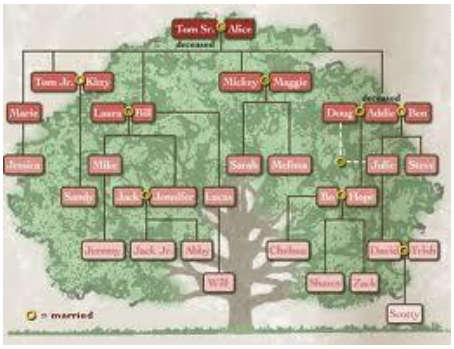
CSC115 Lecture 22

Inheritance – The Java Heirarchy



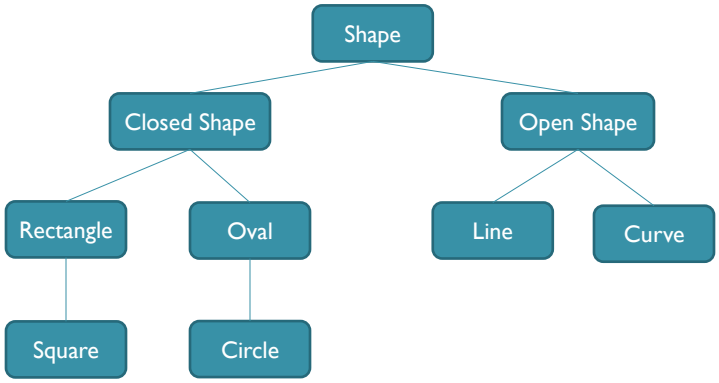
Some terms today

- Inheritance
- Super Class
- Sub Class
- Is-A
- Has-A
- Dynamic Binding
- Abstract Method / Abstract Class
- (Compare to: Interface)

Watch for them…Try to write definitions for them , or distinguish the differences between them.

Inheritance

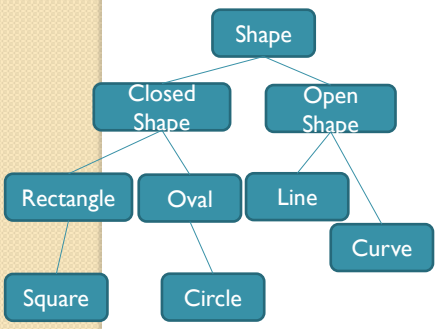
- Allows a class to derive the behavior and structure of an existing class



Inheritance - Terminology

- Superclass or base class
  - A class from which another class is derived
- Subclass, derived class, or descendant class
  - A class that inherits the members of another class
- Benefits of inheritance
  - It enables the reuse of existing classes
  - It reduces the effort necessary to add features to an existing object

The Shape Example



Shape attributes:

- x and y coordinates

Closed Shape attributes:
(in addition to x & y)

- boolean: is it a rectangle or an oval?
- (for rectangle) array of x,y points
- (for oval) bounding box

Open Shape attributes:

- length of the shape

Inheritance

- A subclass can
  - Add new members to those it inherits
  - Override an inherited method of its superclass
    - A method in a subclass overrides a method in the superclass if the two methods have the same declarations
    - Replacement: Provides a new implementation for the method
    - Refinement: Uses the superclass method as part of the subclass method.
      - <Examples next slide>

Constructors are automatically *refined*

The Shape Example

<See example code in Lecture 22 Code folder>

Replacement and Refinement

Notice that method defined in a superclass can be re-defined in the subclass

Inheritance

- A subclass inherits private members from the superclass, but cannot access them directly
- Methods of a subclass can call the superclass's public methods
- Clients of a subclass can invoke the superclass's public methods
- An overridden method
  - Instances of the subclass will use the new method
  - Instances of the superclass will use the original method

Java Access Modifiers

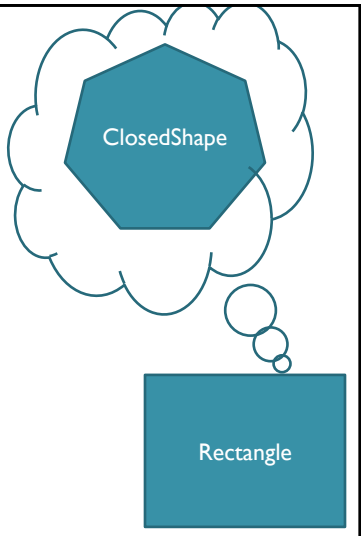
- Membership categories of a class
  - Public members can be used by anyone
  - Members declared without an access modifier (the default) are available to
    - Methods of the class
    - Methods of other classes in the same package
  - Private members can be used only by methods of the class
  - Protected members can be used only by
    - Methods of the class
    - Methods of other classes in the same package
    - Methods of the subclass

Is-a and Has-a Relationships

- Two basic kinds of relationships
  - Is-a relationship
  - Has-a relationship

Is-a Relationship

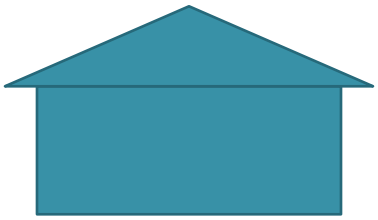
- Inheritance impies an is-a relationship between the superclass and the subclass
- Example:
  - If the class `Rectangle` is derived from the class `ClosedShape`
    - A rectangle is a ClosedShape



- Compatibility: An instance of a subclass can be used instead of an instance of the superclass, but not the other way around

Has-a Relationships

A house "has a" or "contains a" rectangle



- Also called containment
- Cannot be implemented using inheritance
  - Example: To implement the has-a relationship between a pen and a ball
    - Define a data field `rectangle` – within the class `DrawAHouse`

Dynamic Binding

- A polymorphic method: A method that has multiple meanings
  - Created when a subclass overrides a method of the superclass
- Late binding or dynamic binding
  - The appropriate version of a polymorphic method is decided at execution time

Abstract Classes

- An abstract class is used only as the basis for subclasses
  - It defines a minimum set of methods and data fields for its subclasses
- An abstract class has no instances
- An abstract class should, in general, omit implementations except for the methods that
  - Provide access to private data fields
  - Express functionality common to all of the subclasses
- A class that contains at least one abstract method must be declared as an abstract class
- A subclass of an abstract class must be declared abstract if it does not provide implementations for all abstract methods in the superclass

Java Interfaces Revisited

- A Java interface
  - Specifies the common behavior of a set of classes
  - Common uses
    - Facilitate moving from one implementation of a class to another
      - A client can reference a class's interface instead of the class itself
    - Specify behaviors that are common to a group of classes

Abstract Classes

- Abstract classes
  - An abstract class is used only as the basis for subclasses
    - It defines a minimum set of methods and data fields for its subclasses
  - An abstract class has no instances
  - An abstract class should, in general, omit implementations except for the methods that
    - Provide access to private data fields
    - Express functionality common to all of the subclasses

© 2006 Pearson Addison-Wesley. All rights reserved

Abstract Classes

- Abstract classes (Continued)
  - A class that contains at least one abstract method must be declared as an abstract class
  - A subclass of an abstract class must be declared abstract if it does not provide implementations for all abstract methods in the superclass

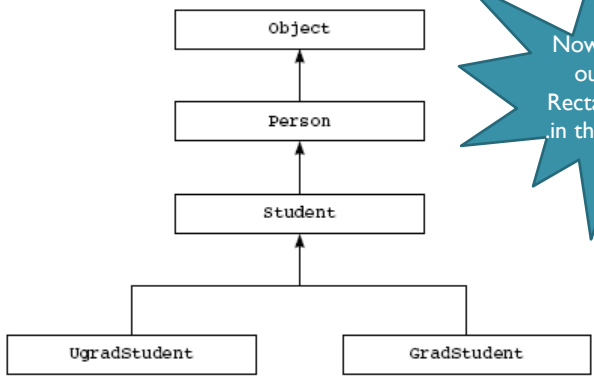
© 2006 Pearson Addison-Wesley. All rights reserved

Java Interfaces Revisited

- A Java interface
  - Specifies the common behavior of a set of classes
  - Common uses
    - Facilitate moving from one implementation of a class to another
      - A client can reference a class's interface instead of the class itself
    - Specify behaviors that are common to a group of classes

© 2006 Pearson Addison-Wesley. All rights reserved

Abstract Classes



Now, lets draw our Shape, Rectangle, etc. . . .in this heirarchy

Figure 9-10

Sample class hierarchy

Summary

- A subclass inherits all members of its previously defined superclass, but can access only the public and protected members
- Subclasses and superclasses
  - A subclass is type-compatible with its superclass
  - The relationship between superclasses and subclasses is an is-a relationship
- A method in a subclass overrides a method in the superclass if they have the same parameter declarations

© 2006 Pearson Addison-Wesley. All rights reserved

Summary

- An abstract method in a class is a method that you can override in a subclass
- A subclass inherits
  - The interface of each method that is in its superclass
  - The implementation of each nonabstract method that is in its superclass
- An abstract class
  - Specifies only the essential members necessary for its subclasses
  - Can serve as the superclass for a family of classes

© 2006 Pearson Addison-Wesley. All rights reserved