Object-Oriented Software Design
and Construction with C++

Computer Science 2704

Mike Parks
Summer 1999

Based on Slides by Dennis Kafura Summer, 1998

Beginning

• Overview of the course
• Lab Accounts and Software
• Progression of Roles
• First Topics
  – abstraction
  – separation

Class Web Page

http://ei.cs.vt.edu/~cs2704/summer99/
Lab accounts and software

- If you do not have an account for the 116/118 lab, fill out the form being distributed (you must have a university PID).
- The software for the course is installed on all of the lab machines and should be used for your course work. You may also set up the software on your own machine.

Progression of Roles

Design Strategies in OOP

- Abstraction
  - modeling essential properties
- Separation
  - treat what and how independently
- Composition
  - building complex structures from simpler ones
- Generalization
  - identifying common elements
Connections

Abstraction

A named collection of attributes and behavior relevant to modeling a given entity for some particular purpose.

Properties of a Good Abstraction

- well named
- coherent
- accurate
- minimal
- complete
Good Abstractions

Mapping Abstraction to Software

Separation

In object-oriented programming, the independent specification of an interface and one or more implementations of that interface.
**Multiple Instances of a Class**

**Software Engineering Goals**

**Quick Overview of Terms**

- **Encapsulation:**
  - only access “internals” through interface
- **Separation:**
  - keeping interface & implementation separate
- **Composition:**
  - make collection of objects (linked list of nodes)
- **Generalization:**
  - “All windows have titles”: organize objects around common properties
### Composition: 2 Kinds

- **Association:**
  - Compose independent objects
  - *Example:* Node object
    - Linked list is composed of nodes that point to each other
- **Aggregation:**
  - Put bunch of objects in a box, close box, and use box as a new object with hidden internals
  - *Example:* hash table object (hidden node objects)

---

### Generalization: 4 Kinds

- **Hierarchy:**
  - Graph representing "is a" relationship
- **Genericity:**
  - "Parameterized" object: Sort object w/ type (int) parameter
- **Polymorphism ("many forms"):**
  - Loop to "iterate" through list - works for arrays or linked list
- **Patterns:**
  - Way to structure objects (e.g., client/server system)

---

### Example of Hierarchy

```
Component
  ├── Button
  │    └── CheckBox
  │    └── ScrollBar
  └── Label
```

---
