Discussion 1

Java Tutorials & ER Diagrams

EECS 484

Why is this course useful?

- We’re surrounded by databases.

  Examples: mobile apps

- In the first half of this course, we will learn how to use databases.
- In the second half, we will focus on the internal design of databases, which helps us use and design them efficiently.

Logistics

- 9 Discussion sections a week on Thursday and Friday
  - Will cover material from this week's lectures (occasionally an intro to next week's lectures)
  - Do not check attendance, but may talk about things not covered in lecture
  - Can attend any section, but please be consistent and attend the same section every week.
- Homework 1
  - Due Sept. 15, 11:55 PM
  - Groups of 1-2. Homeworks are good prep for the exam
  - Only one member needs to submit. Remember to add the partner on Gradescope.
  - Self-enroll code: **V5DP33** - www.gradescope.com
- Project 1
  - Due Sept. 22, 11:55 PM
  - Groups of 1-2. Make sure to add each other as a group before submitting to the Autograder
  - Part 1 is due on Gradescope
  - Parts 2-4 are due on the Autograder: https://autograder.io/

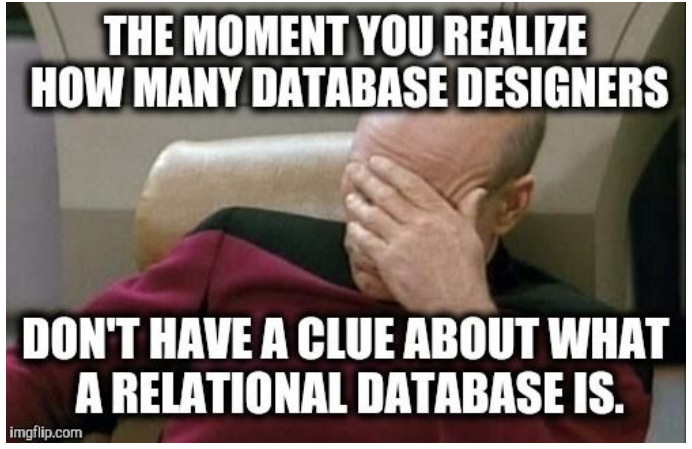
Course Overview

- Intro to database systems
- Entity Relationship (ER) diagrams and the relational model
- Structured Query Language (SQL)
  - Will be spending a lot of time working with (coding assignments)
- Relational Algebra
  - Query language for expressing plans in a mathematical form
- Normalization
  - “Good” way to design relations
- Indexing
  - B+ trees and hash tables
- Query optimization
- Transactions
- Recovery

Database Basics

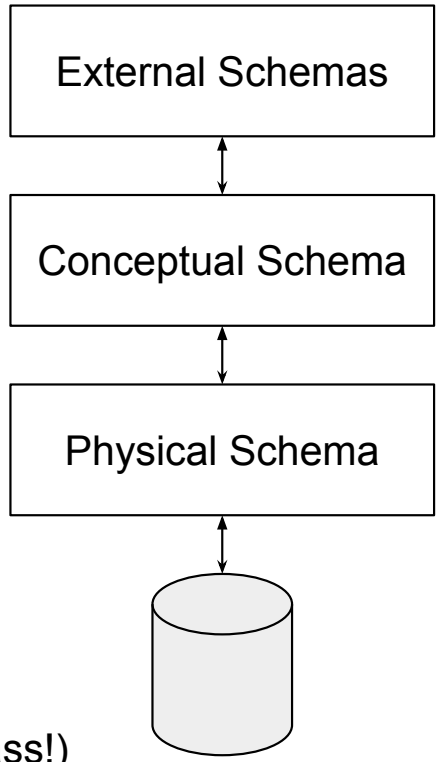
DBMS

- DBMS = Database Management System
  - Oracle SQL, PostgreSQL, MySQL, Transact SQL (Microsoft SQL), etc.
  - Provides declarative system to store data
    - We tell it what we want
    - As opposed to imperative (we don't care how the DBMS stores the data in files)
- Relational database systems
  - Collection of relations (think tables)
  - Defined by a schema
    - Relation name and columns (data type and names)
    - Any other attributes



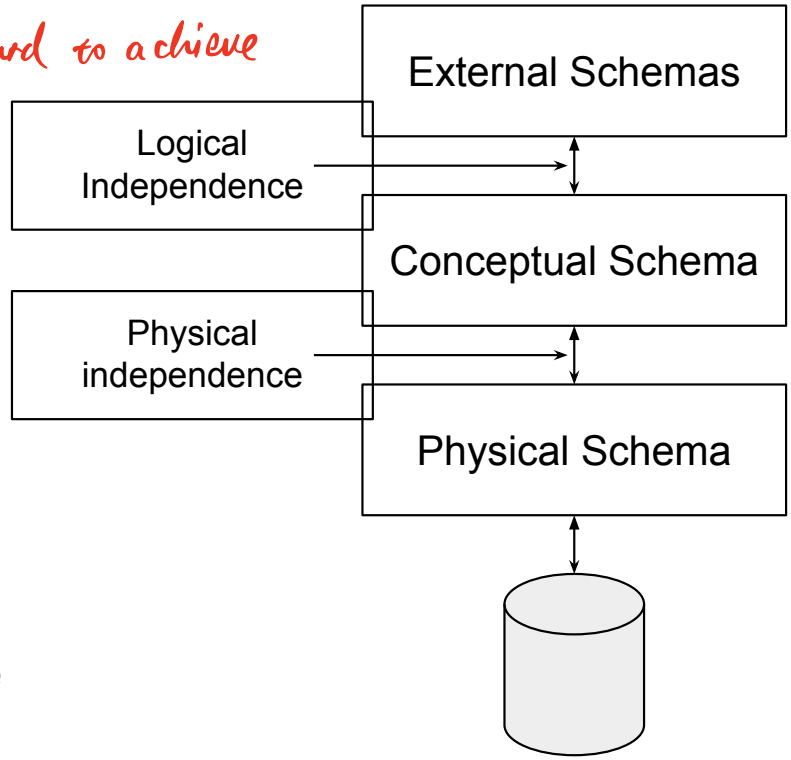
Abstraction

- Different types of schemas
  - Physical schema - how data is stored in memory with files
    - Example - Files for each relation
  - Conceptual schema - what is the the logical structure in terms of the data model
    - Example - Student relation with columns:
      - umid (number)
      - grade percentage (number)
      - name (string)
  - External schema - how is the data represented to a viewer
    - There can be multiple external schemas!
    - Example - Grader view:
      - Grader can see umid, grade (want to obscure name)
    - Example 2 - Canvas coursepage view:
      - Students can see names only (see which friends are in the class!)



Data Independence

- Logical data independence - protection from changes in logical structure of the model
  - Columns in a table within the registrar's database change but instructors don't know anything changed
- Physical data independence - protection from changes in physical structure of the model
  - Oracle releases an update changing how the database is stored on your computer but we don't notice any changes in our pre-existing database
- Logical data independence is hard to achieve!
  - If I change some of the fields, APIs that depend on the data could behave incorrectly
    - Changing primary key from SSN to user_id



Pop Quiz :D

1. In a relational data model, a schema provides what information?
    a. The total size of your table
    b. The data in your table
    c. The data types and the names of the fields

Pop Quiz :D

1. In a relational data model, a schema provides what information?
   a. The total size of your table
   b. The data in your table
   c. The data types and the names of the fields

e.g. name (string)
id (integer)

Pop Quiz the SQL! :D

2. What type of schema abstraction would use for each of the following:
   a. The schema a CAEN admin sees when upgrading the student information database

   b. The schema Canvas displays when showing you the other students in the course

   external

   c. The binary files living in CAEN somewhere that contains your student personal info

   physical

   d. The schema you see in Wolverine Access when editing your personal information

   external

Remember, the types of schemas are external, conceptual, and physical

Pop Quiz the SQL! :D

2. What type of schema abstraction would use for each of the following:
   a. The schema a CAEN admin sees when upgrading the student information database
      Conceptual Schema
   b. The schema Canvas displays when showing you the other students in the course
      External Schema
   c. The binary files living in CAEN somewhere that contains your student personal info
      Physical Schema
   d. The schema you see in Wolverine Access when editing your personal information
      External Schema

Remember, the types of schemas are external, conceptual, and physical

Pop Quiz the 3QL!??! :D

3. If I am updating the DMV database, replacing their eye color with whether a student prefers coffee or tea, which type of data independence will I be most concerned about?

   ☑ Conceptual layer: data type and name.

   a. Logical independence
   b. Physical independence
   c. Probably a bit more concerned about the laws at play
   d. Pls make it stop, I'm tired of the quiz D:

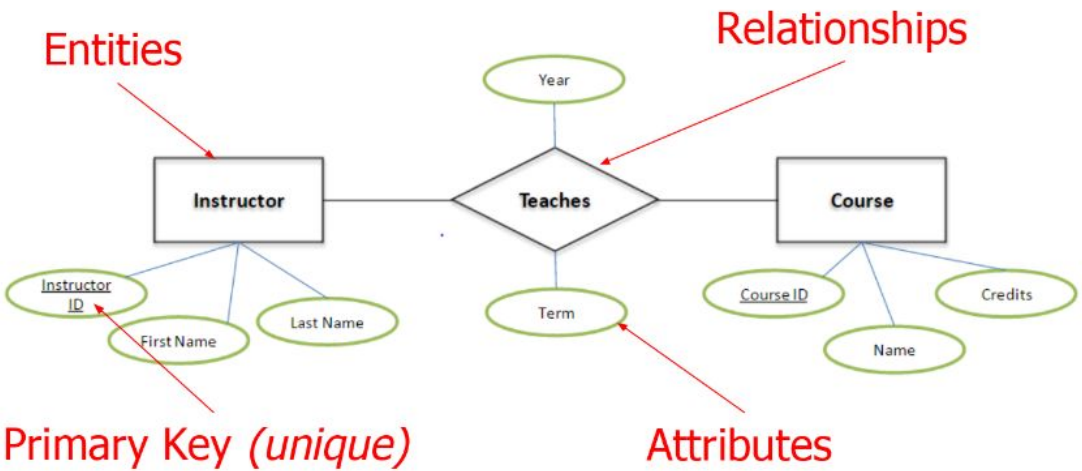
Pop Quiz the 3QL!??! :D

3. If I am updating the DMV database, replacing their eye color with whether a student prefers coffee or tea, which type of data independence will I be most concerned about?
   a. Logical independence
   b. Physical independence
   c. Probably a bit more concerned about the laws at play
   d. Pls make it stop, I'm tired of the quiz D:

ER Diagrams

ER Diagram Basics

- Data model that describes database schema/design
  - Entities are things (Actors, Movies, Citizens, Presidents, Types of Tea)
  - Relationships are are actions/verbs/states (Acted in, Lives in, Is president of, Drinks)
  - Attributes are characteristics (Eye color, Rating, SSN, Political Party, Plant derived from)
    - Primary key is unique identifier (can consist of multiple attributes or just one)



Key Constraints ("at most one")

Many-to-Many

(a)

One-to-One

(b)

(c)

Many-to-One

Many actors can be in one movie

= 'one' is always the thing being pointed at

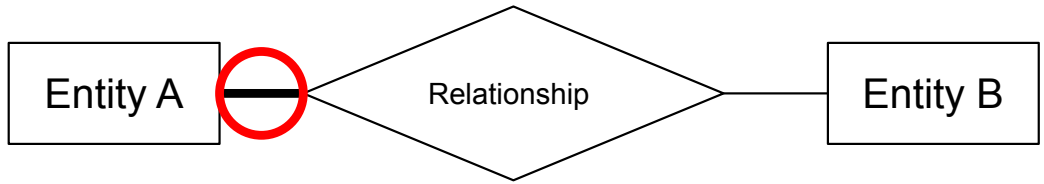
(d)

One-to-Many

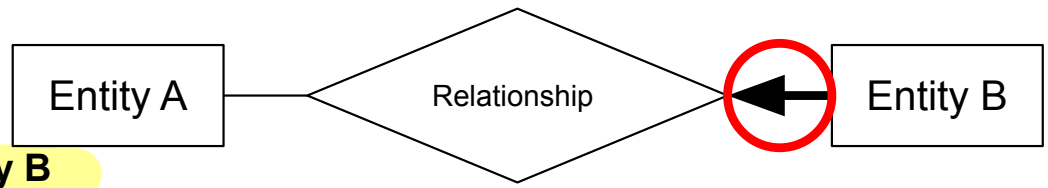
Many movies can have the same actor

Participation Constraints ("at least one")

- Heavy line denotes each Entity A must participate in a relationship with at least one Entity B
  - Could participate with more than one
  - No restriction on Entity B
    - Example: 5 Entity B's do not participate with any Entity A

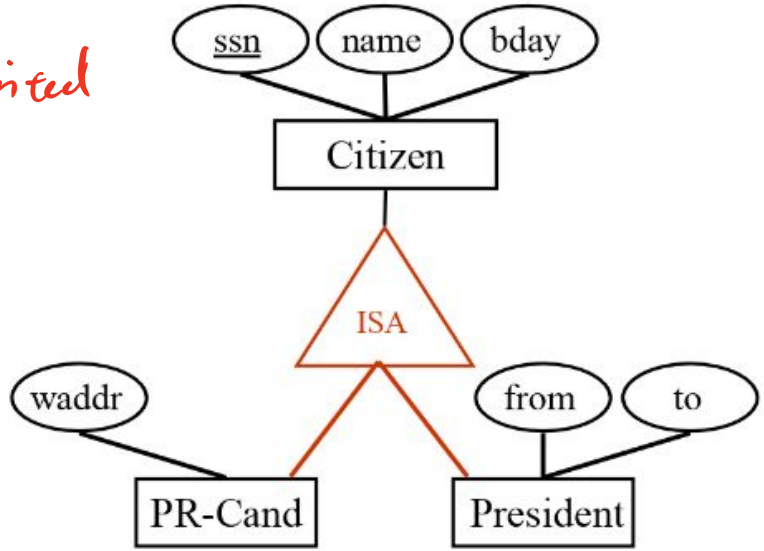


- Heavy line denotes each Entity B must participate in a relationship with at least one Entity A
  - But we know from earlier this arrow means that a single Entity B can relate to at most one Entity A
  - **Net result: 1 and only 1 Entity A per Entity B**



ISA (‘is a’) Hierarchies

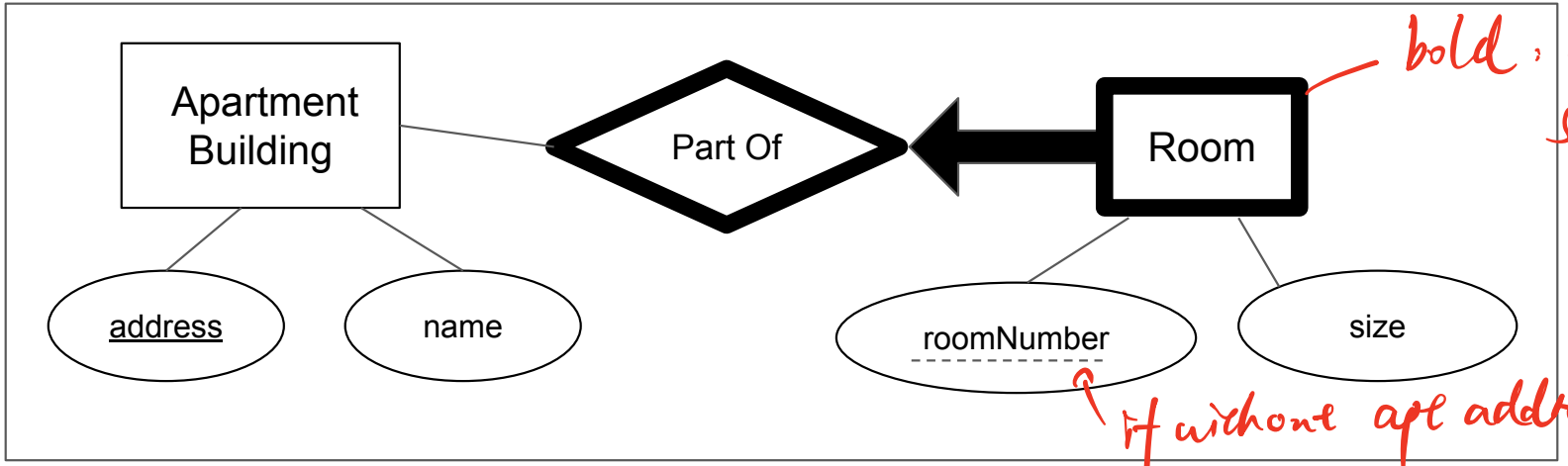
- Equivalent of subclasses
  - All attributes from superclass are in subclasses
- Overlapping vs Disjoint
  - Overlapping if two subclasses can contain the same entity. Otherwise disjoint
    - Example A: Each president was a presidential candidate at some point (overlapping)
    - Example B: A student can either be a graduate or an undergraduate (disjoint)
- Covering vs Partial
  - Is the union of all the subclasses the same as the super class?
    - Example A: Are all citizens either presidential candidates or presidents (no - partial)
    - Example B: Are all students either graduate or undergraduates (yes- covering)



Weak Entities

depend on some else entity exists

- Weak Entity: Room
  - Partial key: roomNumber
    - Primary key: ApartmentBuilding.address and Room.roomNumber
  - Without a building, you can't have a room



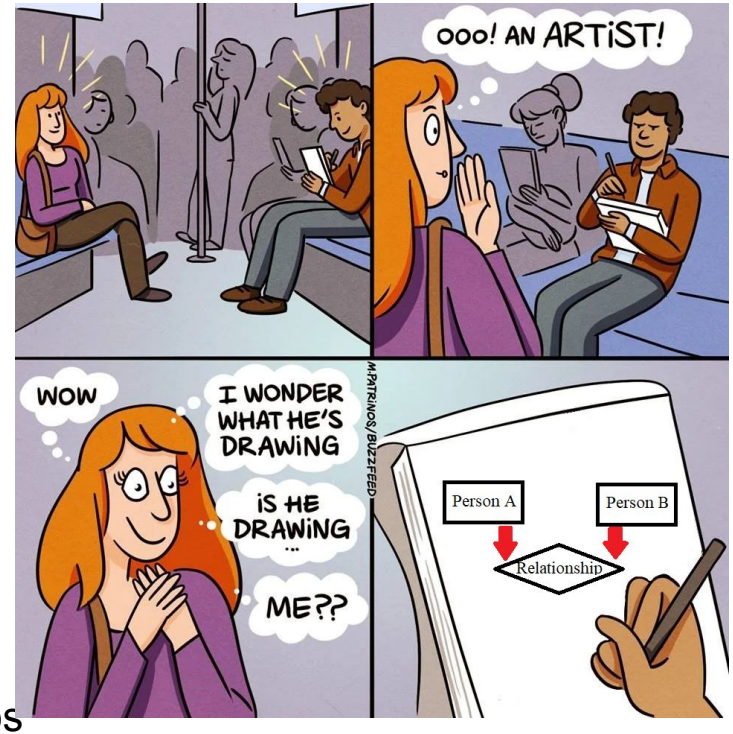
bold: depend on sth else existing

if without apt address ⇒ not unique ID ⇒ partial key

(Fun fact: the only way to convince people that an arrow is bold is to make it comically large)

Creating ER Diagrams

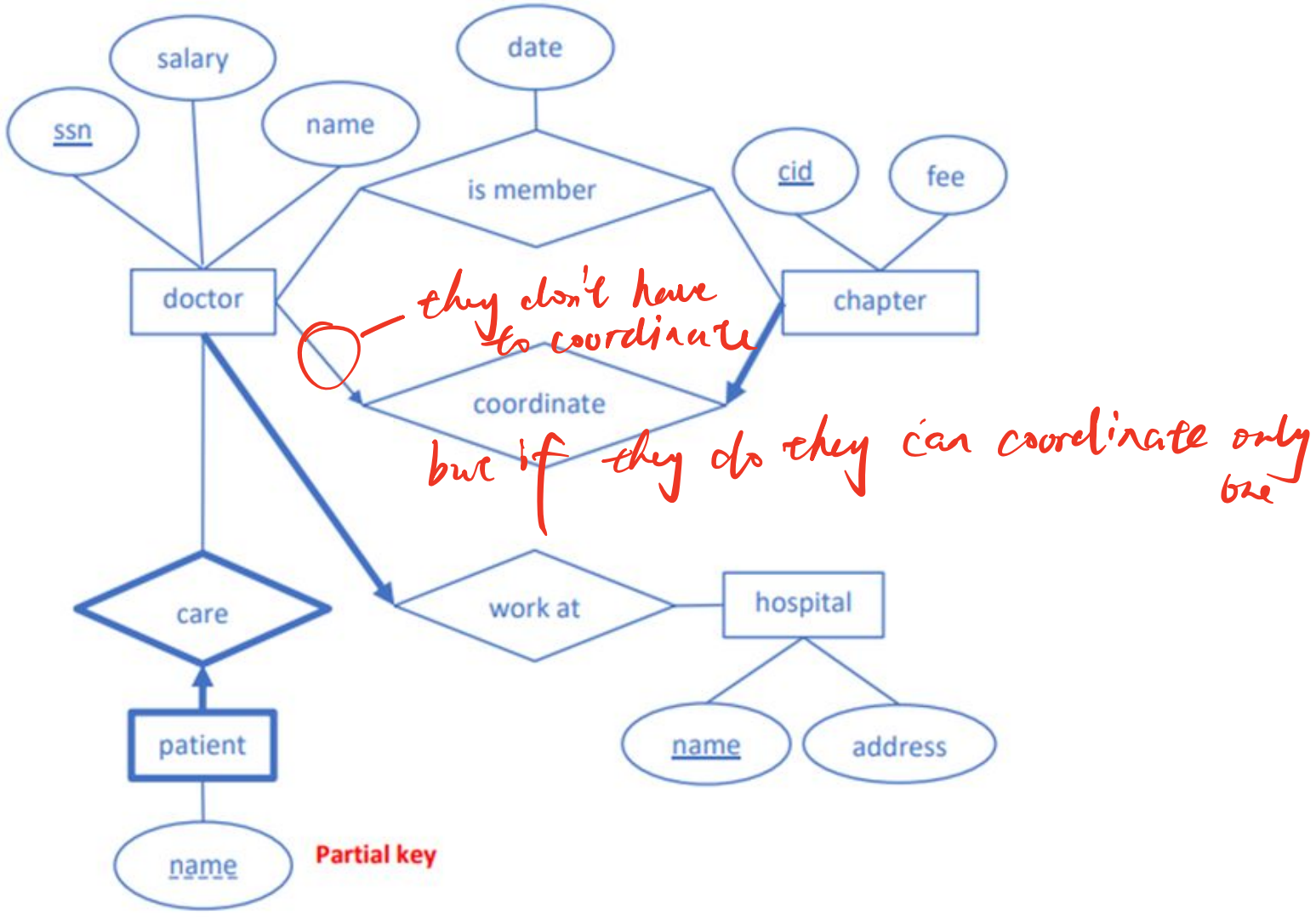
- Three key steps to success
  - Start with the entities and relationships
    - Make your entity squares **(no attributes yet)**
    - Make your relationship diamonds between squares
      - Don't worry about constraints yet
    - Handle ISA hierarchies
  - Add in attributes
    - Determine if they should belong to entity or relationship
    - Determine primary keys
  - Resolve constraints
    - Handle key and participation constraints
    - Determine what weak entities exist
    - Check relationships for potential ternary relationships



ER Diagram Example - Hospital

- Each doctor works at **exactly one** hospital. Doctors have name, salary, and a **unique** ssn. Hospitals have address and a **unique** name.
- Each patient **must** be associated with **exactly one** doctor, and no two patients of a given doctor have the same name (though two patients of the different doctors can have the same name).
- In the database, patient tuples should be automatically deleted if the corresponding doctor tuple is deleted.
- Doctors can join zero or more chapters in the American Medical Association. Each chapter should have a **unique** cid, and a membership fee. It is important to maintain the date on which a doctor joined a chapter.
- Each chapter has **exactly one** coordinator, and only doctors can serve as chapter coordinators. No doctor can coordinate more than one chapter.

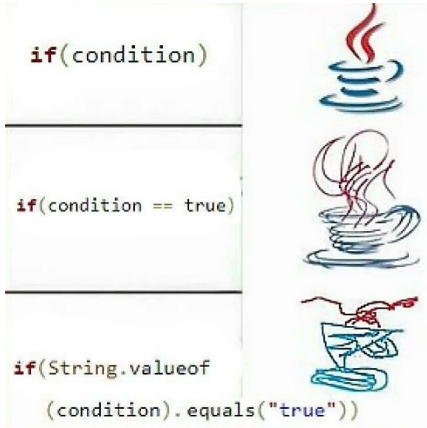
Solution

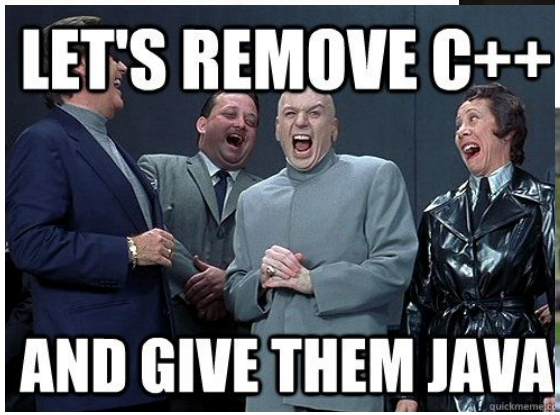

salary
date
ssn
name
is member
cid
fee
doctor
chapter
they don't have to coordinate
coordinate
but if they do they can coordinate only
care
work at
hospital
patient
name
address
name
Partial key

Java Tutorial

Java Tutorial

- We’re going to be using Java!
  - Used in Project 2 extensively
  - Java is made by Oracle
    - Good synergy with Oracle DBMS
- Java has some differences from C++
  - No pointers!!!
    - No dynamic memory leaks!!!!
    - Automatic garbage collection
      - But what if I want dynamic memory? :(
  - Some of the utility classes are weird
    - ArrayList, String
    - System.out.println instead of cout





Java Tutorial

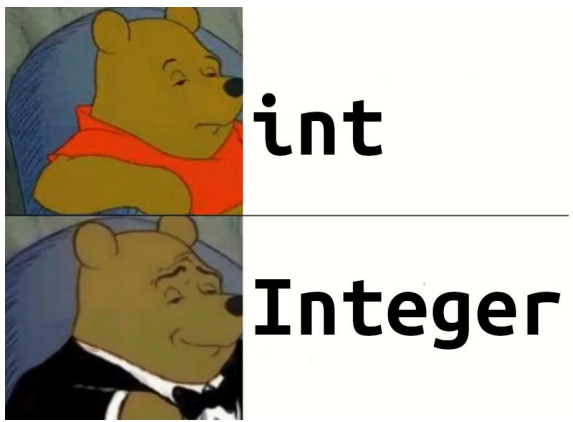
- Java is really good at being used by a lot of different devices
  - Better than C++ for easy GUI development
    - Weak in comparison to C# but that only has support for Windows
- Object oriented to the extreme
  - Everything is a subclass of Object
- Java was designed to be used for the web
  - Uses packages to organize files
  - Historically packages are the domain name backwards
    - com.supersecretpage.eecs484iscool
- Instead of building an exe, java builds jars
  - You can open them up with 7Zip and look at all the files!
  - If you want an exe you have to use a tool like exe4j

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];
int [] numbersArray = new int[10];	int numbersArray[10];
numbersArray[i];	numbersArray[i];
String message = “Hello World”;	string message(“Hello World”);
boolean e = message.equals(“Hello World”);	bool e = message == “Hello World”;

Java Tutorial

- Objects act as if they are passed by reference*
  - Change an array that was passed in a function as a parameter and the original changes
- Primitives cannot be put into collections
  - ArrayList, etc.
  - Use a wrapper class instead
    - int -> Integer, double -> Double, boolean -> Boolean
  - Arrays work like expected however
- Most classes have capital letters
  - String as opposed to string
- New keyword is everywhere
  - Heap objects that are dynamically allocated but managed by Java so it's okay
- API: https://docs.oracle.com/javase/7/docs/api/



*They're not actually passed by reference, but they exhibit a similar behavior

Get started on HW 1!