ODL Basics

ODL = Object Definition Language

- Can be used like E/R as a design language
- Can also be direct input to OODBMS
- Specified by ODMG (Object Database Management Group)
- Comes with a query language called OQL (Object Query Language):
  more on OQL in the second half of the course

ODL Class Declarations

Class declarations: schema (defines types of objects and their relationships)
Objects of classes: data
A class declaration includes:

- attribute declarations:
  
  attribute type name;

- relationship declarations:
  
  relationship rangeType name
  inverse inverseRelationshipName;

- other declarations: method, key, inheritance, etc.

Example: Student and OracleAccount

Example data:

```java
interface Student {
  attribute integer SID;
  attribute string name;
  attribute string address;
  relationship OracleAccount account
    inverse OracleAccount::owner;
}

interface OracleAccount {
  attribute string login;
  attribute integer quota;
  relationship Student owner
    inverse Student::account;
}
```

Jun Yang 1 CS145 Spring 1999
• All relationships are binary!
• Relationships have inverses
• Things from another class are indicated by prefixing otherClassName ::
• Relationships not necessarily implemented by pointers: design is logical, not physical

**ODL Type System**

Allowable attribute types:

(1) Basic types: integer, float, string, Enum
   Example: student type

(2) Struct built from (1)
   Example: address

(3) Set, Bag, List, Array of (1) or (2)
   Example: set of addresses

Allowable relationship types:

(1) Interface types

(2) Set, Bag, List, Array of (1)
   Example: students take courses

**Multiplicity of Relationships**

If class C is the “many” in a relationship:
\[ \sim \rightarrow \] the relationship to C has type Set\(<\text{C}\>

If class C is the “one” in a relationship:
\[ \sim \rightarrow \] the relationship to C has type C

• Many-many: students take courses
• Many-one: courses are taught by instructors

• One-one: students and their oracle accounts
Multiway Relationships in ODL?

No such things; let’s hack!
Remember the E/R trick of using a connecting entity set and n binary relationship sets to model a n-ary relationship set? Essentially the same trick—introduce a connecting class
Example: students, courses, TA’s + enrollment

Keys in ODL

Like E/R:
- A key is set of attributes whose values uniquely identify an object in a class
Unlike E/R:
- Keys are completely unnecessary in ODL because “object identity” (OID) serves to distinguish objects
- Some classes don’t even have attributes, let alone keys (e.g., a connecting class)
- Multiple keys can be specified
Syntax:
interface Student (key SID) {...}
interface Student (key (name, address)) {...}
interface Student (key SID, (name, address)) {...}

Subclasses in ODL

Follow name of subclass by colon and its superclass:
interface GradStudent:Student {...}
GradStudent objects acquire all attributes and relationships of the Student class
Difference in subclass viewpoints:

- In ODL, an object is in exactly one class
  \[\leadsto\] it inherits properties of its superclass(es)

- In E/R, an entity has “representation” in all classes to which it logically belongs
  \[\leadsto\] its properties are the union of the properties of these classes

- This distinction matters later, when we convert ODL and E/R to relations

Example: