Composition, Inheritance

EE 312

ECE, UT Austin

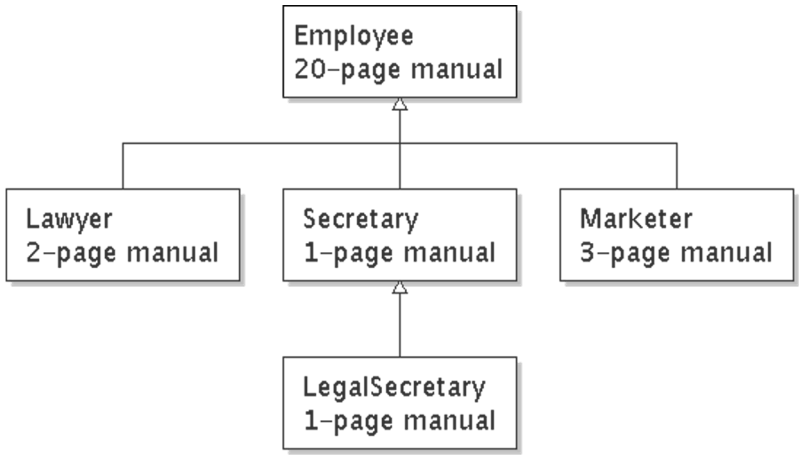
Composition

- A class has member objects that make up, or *compose*, the class.
  - E.g. a Node class has two other Node objects – left and right.
  - A Car class has an Engine object as a member.
  - A Customer class has a string object as a member.

Law firm employee analogy

- common rules: hours, vacation, benefits, regulations ...
  - all employees attend a common orientation to learn general company rules
  - each employee receives a 20-page manual of common rules

- each subdivision also has specific rules:
  - employee receives a smaller (1-3 page) manual of these rules
  - smaller manual adds some new rules and also changes some rules from the large manual



Separating behavior

- Why not just have a 22 page Lawyer manual, a 21-page Secretary manual, a 23-page Marketer manual, etc.?

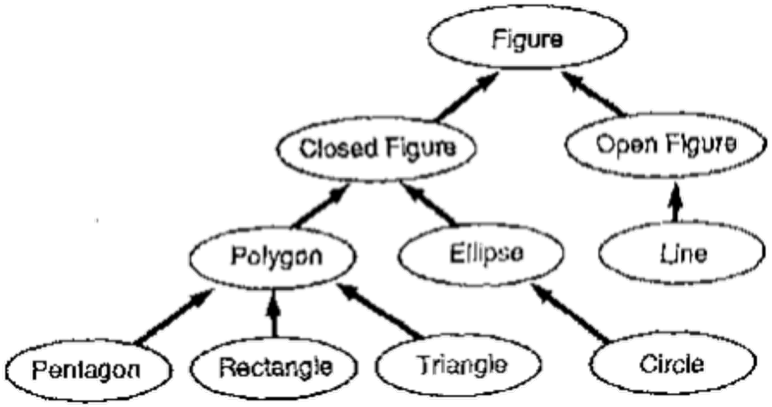
- Some advantages of the separate manuals:
  - maintenance: Only one update if a common rule changes.
  - locality: Quick discovery of all rules specific to lawyers.

- Some key ideas from this example:
  - General rules are useful (the 20-page manual).
  - Specific rules that may override general ones are also useful.

Is-a relationships, hierarchies

- **is-a relationship**: A hierarchical connection where one category can be treated as a specialized version of another.
  - every marketer *is an* employee
  - every legal secretary *is a* secretary

- **inheritance hierarchy**: A set of classes connected by is-a relationships that can share common code.



Employee regulations

- Consider the following employee regulations:
  - Employees work 40 hours / week.
  - Employees make $40,000 per year, except legal secretaries who make $5,000 extra per year ($45,000 total), and marketers who make $10,000 extra per year ($50,000 total).
  - Employees have 2 weeks of paid vacation leave per year, except lawyers who get an extra week (a total of 3).
  - Employees should use a yellow form to apply for leave, except for lawyers who use a pink form.

- Each type of employee has some unique behavior:
  - Lawyers know how to sue.
  - Marketers know how to advertise.
  - Secretaries know how to take dictation.
  - Legal secretaries know how to prepare legal documents.

A Useful class

```
#ifndef USEFUL_H
#define USEFUL_H

class X {
  int i;
public:
  X() { i = 0; }
  void set(int ii) { i = ii; }
  int read() const { return i; }
  int permute() { return i = i * 47; }
};
#endif // USEFUL_H ///:~
```

Desire for code-sharing

- We'd like to be able to say:

```
// A class to represent Y.
class Y {
    copy all the contents from the X class;

    public void newFnUniqueToY () {
    }
}
```

A Derived class

```
class Y : public X {
      int j; // Different from X's i
    public:
      Y() { j = 0; }
      int change() {
        j = permute(); // Different name call
        return j;
      }
      void set(int ii) {
        j = ii;
        X::set(ii); // Same-name function call
      }
};
```

Inheritance

- **inheritance**: A way to form new classes based on existing classes, taking on their attributes/behavior.
  - a way to group related classes
  - a way to share code between two or more classes

- One class can *extend* another, absorbing its data/behavior.
  - **Base class**: The parent class that is being extended.
  - **Derived class**: The child class that extends the base class and inherits its behavior.
    - Derived class gets a copy of every field and method from base class

Overriding methods

- **override**: To write a new version of a method in a derived class that replaces the base class's version.
  - No special syntax required to override a base class method. Just write a new version of it in the derived class.

Levels of inheritance

- Multiple levels of inheritance in a hierarchy are allowed.