture of what database should look like**
  - **Also gives good picture of underlying system rules**

## Aircraft Data Model




## Aircraft System Rules

- **An aircraft can be in many categories**
- **A category can be associated with many aircraft**
- **An aircraft model is built by one aircraft manufacturer**
- **An aircraft manufacturer builds many aircraft models**
- **An aircraft model is of one type**
- **An aircraft type can be associated with many aircraft models**
- **An engine type can be represented by many engine models**
- **Each engine model is of one engine type**
- **An aircraft model has one engine type**
- **An engine type may be in many aircraft types**
- **An aircraft has one engine model (it may have >1 engine)**
- **An engine model may be in many aircraft**
- **An engine manufacturer builds many engine models**
- **An engine model is built by one engine manufacturer**

## Data model purpose

- **Engineer needs to build logical data model so users and engineers both understand system rules**
  - **Models enable users and developers to have single view of system**
  - **Sometimes users note this is first time they understood system rules!**
- **Converting logical to physical data model (database) is very straightforward these days.**
  - **Little need for separate physical model for online databases**
  - **Create integer system-generated keys instead of strings and composite keys for performance**
  - **We still create separate physical models for data warehouses, read-only databases and some other special cases**
- **Model also serves as basis of class diagram for code**

## Data modeling concepts

- **Entities (classes, tables)**
- **Attributes (properties)**
- **Relationships**
- **Keys (primary and foreign)**
- **Referential integrity**

## Entity type and entity occurrence

**Entity type**

**Department**

**DeptNbr**
**DeptName**
**DeptType**
**DeptStatus**

Table, class

**Entity occurrence**

| Department | | | |
|---|---|---|---|
| DeptNbr | DeptName | DeptType | DeptStatus |
| 930 | Receiving | Mfg | Active |
| 378 | Assembly | Mfg | Active |
| 372 | Finance | Adm | Active |
| 923 | Planning | Adm | Active |
| 483 | Construction | Plant | Inactive |

Row, object

## Entities

- **"Department" is an entity type**
  - **In Java, "department" is a class**
- **"Department 101" is a row, or an occurrence of entity type "Department"**
  - **In Java, "department 101" is an object, which is an instance of class "department"**
- **Entities are things, often physical, that have facts associated with them.**
- **Processes are almost never entities. For example:**
  - **Aircraft certification is not an entity**
  - **Aircraft purchase is not an entity**
  - **Reports are not entities**

## Attributes

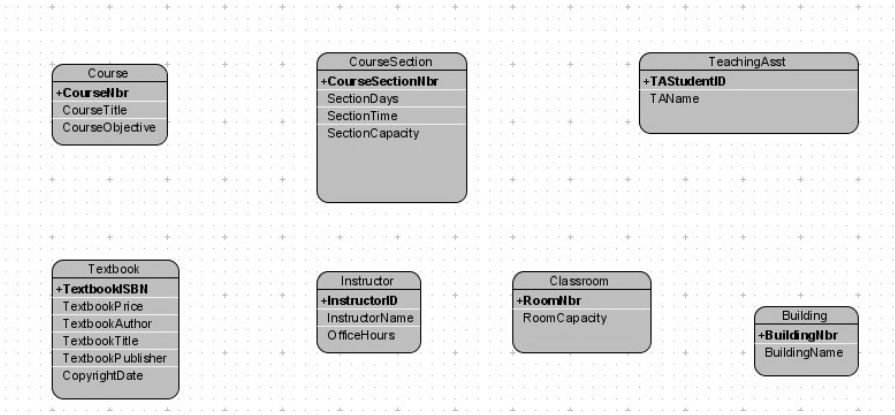
- **Attributes are a data item or property associated with an entity type**
  - **They are typically nouns (quantity, type, color, …)**
  - **Example: Employee**
    - **ID**
    - **Name**
    - **Social security number**
    - **Address**
    - **Phone**

## Entity type/attribute example

**Identify which are types and which are attributes:**

- **Instructor**
- **Teaching assistant (TA)**
- **Course section number**
- **Building name**
- **Course number**
- **Textbook price**
- **TA name**
- **Instructor ID**
- **Textbook author**
- **Course title**
- **Textbook**
- **Classroom**
- **Textbook ISBN**
- **Section days**
- **Section time**
- **Instructor office hours**
- **Textbook title**
- **Classroom number**
- **TA student ID**
- **Instructor name**
- **Textbook publisher**
- **Section capacity**
- **Course objective**
- **Copyright date**
- **Building number**
- **Course section**
- **Course**
- **Building**
- **Classroom capacity**

# Example




# Domain entity type

- **Also called pick list, validation list, etc.**
- **Department name example**

**Domain entity type**

| Department | | | |
|---|---|---|---|
| **DeptNbr** | **DeptName** | **DeptType** | **DeptStatus** |
| 930 | Receiving | Mfg | Active |
| 378 | Assembly | Mfg | Active |
| 372 | Finance | Adm | Active |
| 923 | Planning | Adm | Active |
| 483 | Constructi | Plant | Inactive |

| ValidDeptType |
|---|
| **DeptType** |
| Mfg |
| Adm |
| Plant |
| Sales |
| Operations |

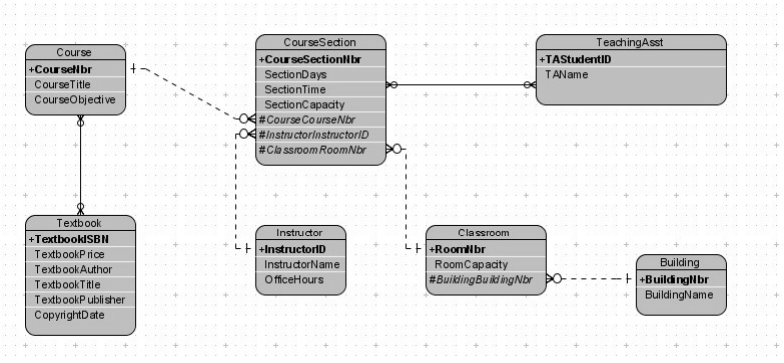
# Relationships

- **Entities are drawn as boxes, as in the broker diagram**
- **Relationships are lines between boxes**
- **Cardinality is the expected number of related occurrences between the two entities in the relationship**
- **Relationships + cardinality = system rules**


**One** **Zero or many**

**(Instructor)** **(Course section)**

# Example




**We're getting there: we've defined entities, attributes and relationships. We still have to add keys and more entities**

## Course example

- Course may be offered in many (0,1 or more) sections
- Course section must be associated with a course
- Course section may be taught by many (0,1 or more) TAs
- TA may teach many (0, 1 or more) course sections
- Course section must be taught by 1 instructor (??)
- Instructor may teach many sections
- Course may use many textbooks (all sections use same)
- Textbook may be used in many courses
- Building may contain many rooms
- A room is in only one building
- A course section may use a room
- A room may be used by many course sections (not at same time)

# 1.204 Computer Algorithms in Systems Engineering

Spring 2010