Intro to OO Design: Mentor DB example

9 Nov 2010
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

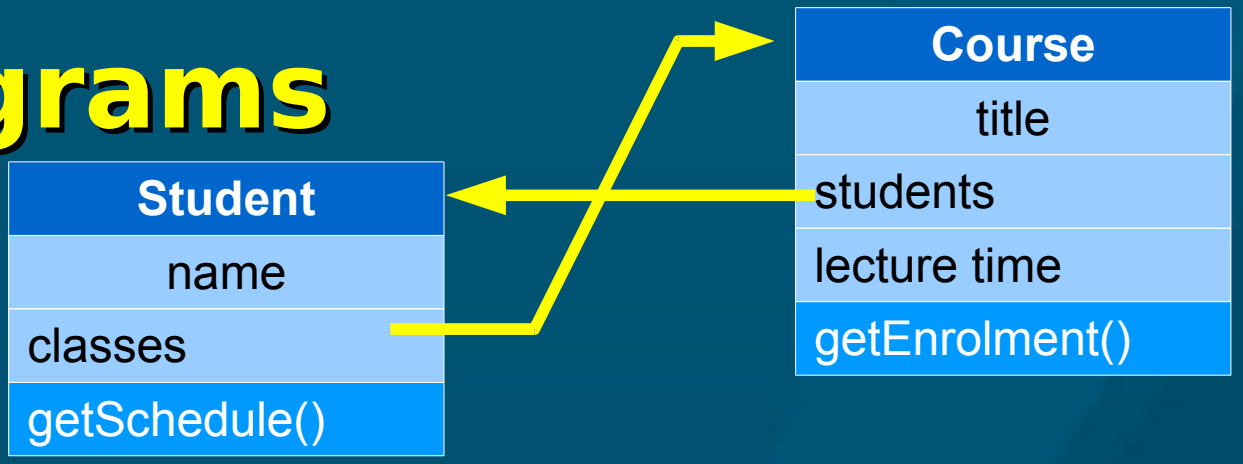
Outline for today

- Top-down OO design, class diagrams
- MentorDB example
- Data model: objects and relationships
  - Multiplicity
- Functional specification: methods
  - add a student
  - search database by name
  - find all mentees of a mentor
  - save/load to disk

Top-down design

- 'D' in WADES: breaking down the task into smaller chunks
- Data model: what information to track? (nouns)
  - Classes and attributes
- Functional specification: what actions? (verbs)
  - Methods in classes
- “Show me your flow charts and conceal your tables and I shall continue to be mystified, show me your tables and I won't usually need your flow charts; they'll be obvious.”
  (Fred Brooks, “The Mythical Man-Month”, 1975)

OO class diagrams



- An OO program is a collection of classes
  - Create objects: instances of the classes
  - Objects pass messages to each other
- A class diagram shows the classes and their relationships to each other:
  - Name of class
  - Attributes (public and private)
  - Methods (public and private)

Example: Mentoring Program

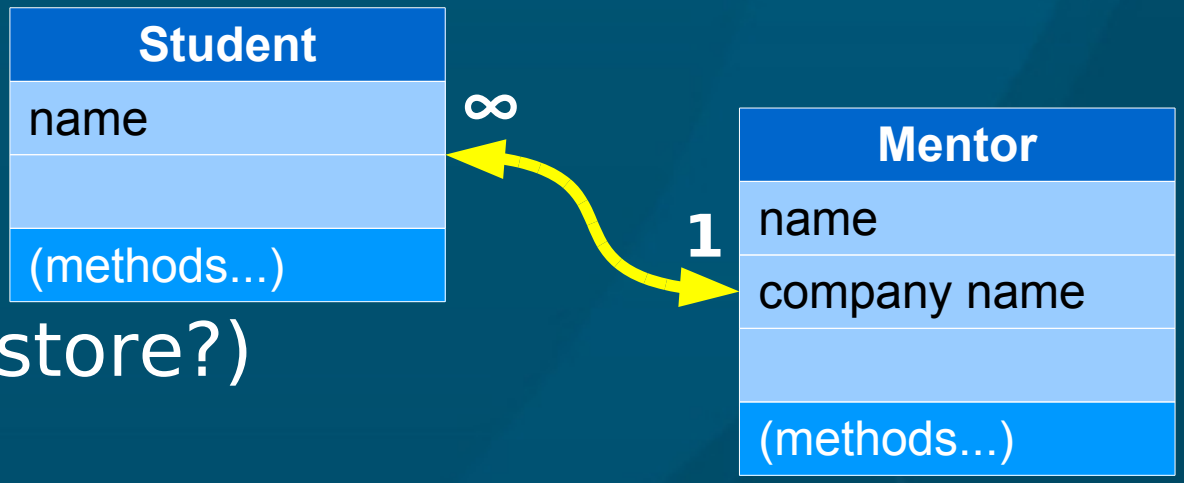
- TWU School of Business mentoring program:
  - Matches students with local businesspeople
  - Each student has 1 (or 2) mentors per year
- Keep track of students and mentors
- View sorted lists of students, mentors
- Select a mentor,
  and see all students they have mentored
- Also store contact info, photo, etc.
- Other requirements? → extensible design

MentorDB: simplified spec

- Let's make a few simplifying assumptions:
- Biographical info:
  for each student/mentor, limit to just name,
  plus company name for mentor
- Let's not worry about sorting the output for now
- Assume each student has only one mentor
  (but still allow each mentor to have multiple mentees)
  - → Many-to-one relationship
    (many students to one mentor)

MentorDB: data model

- Classes, attributes?
- Student:
  - name
  - mentor (how to store?)
- Mentor:
  - name
  - company name
  - mentees? Duplicate information?
    Store the relationships in only one place!
- MentorDB: list of Students, list of Mentors



How to store relationships?

- Multiplicity: *N* students to *M* mentors
  - 1:1 – every student has exactly 1 mentor, and vice-versa
  - 1:0..1 – every student has 0 or 1 mentors
  - 0..*:0..1 – every student has 0 or 1 mentors; mentors may have any number of students
- Store relationship in Student (.mentor attribute)
- If many-many (0..*:0..*) relationship, may need to store relationships in separate MentorDB
- Store ID# of mentor, or store a reference/alias to the mentor object

Create classes with attributes

(See mentors.py)

- class Student: name, mentor (obj)

```
def __init__(self, name='', mentor=None):
    self.__name = str(name)
    self.__mentor = mentor

def __str__(self):
```

  - Also set/get methods for name and mentor

- class Mentor: name, company name

```
def __init__(self, name='', company=''):
```

- class MentorDB: student list, mentor list

```
def __init__(self):
    self.__students = []
    self.__mentors = []
```

MentorDB: functional spec

- Student: (attribs: name, mentor)
  - init, str, set/get
- Mentor: (attribs: name, company)
  - init, str, set/get
- MentorDB: (attribs: student list, mentor list)
  - print all students, print all mentors
  - add/del a student/mentor
  - find student/mentor by name (return obj)
  - … anything else?
  - (How to change a student's name?)

MentorDB: add a student

- MentorDB obj stores two lists (student, mentor)
  - List of objects: self.__students
- Add a student to the student list:
  - Assumes a student object has been created

```
def addStudent(self, newStudent):
    if newStudent is not None:
        self.__students.append( newStudent )
```

  - Uses the .append() method of lists
  - Error checking: should also check type of param newStudent

Search database by name

- Search through student list looking for a name:

```
def findStudent(self, targetName):
    for student in self.__students:
        if student.getName() == targetName:
            return student
```

- Parameter: a string targetName to look for
- Iterates through list of students
- Uses getName() set/get method of Student
- Returns an alias/ref to the Student object
  - Client code can then modify the student's record (mutable)

Find all mentees of a mentor

- Given a mentor (ref to a Mentor object), find all students who have that mentor (return a list of references to Student objects)

```
def getMentees(self, mentor):
    mentees = []
    for student in self.__students:
        if student.getMentor() == mentor:
            mentees.append( student )
    return mentees
```

- getMentor() method of Student returns an obj

Save/load DB to disk

- Methods in MentorDB: save(file), load(file)
  - Specify filenames as strings
- Use pickle.dump() and pickle.load()
  - Need to open files in binary mode: 'wb', 'rb'

```
def save(self, fileName):
    import pickle
    with open(fileName, 'wb') as dbFile:
        pickle.dump( self.__students, dbFile )
        pickle.dump( self.__mentors, dbFile )
```

- Stored in Python's own binary pickle format
  - (you can open up the file in WordPad to see)