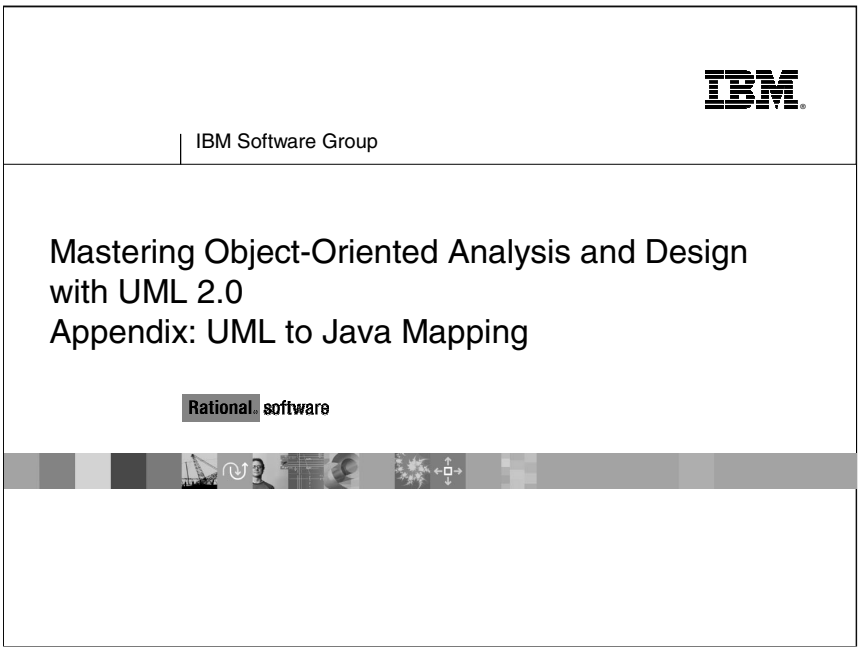
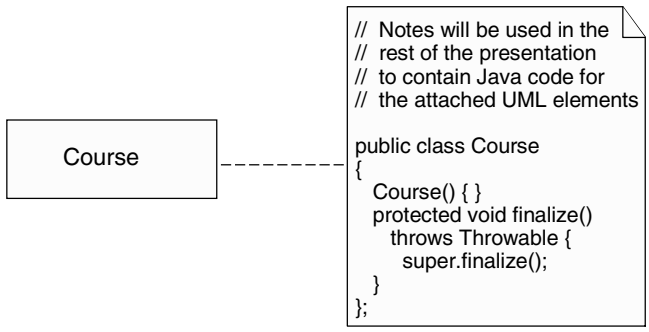
Mastering OOAD w/ UML 2.0 – Instructor Notes

Instructor Notes:



Instructor Notes:

Mapping Representation: Notes

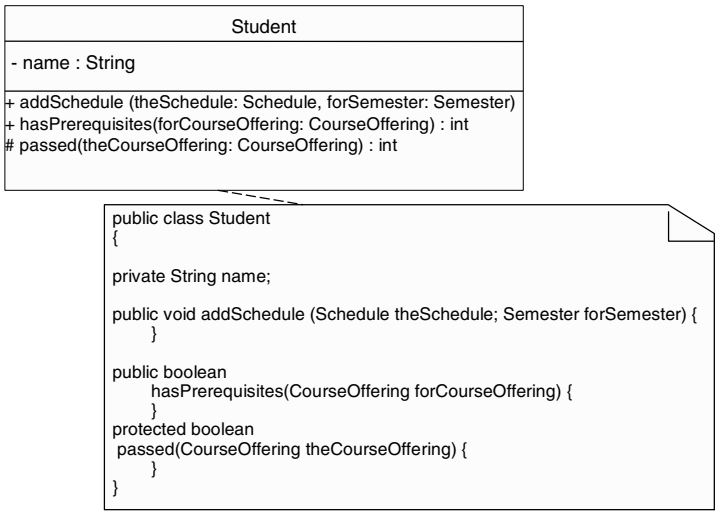


2

If you remember from earlier in the course, a note can be added to any UML element. It is represented as a 'dog eared' rectangle. The note may be anchored to a specific element(s) with a dashed line

Instructor Notes:

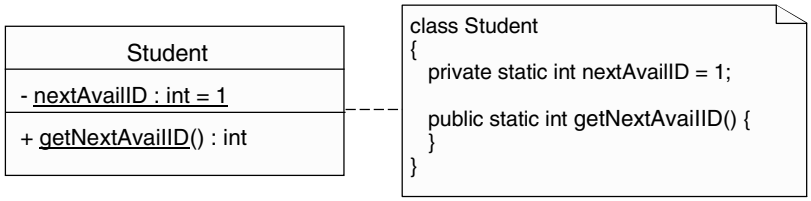
Visibility for Attributes and Operations



3

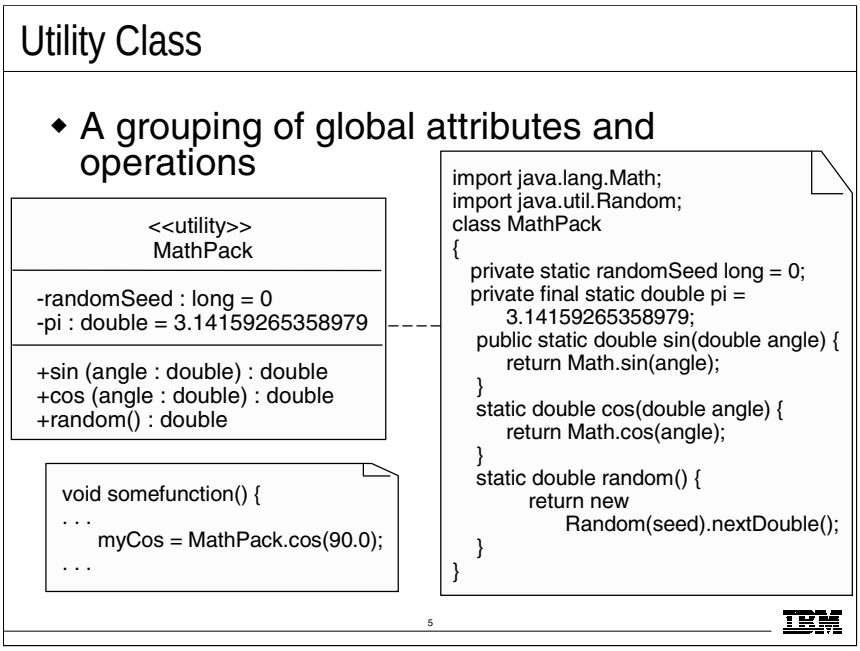
Instructor Notes:

Class Scope Attributes and Operations



4

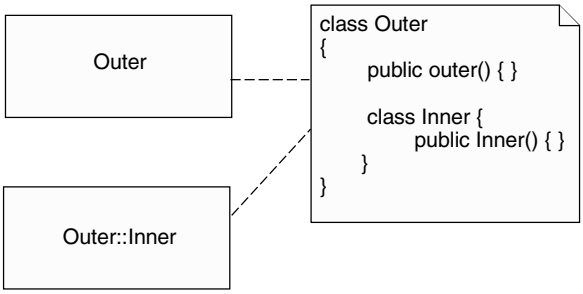
Instructor Notes:



Instructor Notes:

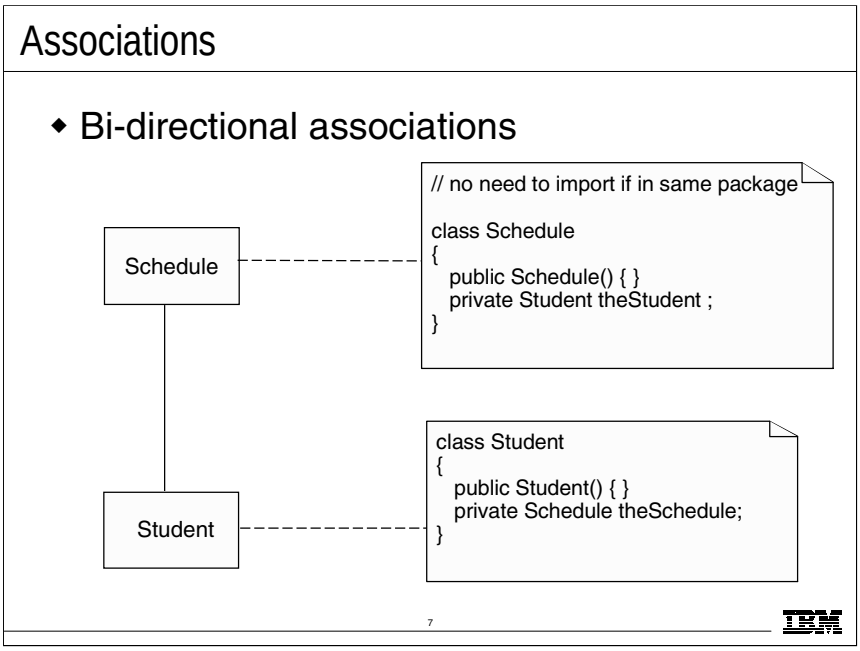
Nested Class

- Hide a class that is relevant only for implementation



6

Instructor Notes:

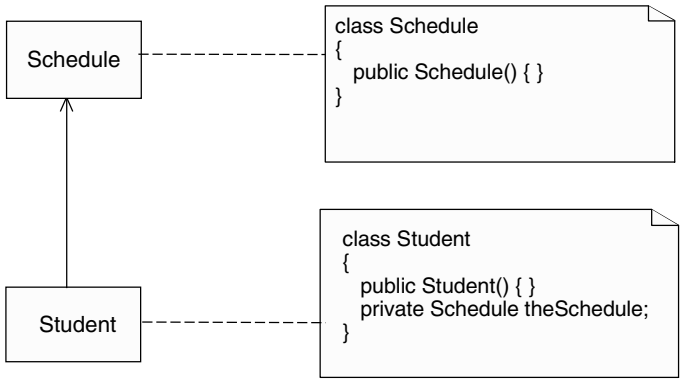


Classes declared in the same Java package can "see" one another.

Instructor Notes:

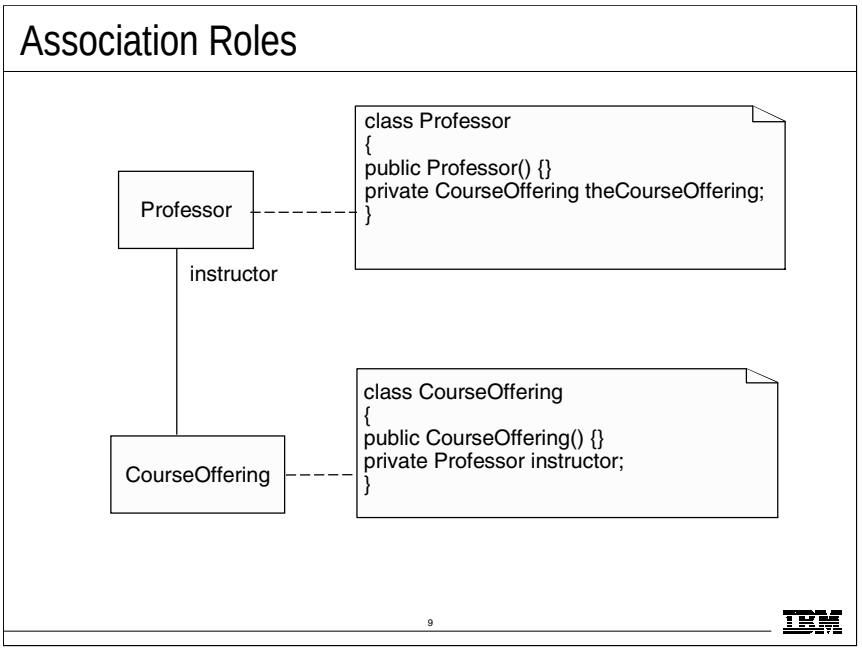
Association Navigability

- Uni-directional associations



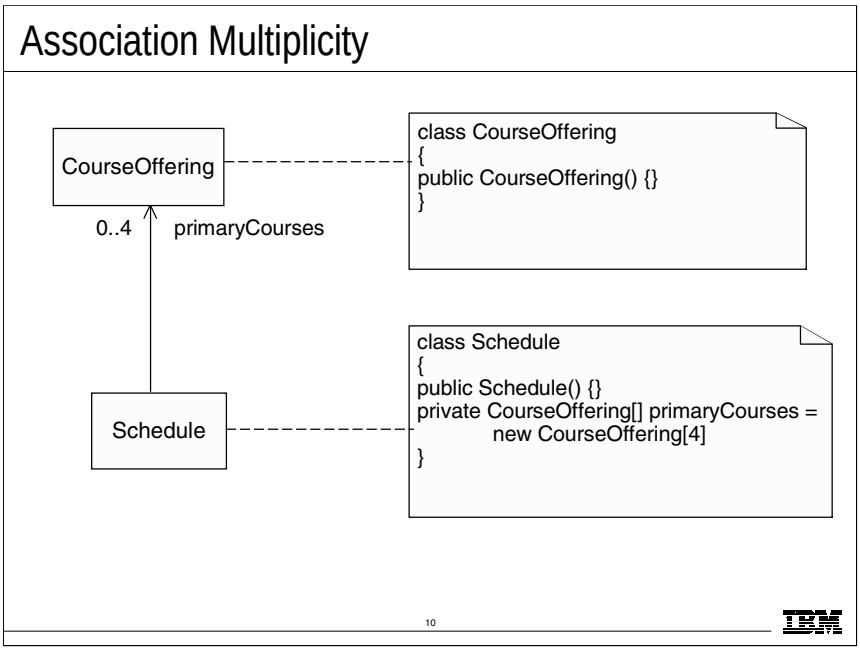
8

Instructor Notes:



Roles on the end of the association can add clarity.

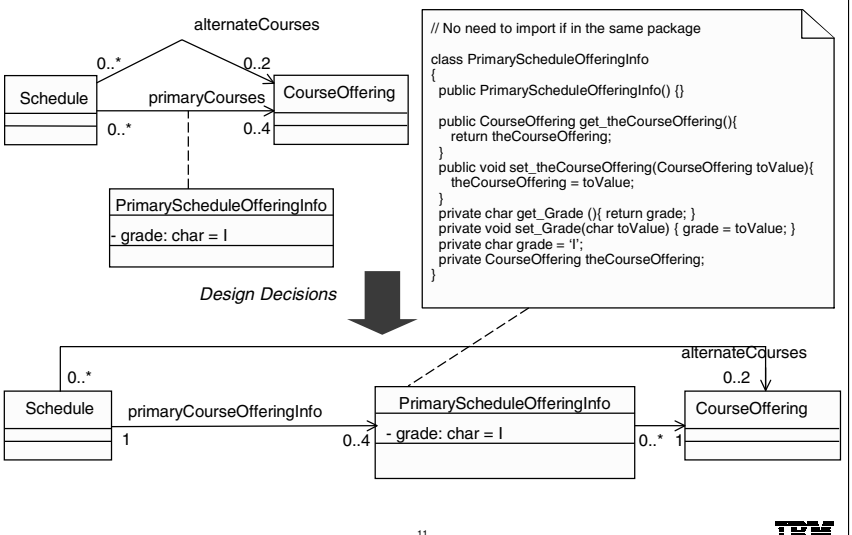
Instructor Notes:



Multiplicity is the number of instances of one class in relation to the other.

Instructor Notes:

Association Class

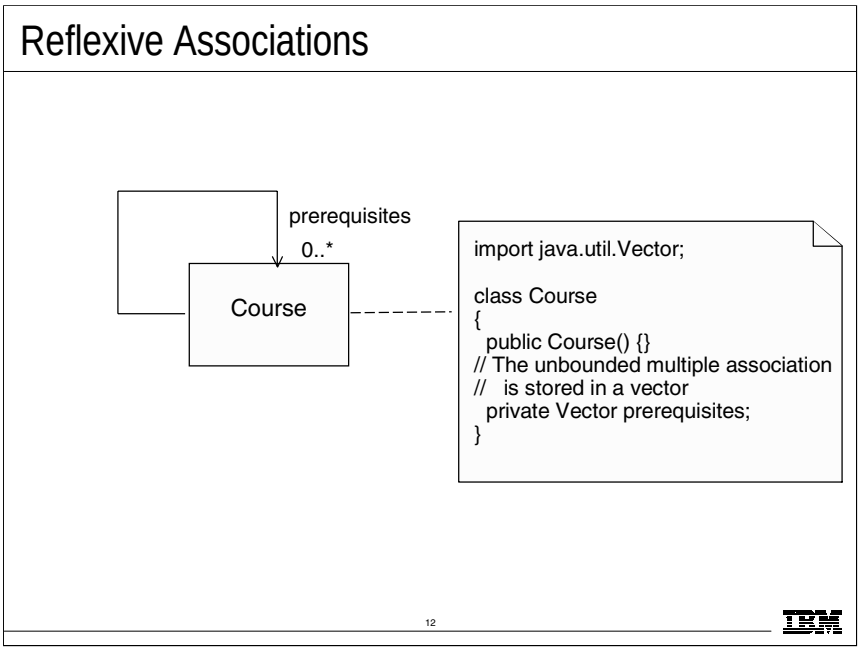


Remember, an association class is a class that is connected to an association. There is an instance of the association class for every instance of the relationship (for example, for every link).

During design, some decisions are made regarding navigation between the involved classes.

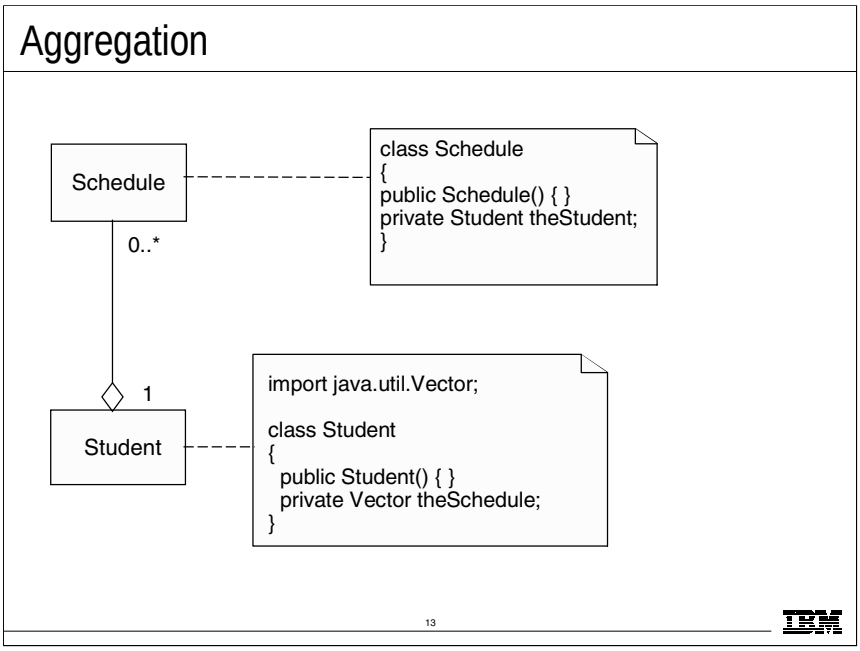
A subset of the class operations and attributes are shown above. For this example, we included a subset to demonstrate the UML construct we are emphasizing.

Instructor Notes:



A class may have an association with objects of the same type.

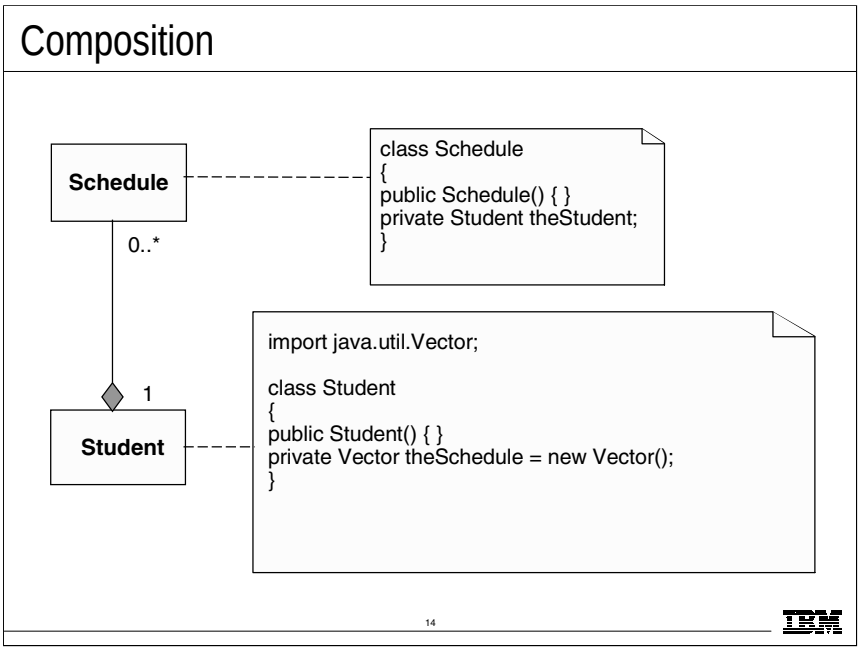
Instructor Notes:



Vector is a reusable list class available in the Java programming environment. It could be replaced by any other available list class.

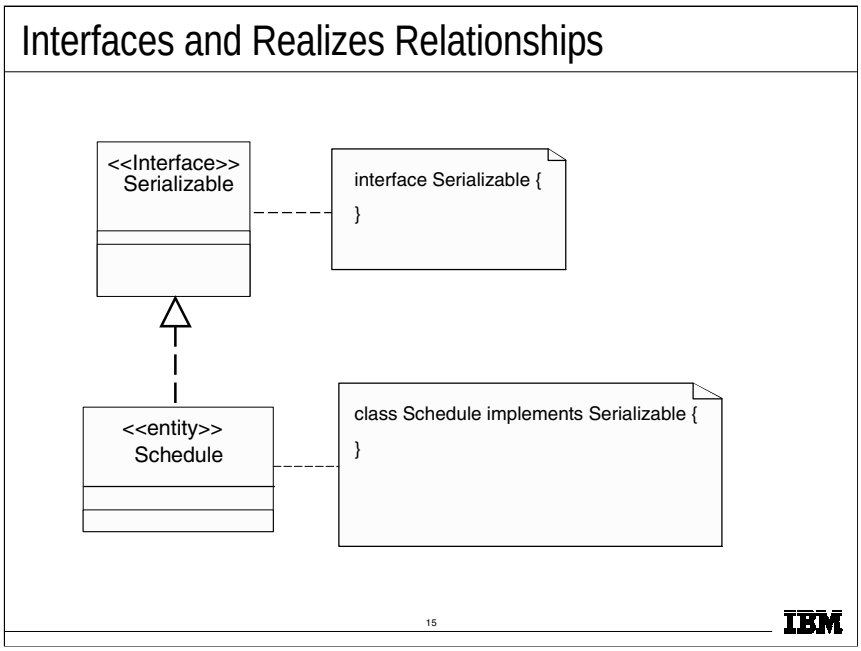
Java has no explicit construct for aggregation. In Java, the code for aggregation looks the same as it does for "vanilla" association.

Instructor Notes:



Java does not support containment by value.

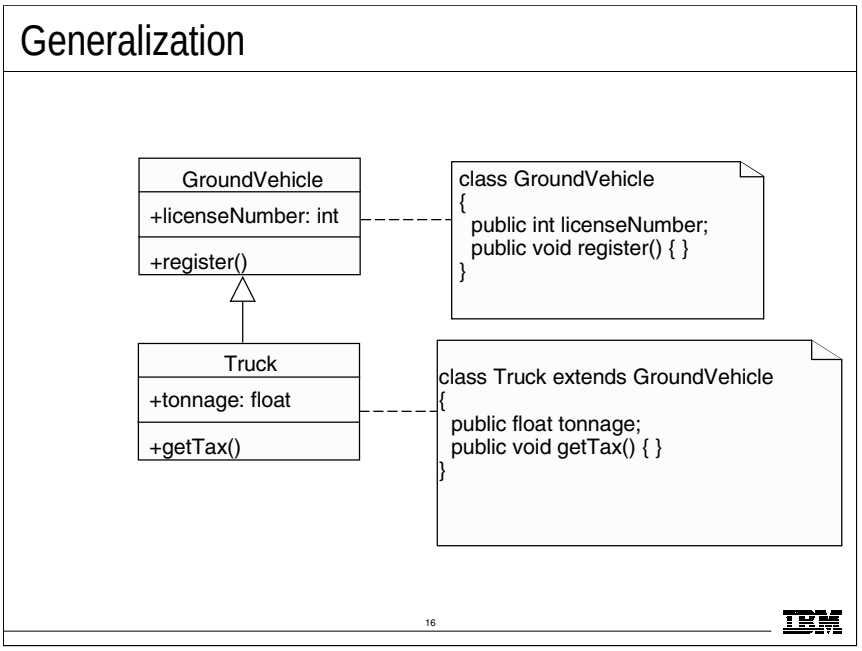
Instructor Notes:



Java has the notion of "implements" which translates to the realizes relationship in the UML. Java classes implement interfaces and they can implement as many interfaces as they see fit. Interfaces can extend other interfaces.

In Java, interfaces may have attributes defined for them. The OMG definition (UML 2) of interface states: "An interface is a named set of operations that characterize the behavior of an element."

Instructor Notes:

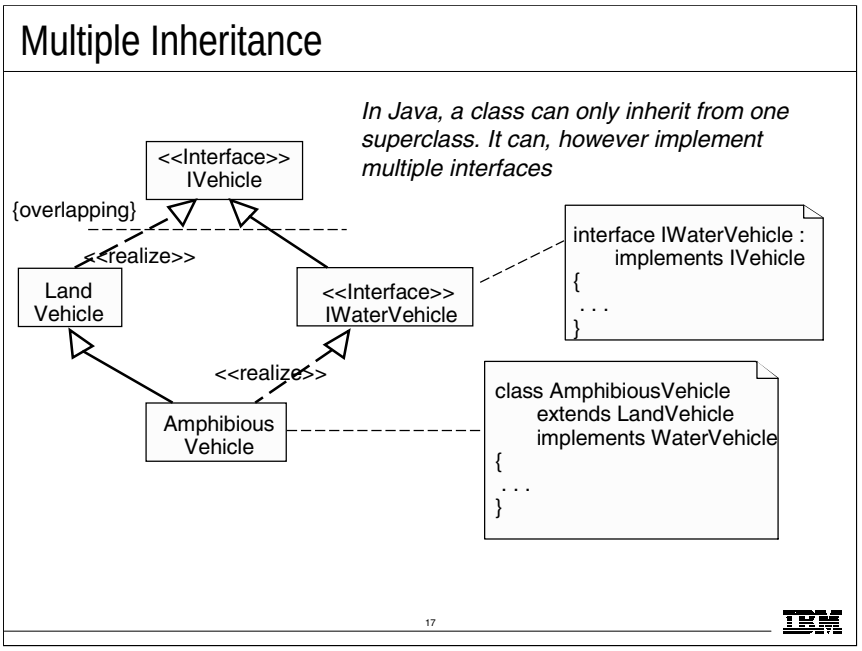


Remember generalization is a "is-a" or "kind-of" association. It is used to represent generalization / specialization.

Generalization is modeled as an open triangular arrowhead pointing to the base on the base class end.

Java has the notion of "extends" which translates to the generalization relationship in the UML. Java classes can extend ONE other class. Interfaces can extend other interfaces. Java interfaces are discussed on a later slide.

Instructor Notes:

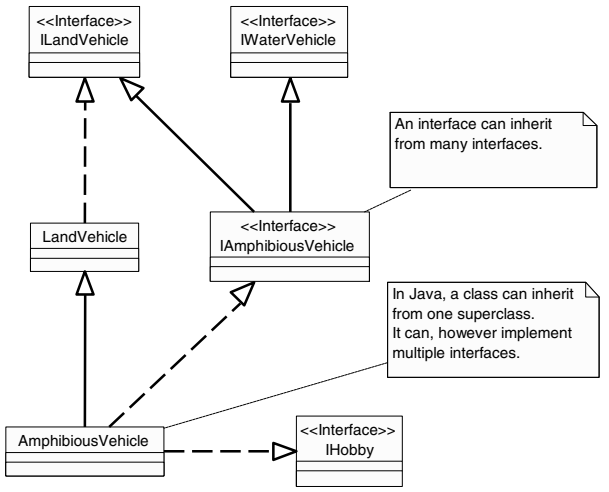


In Java, a class can only inherit from ONE superclass. However, a class can realize multiple interfaces.

Remember, subclasses that are not mutually exclusive can be annotated with the UML {overlapping} constraint. This supports multiple inheritance.

Instructor Notes:

Multiple Inheritance (continued)



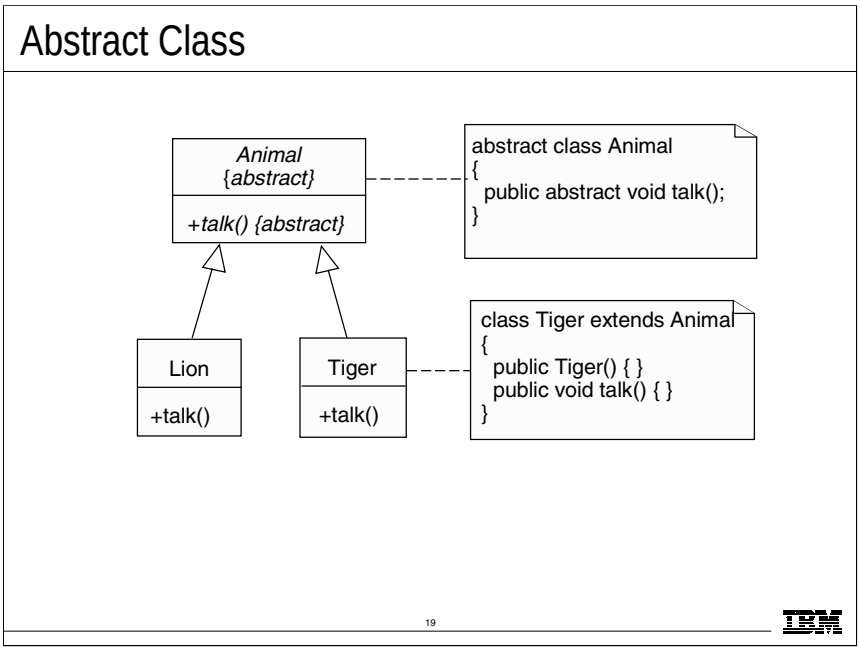
18

Java supports multiple inheritance between interfaces.

In Java, an interface CAN inherit from multiple interfaces.

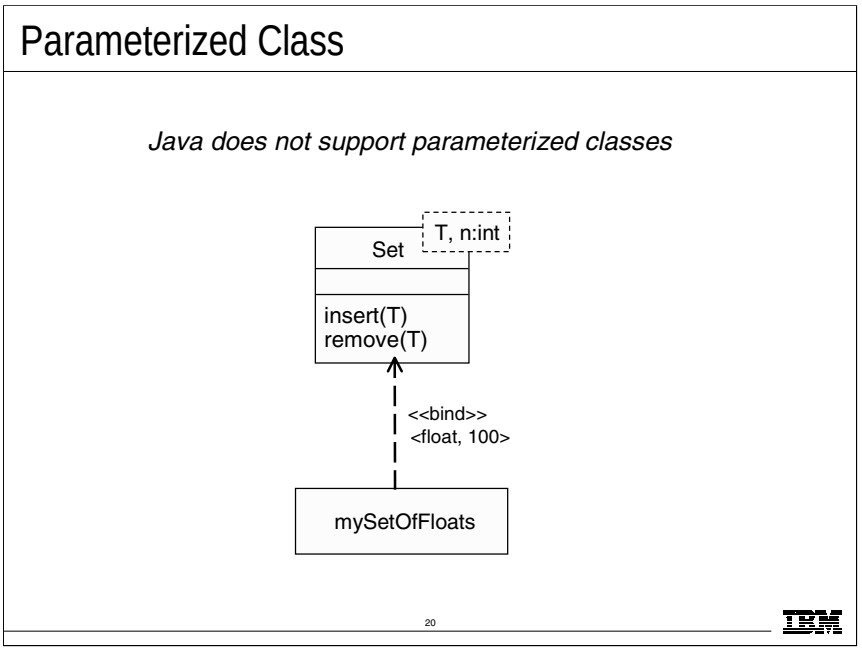
Multiple inheritance of interfaces overcomes the “weakness” of Java regarding true multiple inheritance.

Instructor Notes:



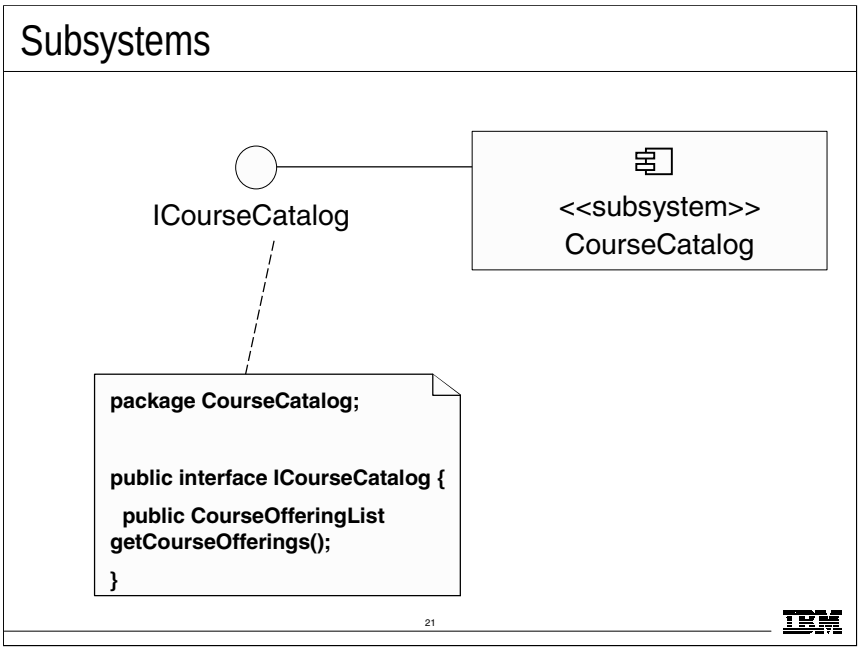
Remember, an abstract class is a class for which no instances are created.

Instructor Notes:



Java does not provide support for parameterized classes.

Instructor Notes:



As discussed within the OOAD course, subsystems are the Design Model representation for components.

Some aspects of UML subsystems can be implemented using Java packages. Java packages have a one-to-one correspondence to UML packages. Packages in Java also correspond to directories.

Packages are declared at the top of the file. All classes defined within the file are considered part of the specified package. All classes defined within the same package can “see” each other automatically. If the package statement is omitted (i.e., no package is specified), the file contents are considered to be in the "default package" (for example, the root package), and all other classes for which a package was not specified can “see” the classes defined within the file.

There can be only 1 public class per file (you can have inner classes in the file as well, but only 1 "top level" class). The name of the file must be the same as the name of the public class.

These restriction are what hinder Java’s support for subsystems. In the UML, the mapping between interfaces and subsystems is many-to-many (subsystems can realize one or more interfaces; interfaces can be realized by one or more subsystems). In Java, the mapping is always one-to-one.

Note: CourseOfferingList is assumed to exist in a separate common package. The import statement has been excluded from the code fragment.

Instructor Notes: