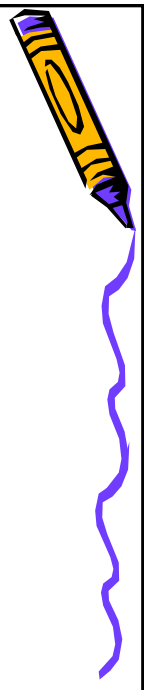
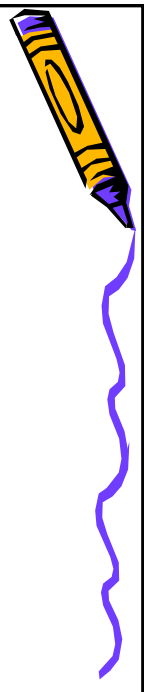
Motivations



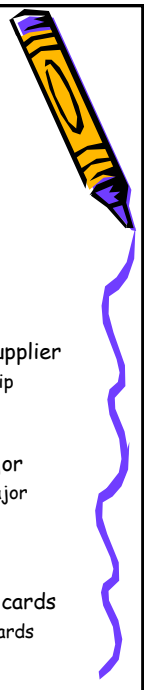
- We've
  - Used console window (TUI)
  - Introduced OO
    - Most we've had is 3 classes
- What about
  - More complex OO
    - What about needing 4 classes? 10 classes? 50 classes?
    - How do these classes relate to each other?

More about Classes and Objects



- Topics
  a) Relationships between classes
  b) Introduction (i.e., review) of testing
  c) 1D Arrays; More on loops; Recursion

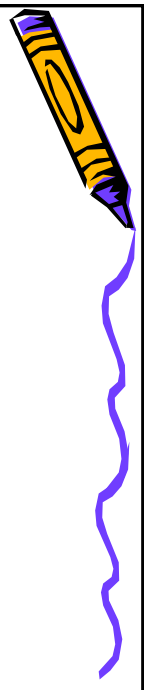
Relationships between Classes



- Inheritance
  - Parent-child relationship — e.g., corn is a vegetable
    - i.e., like biology - child class inherits behavior from parent class
- Association
  - Strong supplier-client relationship — i.e., client references supplier
  - — typically a 1-to-1 relationship
- Aggregation
  - Ownership, non-exclusive — e.g., a student has a major
  - — many students have this major
  - — a 1-to-many relationship
- Composition
  - Ownership, exclusive — e.g., I have many credit cards
  - — no one else has my credit cards
  - — a 1-to-many relationship

CSC 176 More about Classes and Objects Slide 3

Relationships between Classes (cont'd)



- Discuss UML Class Diagram Relationships (pdf)
  - Inheritance
  - Association
  - Aggregation
  - Composition
  - For now, ignore relationship types:
    - Realizes
    - Dependency

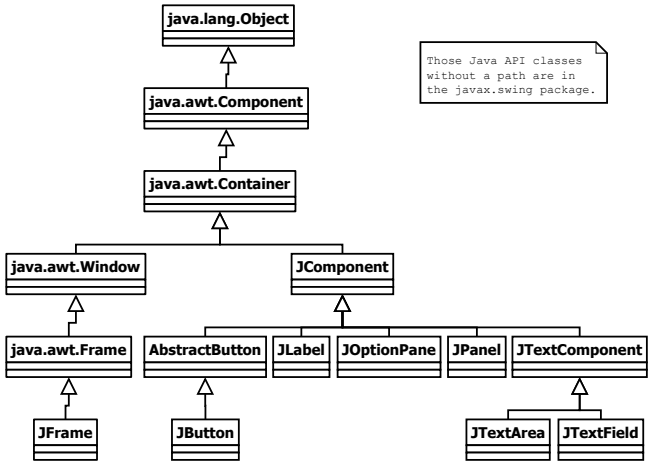
CSC 176 More about Classes and Objects Slide 4

Relationships Example

- Demo Javax Swing version of BMI_6
  - BMI_6_Main_Swing
  - BMI_6_GUI_Swing
  - BMI_5_bmi
- Based on demo
  - Show relationships between the classes/objects in this simple app
    - Draw class diagram for the application
  - Show relationships (next slide) between Javax Swing classes used in demo

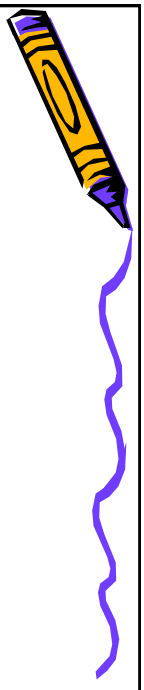
Relationships Example

(javax.swing classes used in BMI_6)



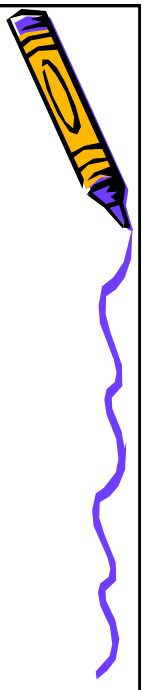
OO Relationships

(Identify, Draw)



- Types of relationships (we're focusing on)
  - Inheritance — parent-child relationship; generalization and specialization
  - Association — uses object reference(s); refers to another object
  - Aggregation — whole-part relationship; non-exclusive ownership
  - Composition — whole-part relationship; exclusive ownership
- Identify the type of OO relationship
  - A novel is a book
  - A student takes a class
  - A motorcycle has two wheels
  - A mother has a child
  - A hammer is a tool
- To-do
  - Draw class diagram for each relationship
  - Identify sample attributes and methods
  - Write code to demo use of inheritance, aggregation, composition

Textbook



- Previously covered
  - Most of chapter 2 (Java basics) & chapter 3 (selection)
  - Covered 4.2 (common math functions)
  - 5.2 (while)
  - 6.2, 6.3 & 6.5 (defining/calling a method, passing args)
  - 9.2, 9.4 & 9.6 (define class, construct obj, Java API)
  - 12.1 & 12.2 (exception handling)
  - Each class in its own source code file!
  - Better separation of concerns and design for reuse
- Just covered
  - 10.2, 10.3, 10.4 (abstraction, encapsulation, objects, relationships)
  - 11.2, 11.3 (superclass, subclass, super keyword)