CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture I (W5L2)

Analysis: Creating the Class Diagram Part 3

Announcements

- Weekly Drop-in:
  - 16:30-17:30 in CS1-028, starting Mon. Week 6
- Weekly Q+A Session:
  - 10:00-11:00 in ICT Lab, starting Wed. Week 6
- SSF CD will be available in Short Loans from Tue. Week 6.

Analysis: Classes

Visibility

- UML uses **+**, **#**, **-**, **and** **~** to distinguish between public, protected, private, and package members.
- If inheritance is private, the derived class does not have access to private members of superclass, and all protected and public members inherited become private.
- When a member is classified as protected, visibility is restricted to the members of its own class, but this visibility can be inherited.

Analysis: Classes

Standard Class Stereotypes

- An interface specifies a set of operations. It may have attributes (usually constants), but does not define an implementation for the operations. An interface class is denoted by <<interface>>
  - Note: interface could not specify attributes in UML 1.x
- <<type>>: a class which has this stereotype is similar to an UML 1.x interface except that it can specify attributes as well as operations. No implementation supporting the declaration of operations.
- <<implementation>>: defines the implementation of its operations and attributes.

- Any object has one implementation class, and can have many types and match several interfaces.

Topic: Interfaces

- In any transaction, what assumptions can the client make about the service provider, and vice versa
- Document these assumptions in contracts
- Must be legally enforceable
- Software engineering uses interfaces.
- Documents the operations that a client can depend on (assume), and the obligations of the service provider.
  - Some elements of the interface are enforcable, specified in programming language constructs.
  - Others are currently not enforceable but should be documented in text

Topic: Interfaces

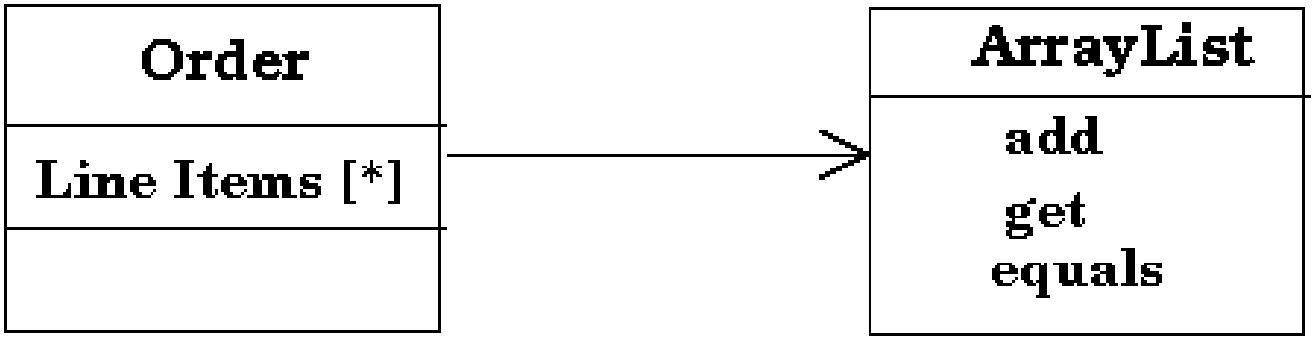
- An interface specifies some operations of some model element such as a class, which are visible outside the element.
  - Need not specify all operations supported, only those that are relevant to client developers.
  - Same element can have many interfaces.

- Design by Contract: Betrand Myers
  - Eliminate need for defensive proggramming i.e. Extensive error handling
  - Instead, specify pre and post conditions

- What about inherited pre and post conditions
  - “Demand no more, promise no less” - than what is specified in the contract/interface between client and server.
  - Precondition cannot be stronger – demand no more
  - Post condition cannot be weaker – promise no less.

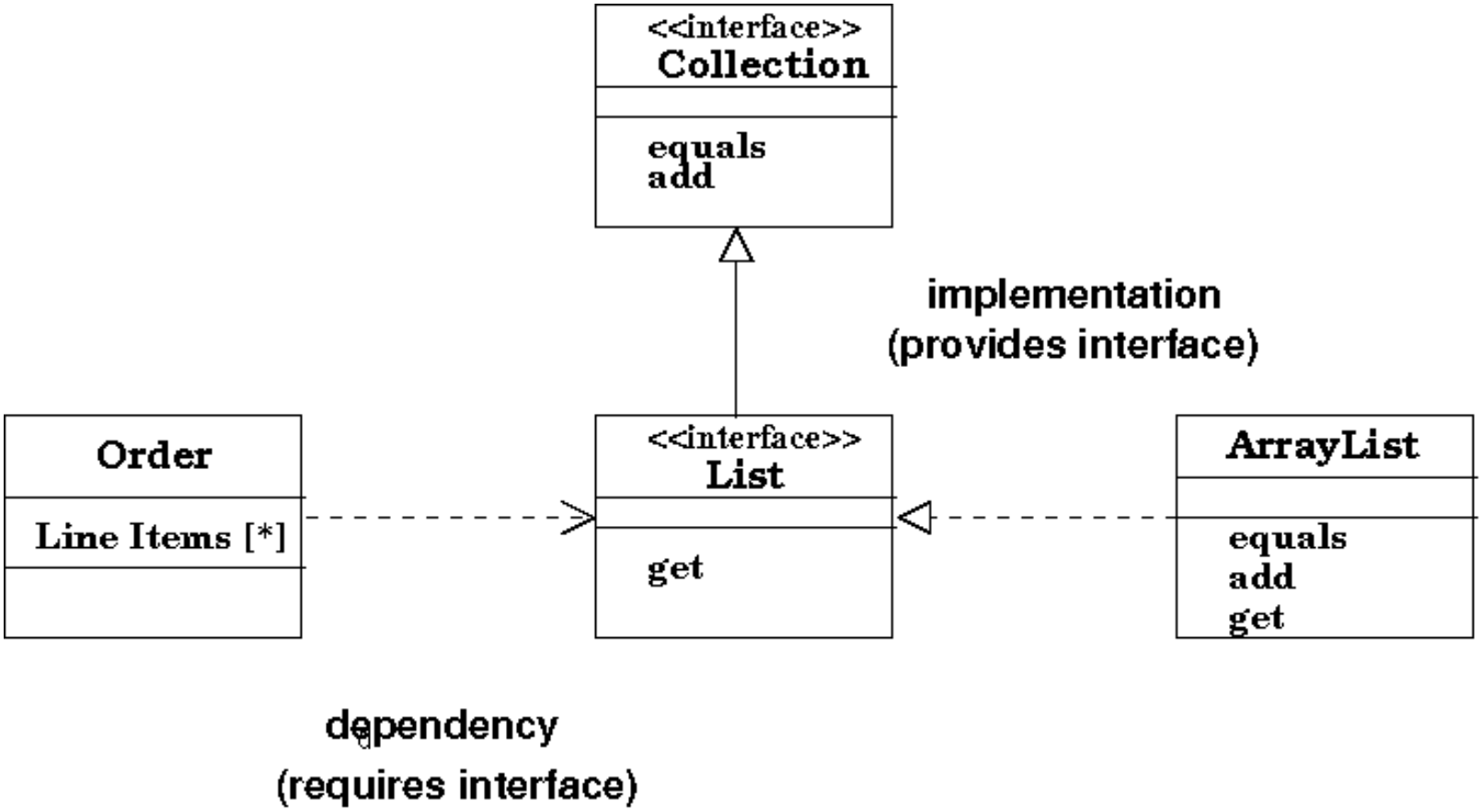
Topic: Interfaces

- Example from pages 67-69 of chapter 5 in
  - Martin Fowler. UML Distilled. Addison-Wesley. 2004.
- Based on pages 17-18 of chapter 1 in
  - Gamma et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
- An example of "programming to implementation" shown below.



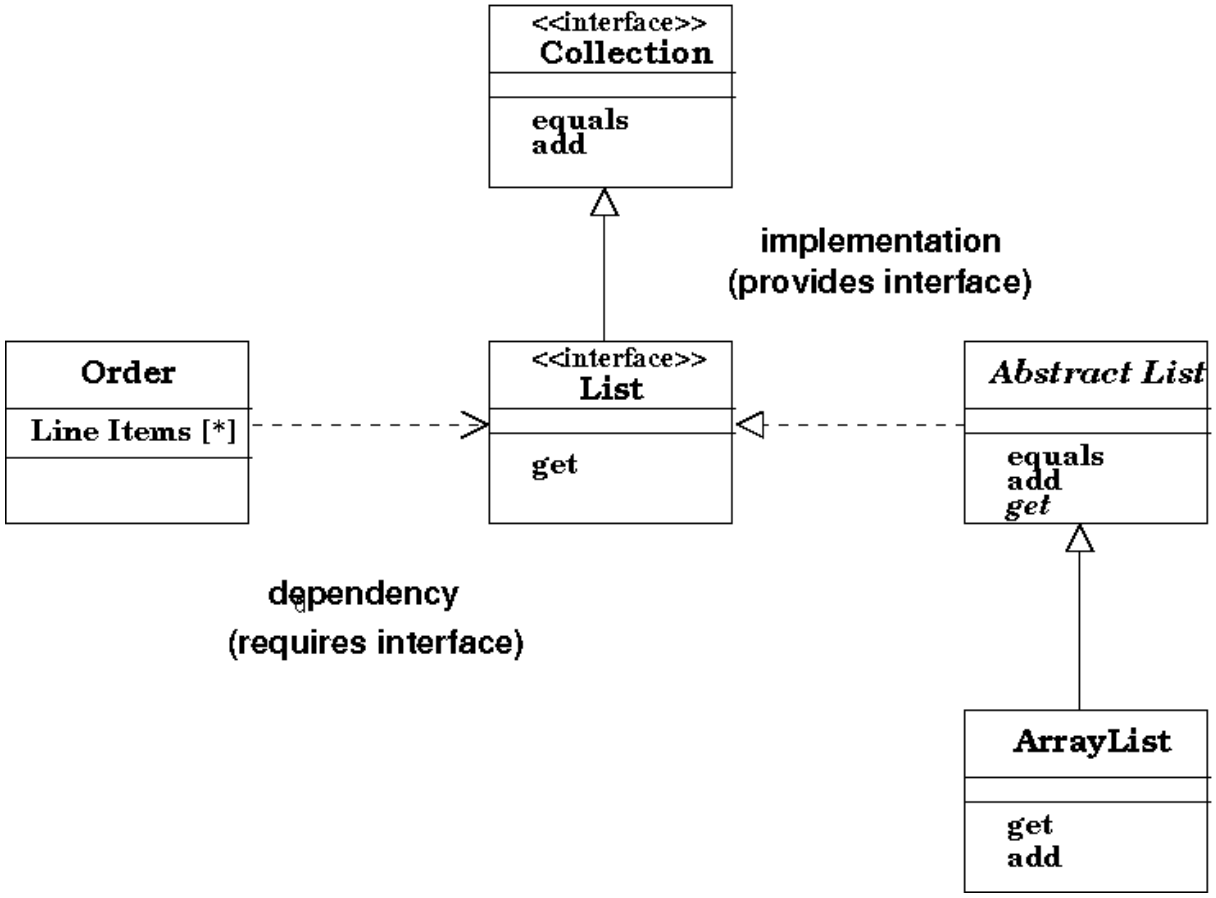
Topic: Interfaces

- An example of "*programming to interfaces*" shown below.



Topic: Interfaces

- Same as previous slide , except that an abstract class has been added.



Topic: Interfaces

- Slide 6
  - **ArrayList** lineItems = new ArrayList()
- Slide 7
  - **List** lineItems = new ArrayList()
  - Slide 8 adds an abstract class to slide 7
- Slide 7 minimises dependency between client and server
  - Minimises cost of change if changing server later
  - Supports modifiability!
- Why?

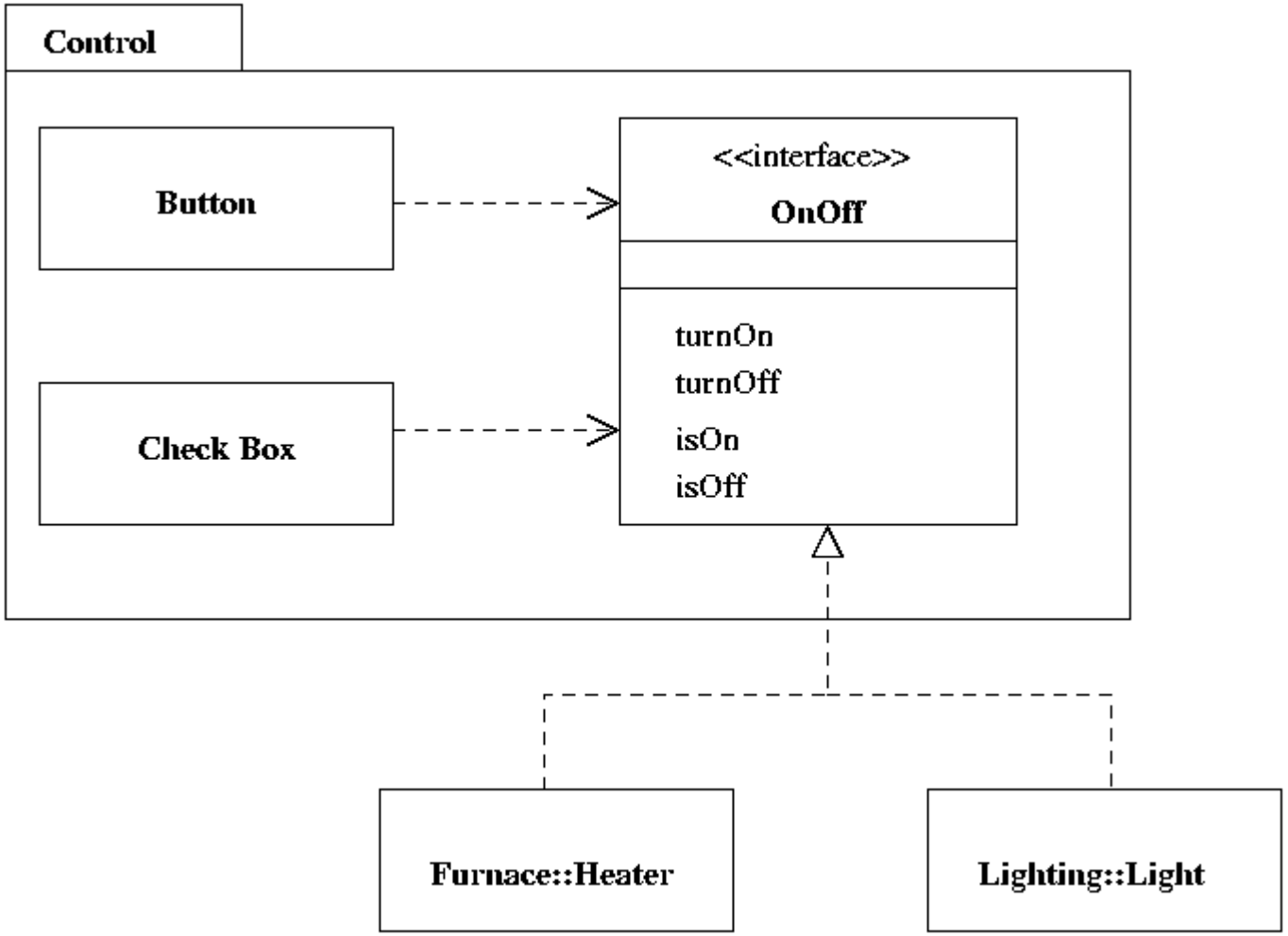
Topic: Interfaces

- Why don't we simply have Order use ArrayList directly (first option)?
- By programming to the intetrface List (second option), make it easier to change the implementation later
- By programming to the interface rather than implementation, avoid having to change all the code later if better implementation available
- Because the type of lineItems is declared as List, no other part of the Order class is dependent on ArrayList
- Pragmatic wrinkle: programming this concept strictly introduces a dependency from lineItems to ArrayList, ignored in practice.

- **PROGRAM TO INTERFACES, NOT IMPLEMENTATION!**

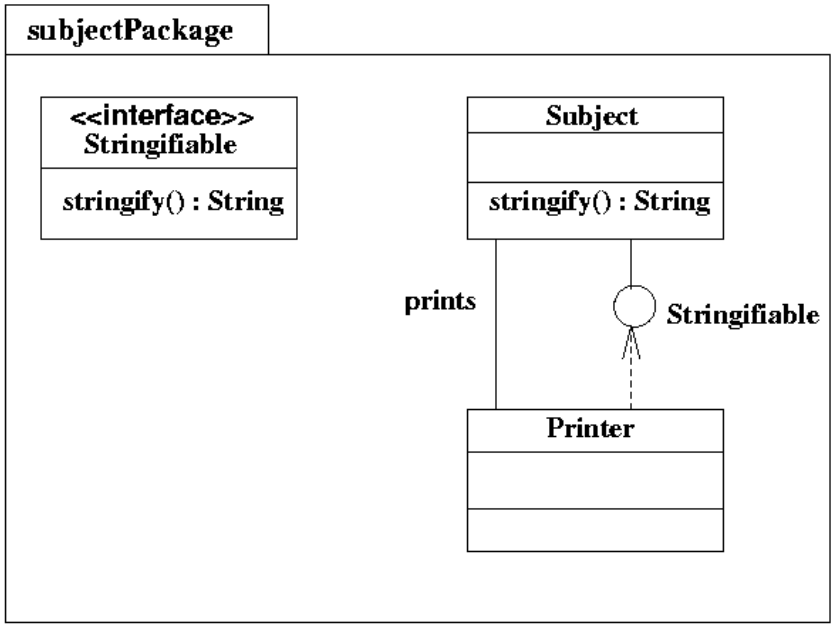
Topic: Interfaces

- Fowler's Pattern of Separated Interfaces

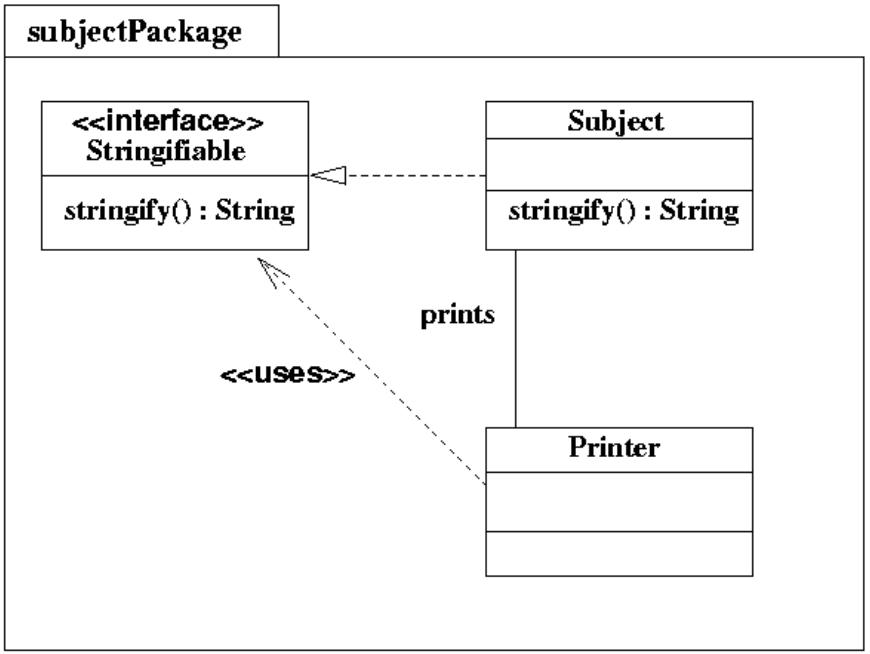


Analysis: Classes - Part I

- Interface Class in UML 1.x

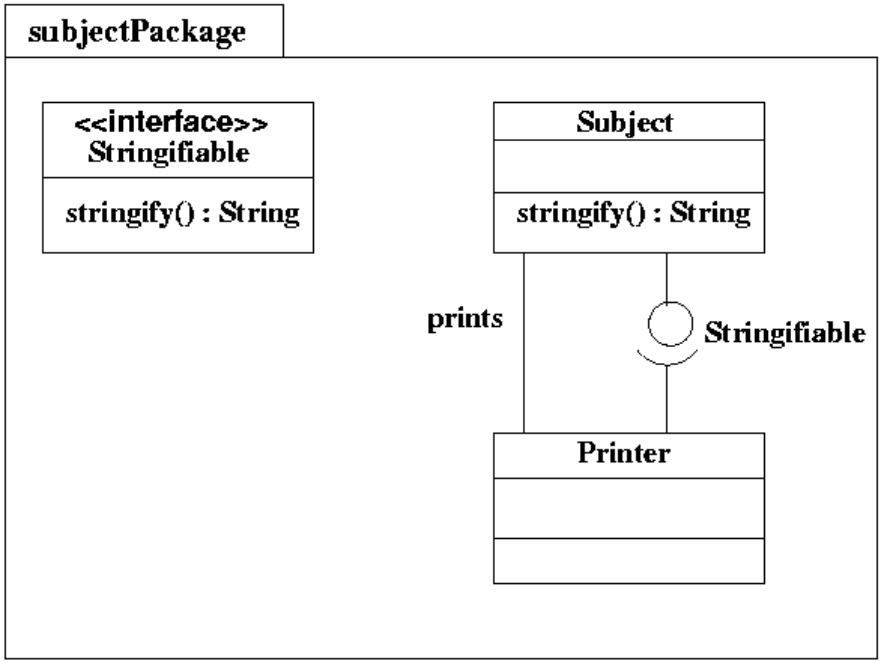


OR

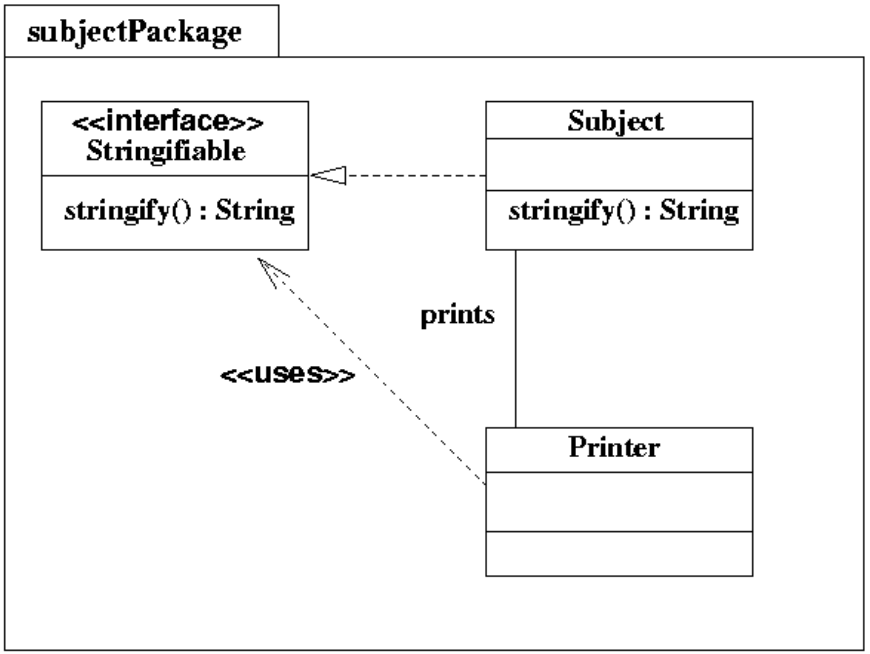


Analysis: Classes - Part I

- Interface Class in UML 2.0



OR



Analysis: Classes - Part I

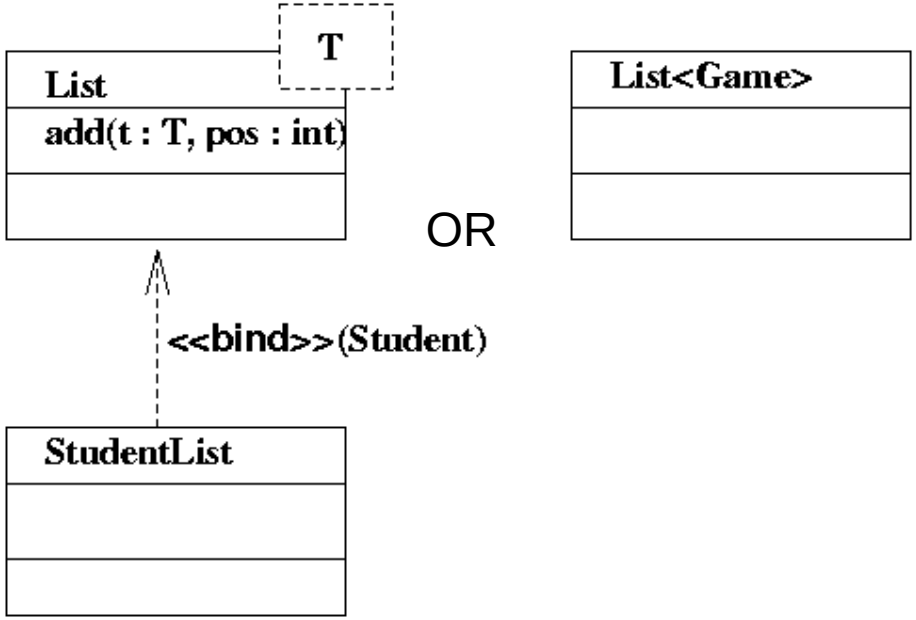
Abstract Classes:

- Shown using the property **{abstract}**.
- In UML, every class has a Boolean property isAbstract
- May have implementations defined for some of its methods.
- BUT for at least one of its methods, no implementation defined.
- Cannot declare an object as an instance of this class.
- An abstract class in which none of the operations have an implementation, is effectively an interface model element.
- A subclass inherits method(s) from the abstract class and implements the interface.

Topic: Parameterised Classes (Templates)

Parameterized Classes

- Also known as a template.
- *Example: List<T>*, which, given a class C to substitute for the formal parameter *T*, describes the class of lists of C objects.
- e.g. *List<Student>* and *List<Game>*.
- Two ways to show that a class is a result of applying a parameterised class to an argument.



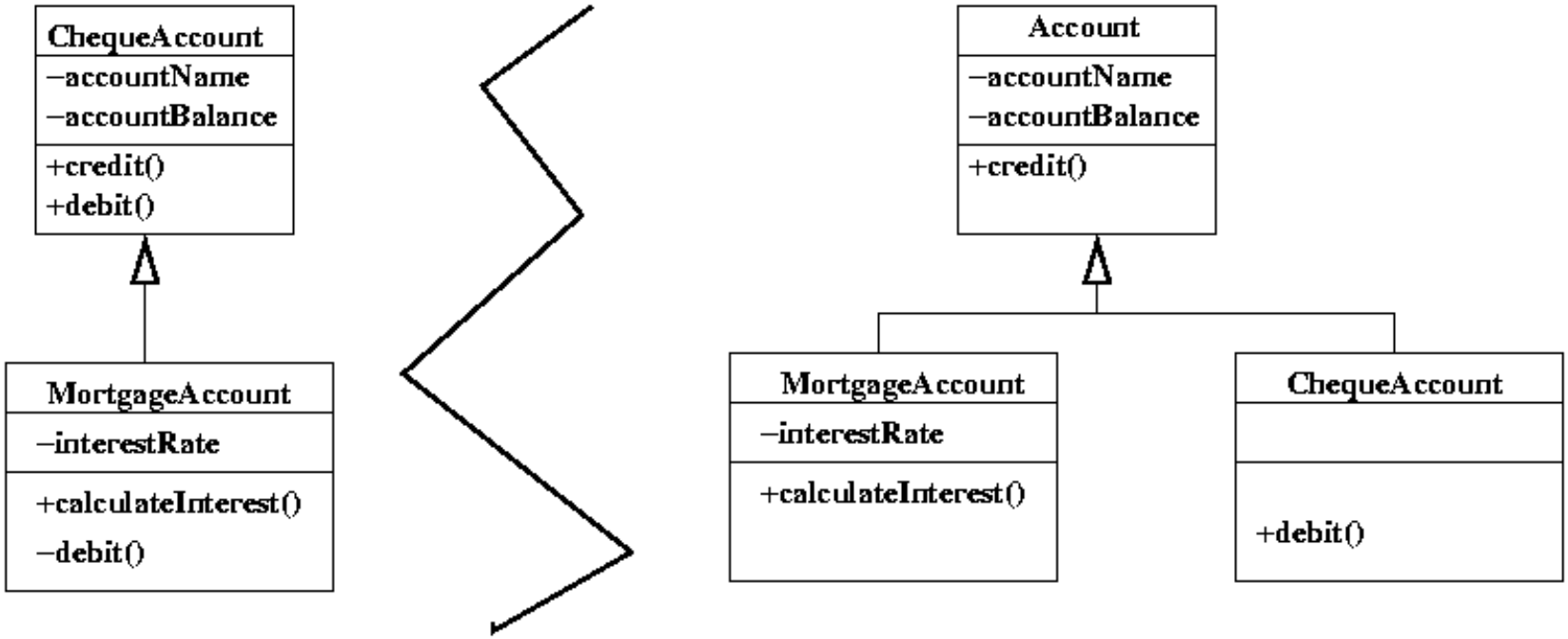
Topic: Parameterised Classes (Templates)

- One explicitly diagrams dependency. Implicit in other.
- Not all languages support this, sometimes known as genericity. C++ does, what about Java?.
- Irrespective of language, should use parameterised classes whenever possible.

- A stereotype is UML's way of attaching extra classification to model items, it is one of the ways that UML is made extensible. Stereotypes are predefined, and the set may be extended by user defined tags e.g. **<<persistent>>** could be defined within a software house.

Topic: Liskov Substitution Principle

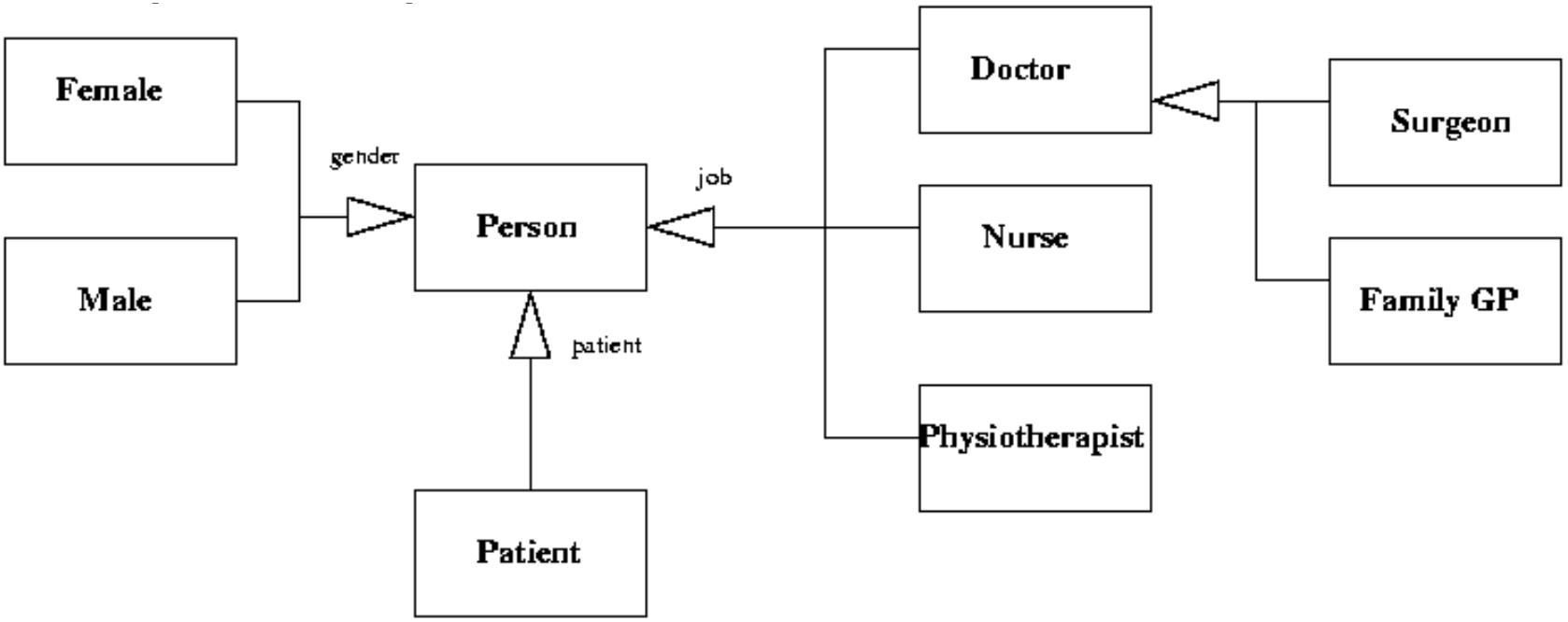
- Applicable to inheritance hierarchies.
- It sates that in object interactions, it should be possible to treat a derived class as if it were a base class.
- If the principle is not applied, then it may be possible to violate the integrity of the derived class.



Application of the Liskov Substitution Principle

Topic: Multiple and Dynamic Classification

- Classification refers to the relationship between an



Multiple Classification

Analysis: Classes, Part 1

dependency is .....

- Seen dependency between:
  - Interface class and its implementation.
  - Superclasses and subclasses.
  - Class and parameterised class.
- An association is not a dependency.
- An association between two classes represents the fact that objects of those classes have a real world relationship.
- A dependency is between the classes themselves, not between the objects of those classes.
- Usually preferable to explicitly represent the dependency.
- Can have dependencies between any two model elements.

10. Reading

- Chapter 6 in Bennett et al. or
- Chapter 7 and 8 in Stevens and Pooley