### Object-Oriented Design Strategies

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

### Abstraction

- Use of classes to represent objects of program
- Objects may serve many purposes

### Properties of a Good Abstraction

- well named – clearly identifies abstraction
- coherent – sensible description
- accurate – only attributes of entity
- minimal – no irrelevant attributes
- complete – everything needed

### Separation

- Separate “what” (interface) from “how” (implementation)
- Goal to localize effect of changing class implementation
- Approach: make interfaces sufficiently abstract to avoid committing to particular implementation
Templates and Separation

- A template class is a parameterized type
- All template classes make some assumptions about type parameter
- With fewer assumptions,
  - more classes can be given as parameter
  - fewer dependencies on assumptions

Inheritance and Separation

- Inheritance creates dependencies between base and derived classes
- Short circuits separation provided by interface
- Dependency is implementation to implementation

Inheritance and Separation

- Can strengthen “separation” when using inheritance by
  - Choosing strong class relationships
  - Using only public inheritance
  - Avoiding protected data

Composition

- Aggregation and Association
- Primary mechanisms for constructing programs
- Differences
  - Where created and destroyed
  - Who has control
  - Who are clients of object

Generalization

- Support reuse of code
  - Code that can be used for different, but similar purposes
  - Data types that can be used in different settings
- Polymorphism in any of its forms
  - templates
  - inheritance and virtual methods
  - operator overloading