UML diagrams and Modules vs. Classes

14 Jan 2009
CMPT166
Dr. Sean Ho
Trinity Western University

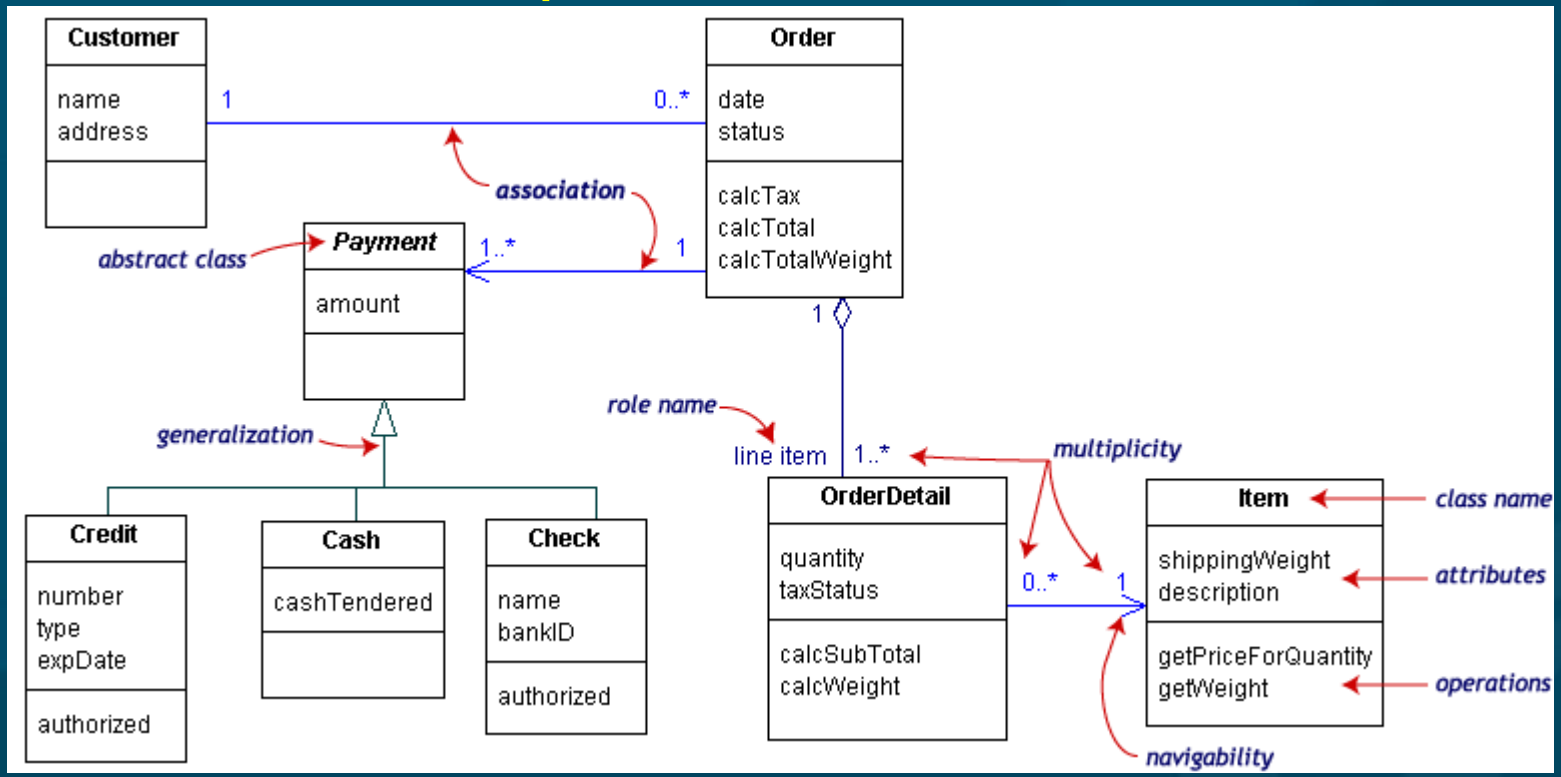
TRINITY WESTERN UNIVERSITY

UML: Unified Modeling Language

- Diagrams for use in designing your programs
- Main diagram types:
  - Static: Class diagram, object, package
  - Dynamic: Use case diagram, sequence diagram, state chart
- Handy for diagramming by hand, or
- UML software tools, e.g., Visio, Sun JSEnterprise
- Developed by Booch, Rumbaugh, and Jacobson, of OMG (Object Management Group)
- Current version is 2.0: www.uml.org

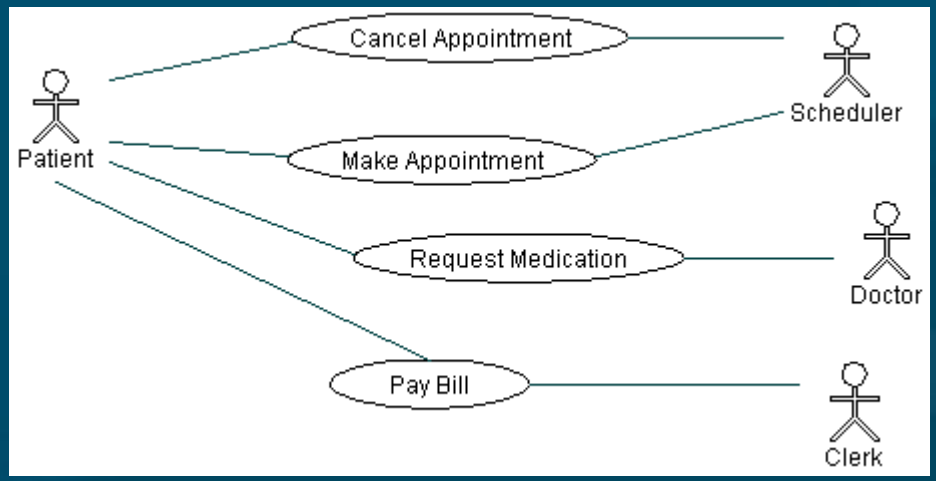
UML: Class diagram

- Each box represents a class (type)
  - Name, attributes, methods
- Lines show relationships between classes



UML: Use case diagram

- Describes relationships between actors:
  - Patient calls the clinic to make an appointment
  - Receptionist books timeslot
  - Patient sees doctor and requests medication
  - Patient pays bill to clerk



- See Borland's UML tutorial for more details

Design patterns

- Commonly used software designs
- Not reinventing the wheel
  - Similar to libraries, but for program design
- Similar to architectural elements: arch, column
- "Gang of Four" standard reference (1995):
  - Gamma, Helm, Johnson, Vlissides, "Design Patterns: Elements of Reusable OO Software"
  - Creational patterns: e.g., abstract factory
  - Structural patterns: e.g., proxy
  - Behavioural patterns: e.g., observer, MVC

Modules vs. Classes

- Both modules (M2) and classes (OO):
  - Have both a public interface (DEF) and a private implementation (IMP)
  - Allow data hiding (in the private portion)
- But there are differences:
  - Data items in modules are singletons;
    - Each instance of a class has its own data items
  - Modules in M2 are not types; OO classes are
  - Modules cannot be derived from other modules
    - Classes can inherit (subclass) from other classes

Declaring classes: OO-M2

- Declaring a class in object-oriented M2:

```
CLASS Rectangle;
    CONST
        sides = 4;
    VAR
        length, width: INTEGER;
    PROCEDURE SetDims (l, w: INTEGER);
    BEGIN
        length := l;
        width := w;
    END SetDims;
BEGIN
        SetDims (0, 0);
END Rectangle;
```

Declaring classes: C++

- Header (public definition) file:

```
class Rectangle {
    const int sides = 4;
    int length, width;
    void SetDims (int l, int w);
}
```

- Code (private implementation) file:

```
void Rectangle::SetDims (int l, int w) {
    length = l;
    width = w;
}
```