COMP2511

Object-Oriented Design and Programming

Object-Oriented Design

Wayne Wobcke

w.wobcke@unsw.edu.au

Today's Lecture

- Object-Oriented Design Process
  - Use Cases & Walkthroughs
  - CRC Cards
  - UML Class Diagrams
  - UML Sequence Diagrams
- Collections

Enrolment System

- Students enrol in courses that are offered in particular semesters
- Students receive grades (pass, fail, etc.) for courses in particular semesters
- Courses may have prerequisites (other courses) and must have credit point values
- Course offerings are broken down into multiple sessions (lectures, tutorials and labs)
- Sessions in a course offering for a particular semester have an allocated room and timeslot
- If a student enrols in a course, s/he must also enrol in some sessions of that course

Use Case

1. System shows list of courses
2. User selects a course
3. User asks System to enrol in course
4. System checks prerequisites passed
5. System shows list of sessions
6. User chooses session
7. System enrols User in course

- Not a Use Case Diagram

CRC Cards

Class	
Responsibilities	Collaborators

- Responsibilities and Collaborators don't have to "line up"

Walkthrough

1. EnrolSystem asks Course for list
2. User selects course
3. EnrolSystem asks Course for prereqs
4. EnrolSystem asks Student if passed
5. EnrolSystem asks Session for free list
6. User chooses Session
7. EnrolSystem asks Session to enrol student
8. EnrolSystem creates Enrolment

UML Class Diagram

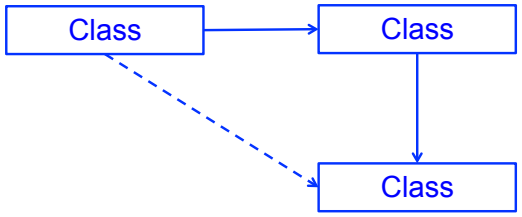
- Dependency
- Association
- Aggregation
- Composition
- Inheritance (extends)
- Realization (implements interface)

Enrolment System

- Could Offering inherit from Course?

Law of Demeter & Loose Coupling

- Avoid (if there is a better way)



Law of Demeter

- Method *m* of object *o* can call methods of
  - *o* itself
  - *m*'s parameters
  - objects created within *m*
  - *o*'s component objects
  - not those of objects returned by a method
- Avoid train wrecks
  - e.g. o.get(name).get(thing).remove(node)

Lists and Iterators

```
ArrayList<String> list = new ArrayList<String>();
Iterator i = list.iterator();

// iterate over list
while (i.hasNext())
        s = i.next();
        // process s

for (String s: list)
        // process s
```


Next Week

- Generic Types
- Polymorphism