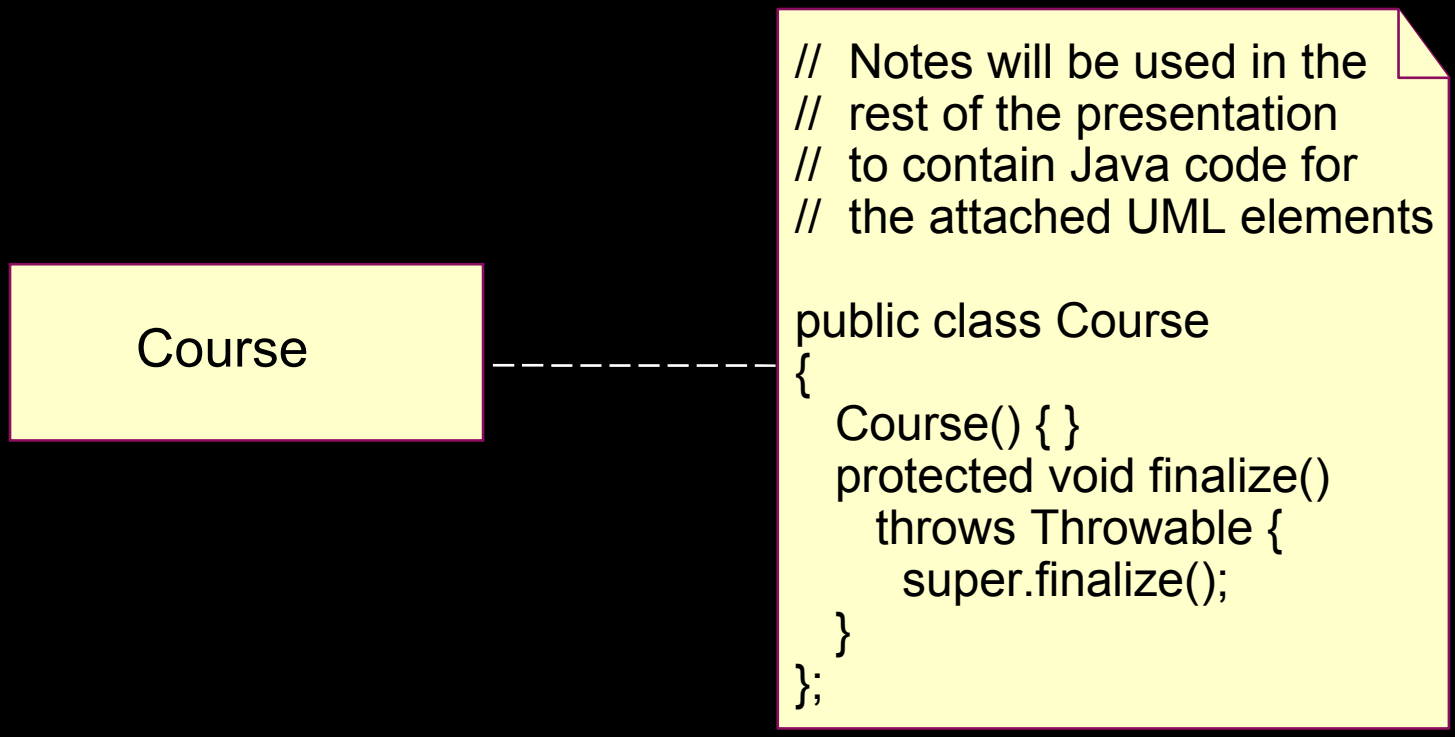
IBM Software Group

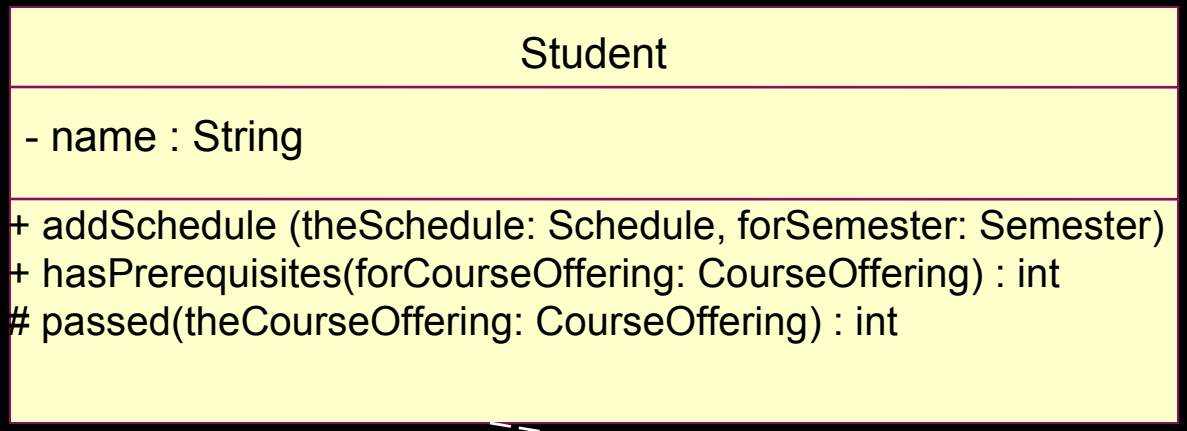
Mastering Object-Oriented Analysis and Design with UML 2.0
Appendix: UML to Java Mapping

Rational. software

Mapping Representation: Notes

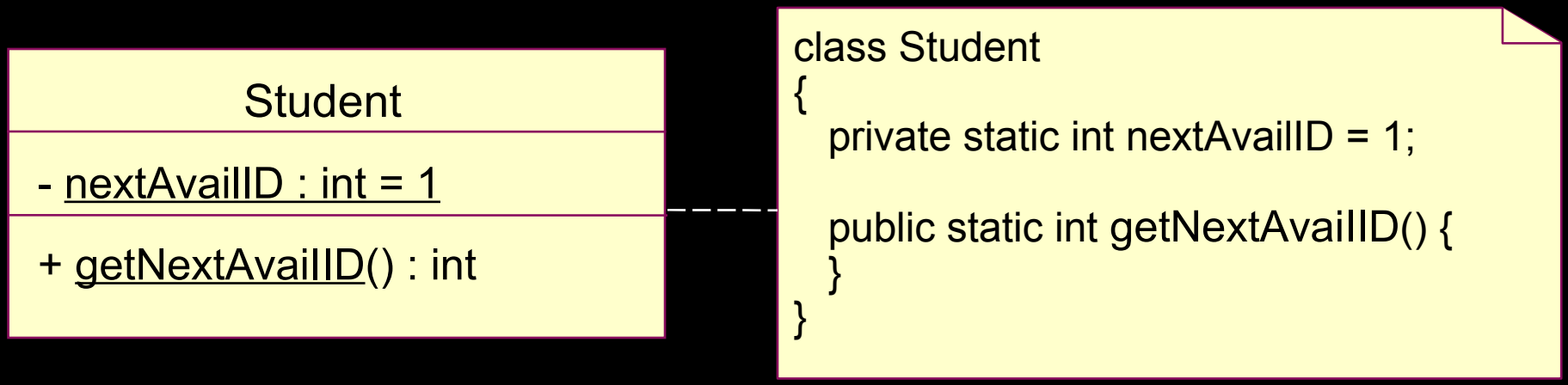


Visibility for Attributes and Operations



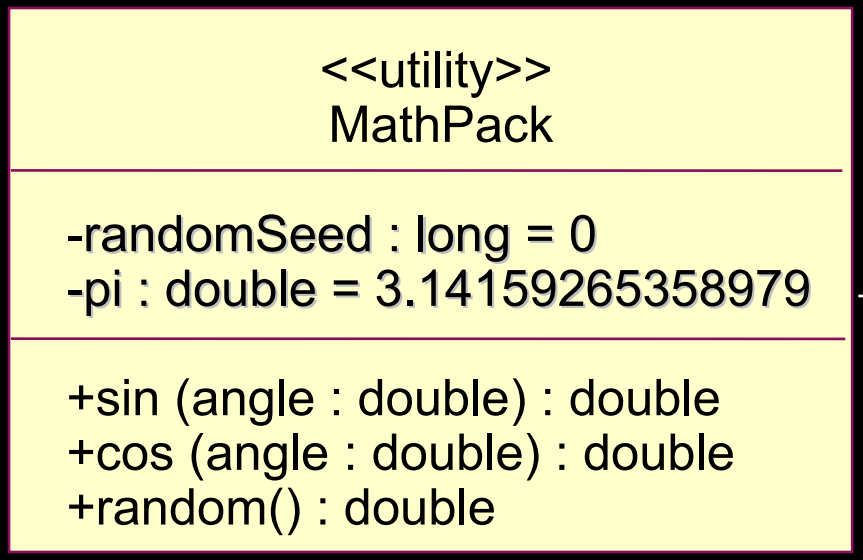

```
public class Student
{

private String name;

public void addSchedule (Schedule theSchedule; Semester forSemester) {
        }

public boolean
        hasPrerequisites(CourseOffering forCourseOffering) {
        }
protected boolean
 passed(CourseOffering theCourseOffering) {
        }
}
```

Class Scope Attributes and Operations



Utility Class

- A grouping of global attributes and operations

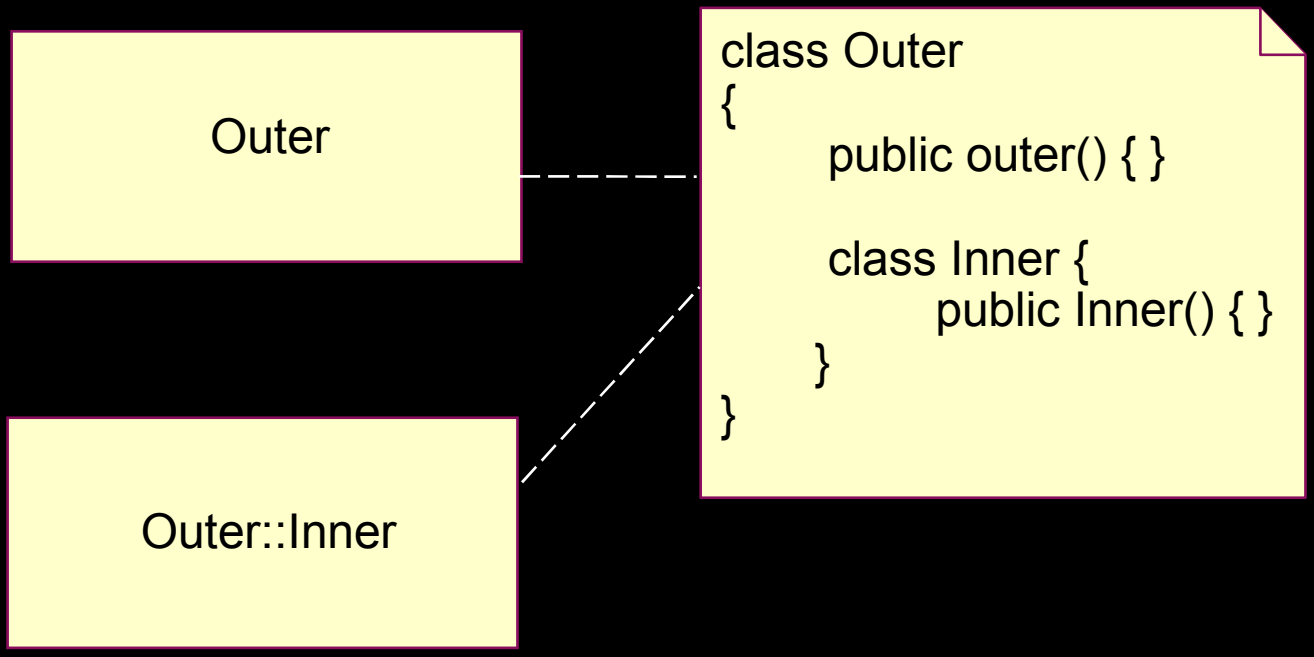



```
void somefunction() {
. . .
      myCos = MathPack.cos(90.0);
. . .
```

```
import java.lang.Math;
import java.util.Random;
class MathPack
{
   private static randomSeed long = 0;
   private final static double pi =
         3.14159265358979;
    public static double sin(double angle) {
         return Math.sin(angle);
    }
    static double cos(double angle) {
         return Math.cos(angle);
    }
    static double random() {
              return new

Random(seed).nextDouble();
    }
}
```

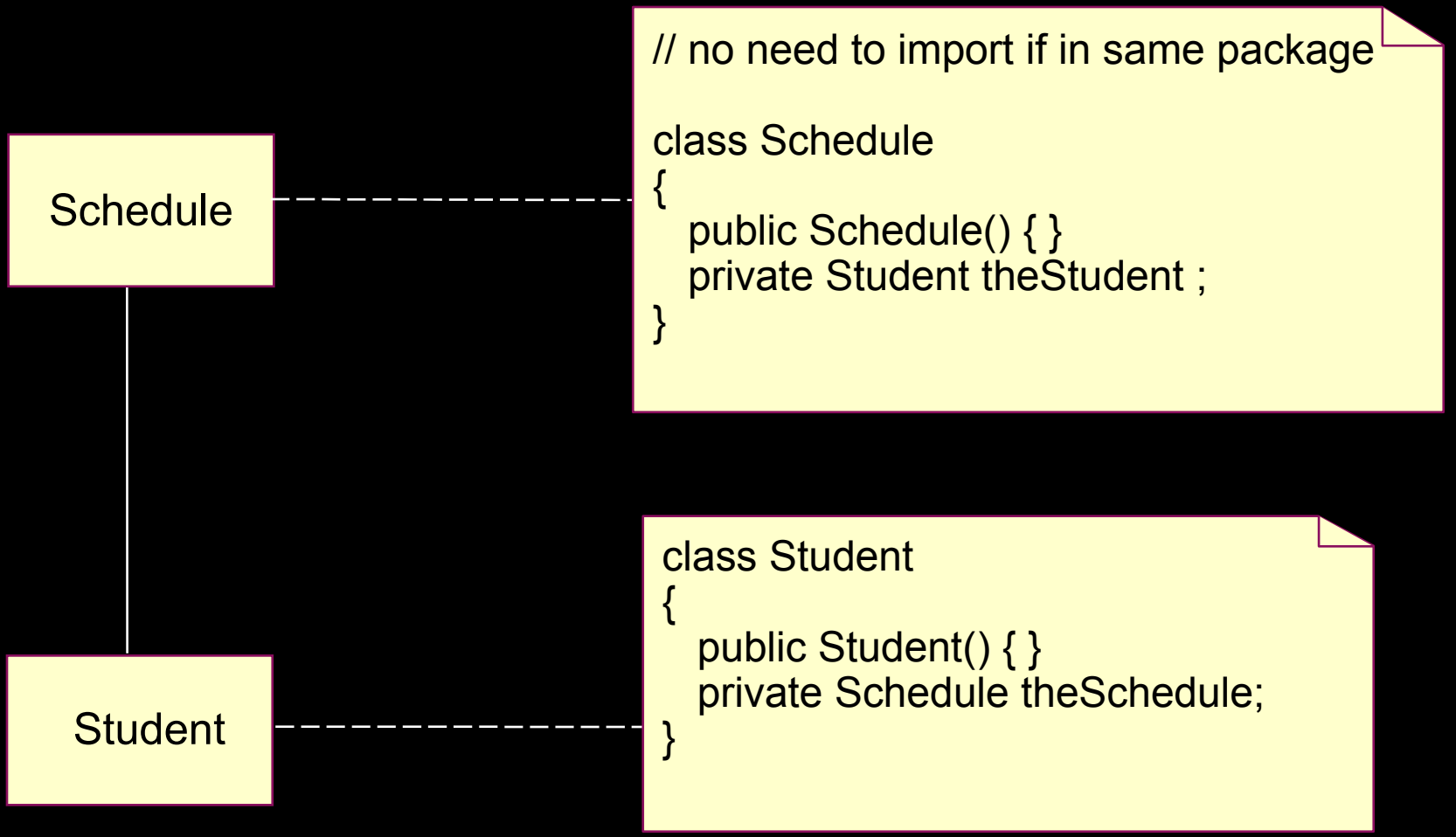
Nested Class

- Hide a class that is relevant only for implementation



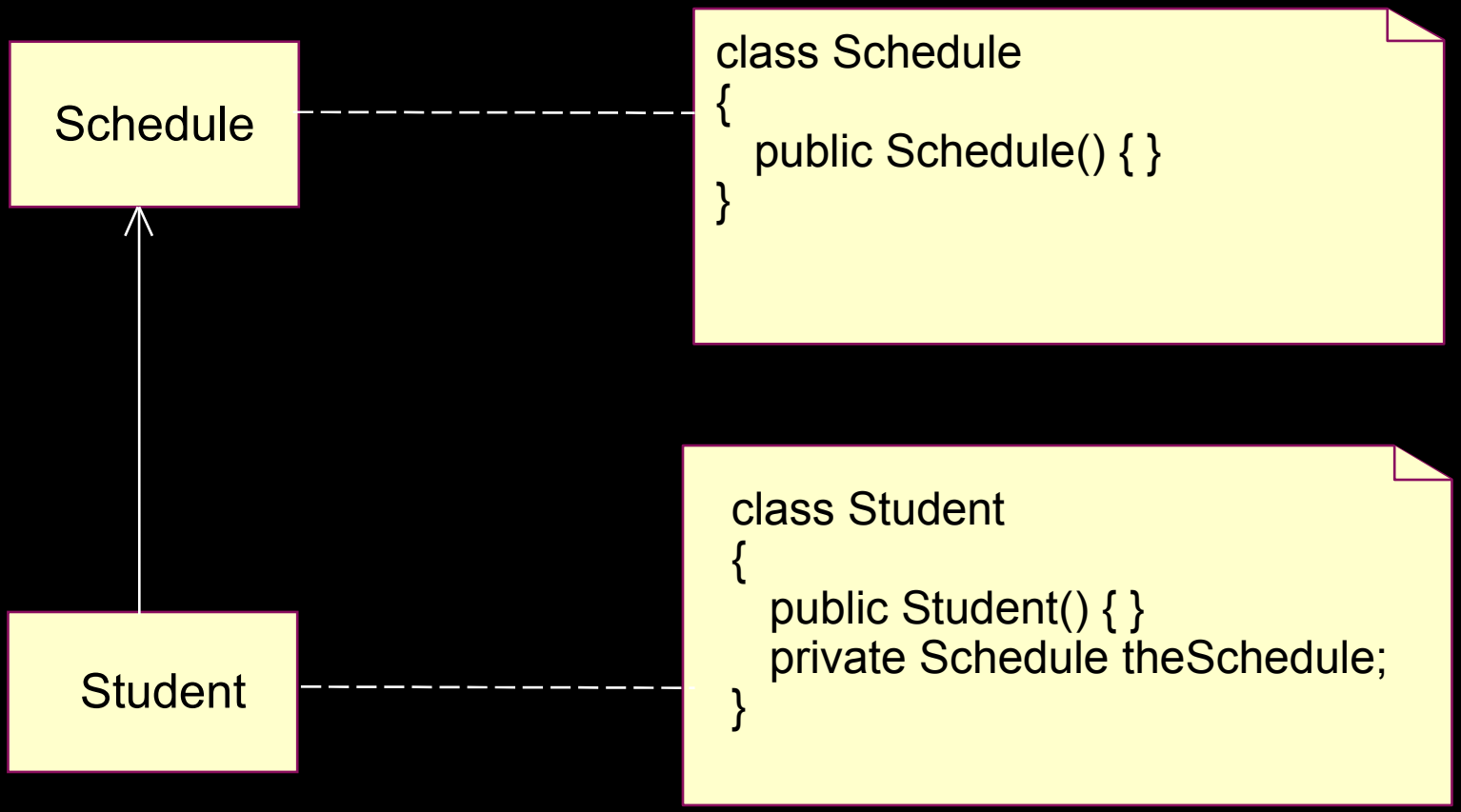
Associations

- Bi-directional associations

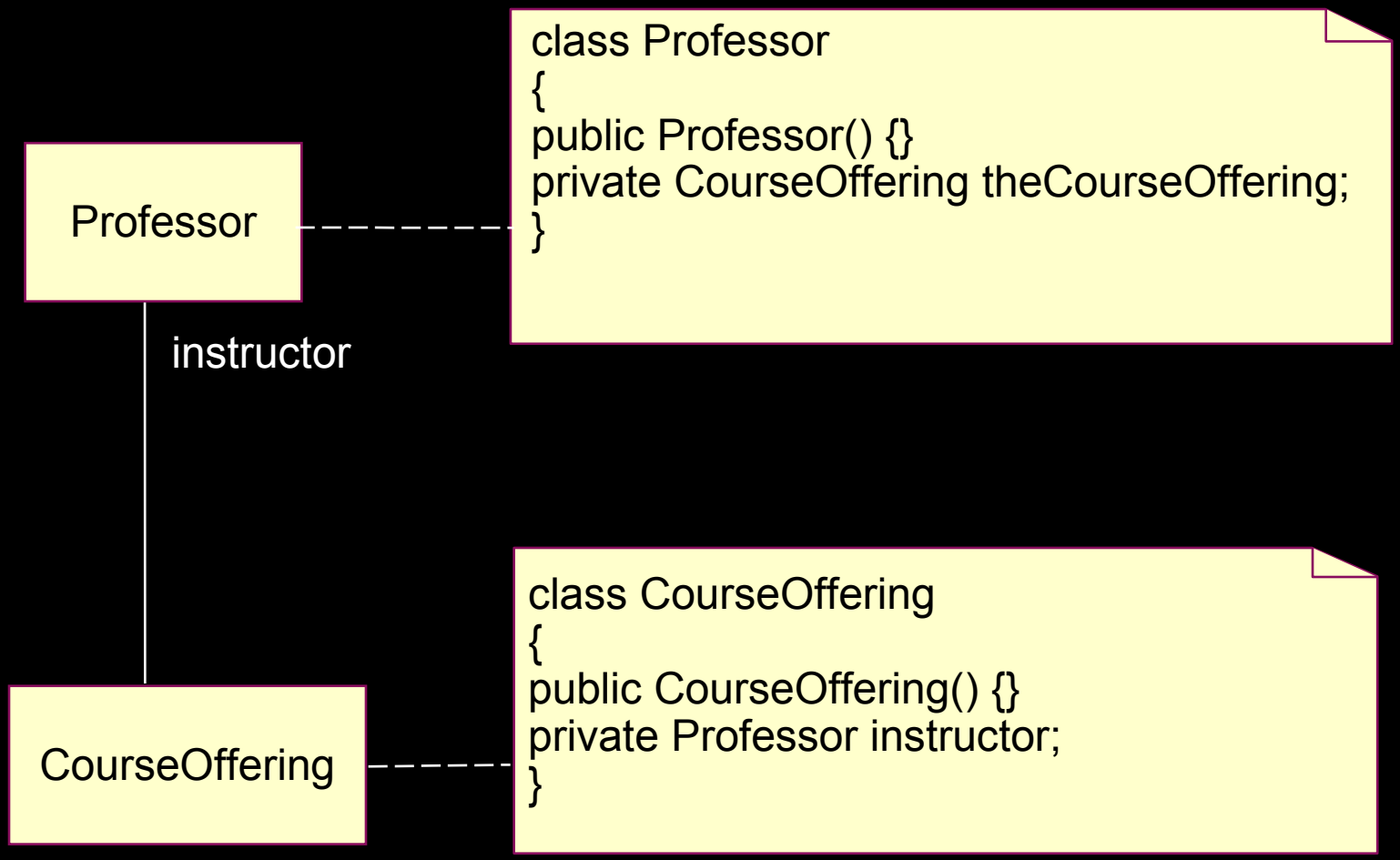


Association Navigability

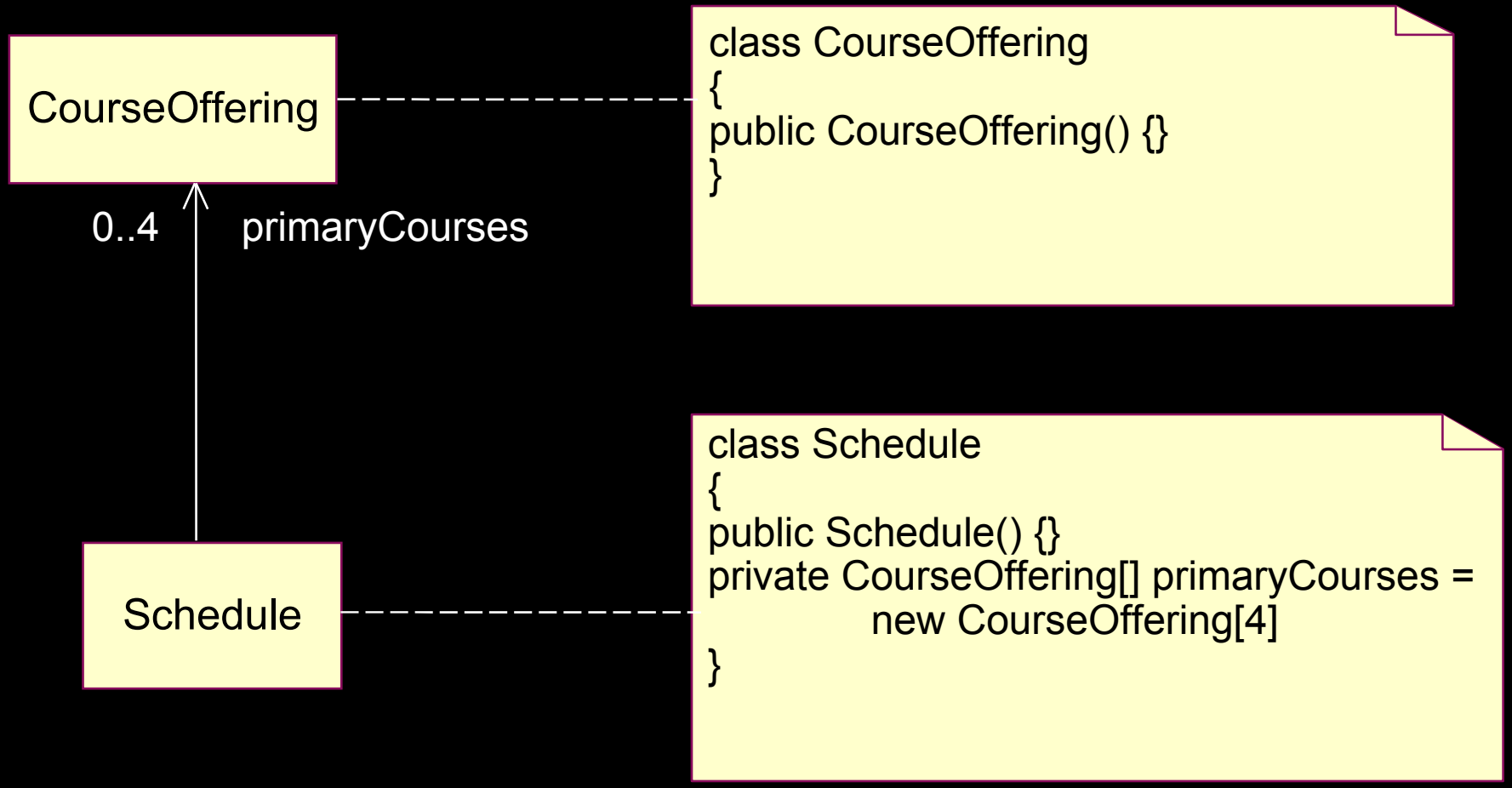
- Uni-directional associations



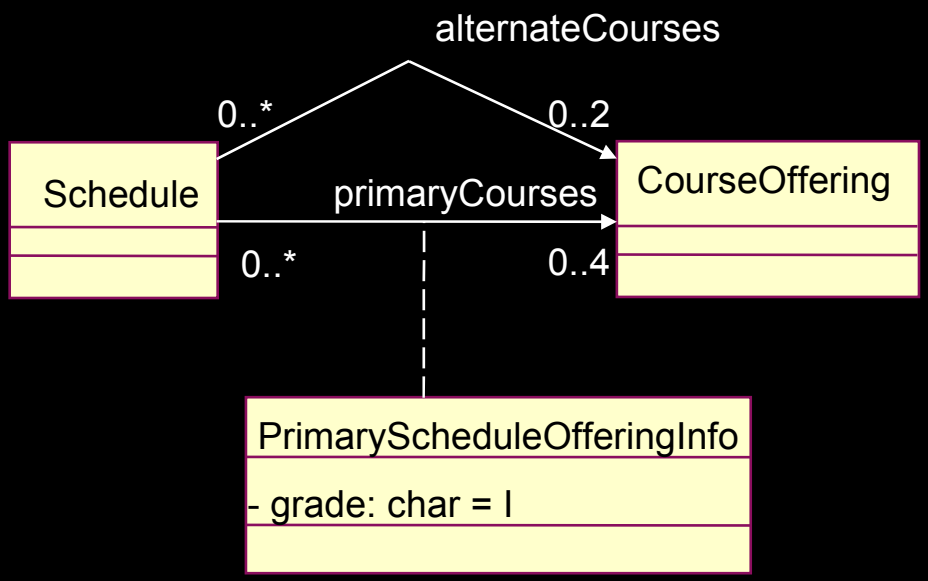
Association Roles



Association Multiplicity

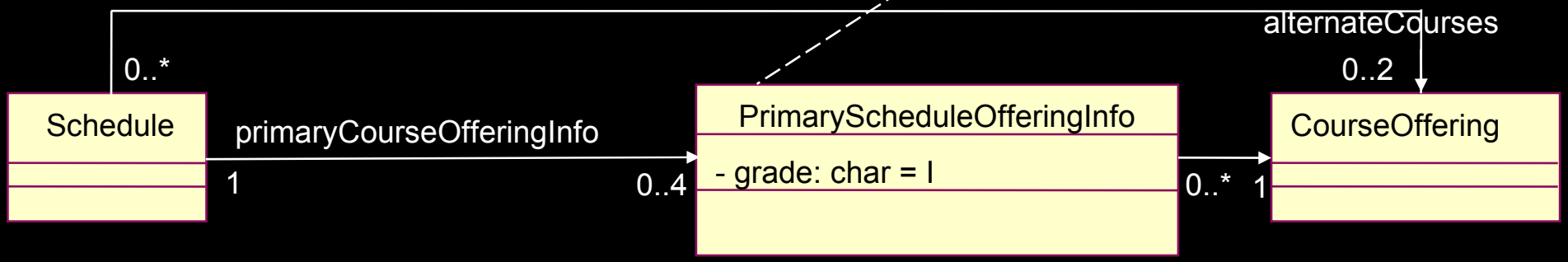


Association Class

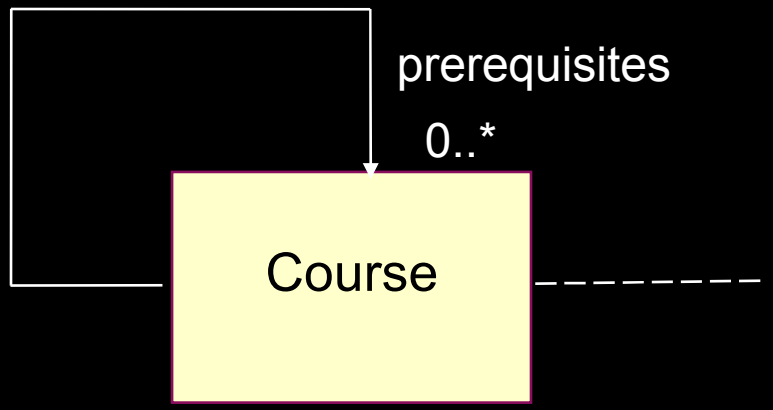



```
// No need to import if in the same package

class PrimaryScheduleOfferingInfo
{
  public PrimaryScheduleOfferingInfo() {}

  public CourseOffering get_theCourseOffering(){
     return theCourseOffering;
  }
  public void set_theCourseOffering(CourseOffering toValue){
     theCourseOffering = toValue;
  }
  private char get_Grade (){ return grade; }
  private void set_Grade(char toValue) { grade = toValue; }
  private char grade = 'I';
  private CourseOffering theCourseOffering;
}
```

Design Decisions

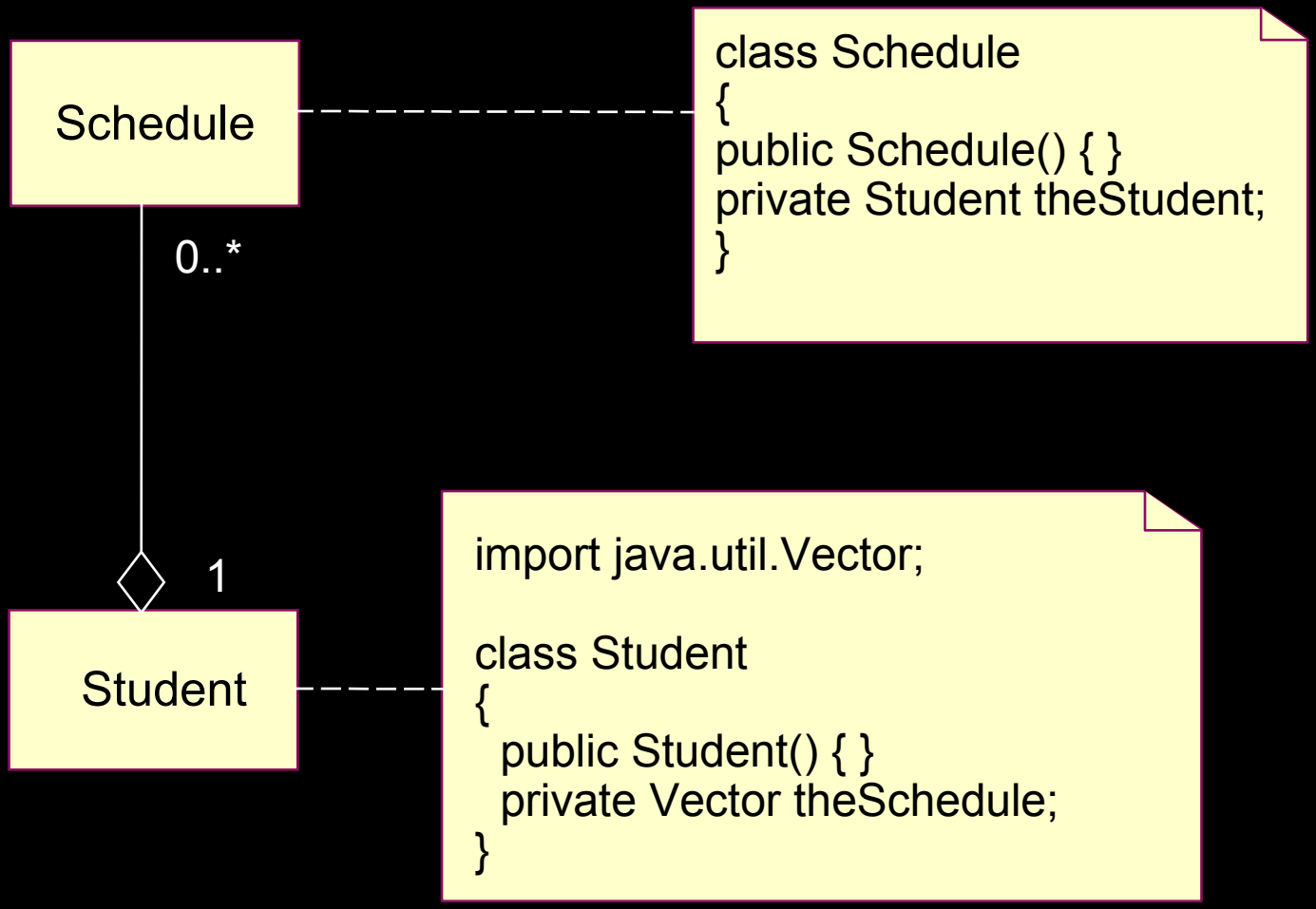


Reflexive Associations

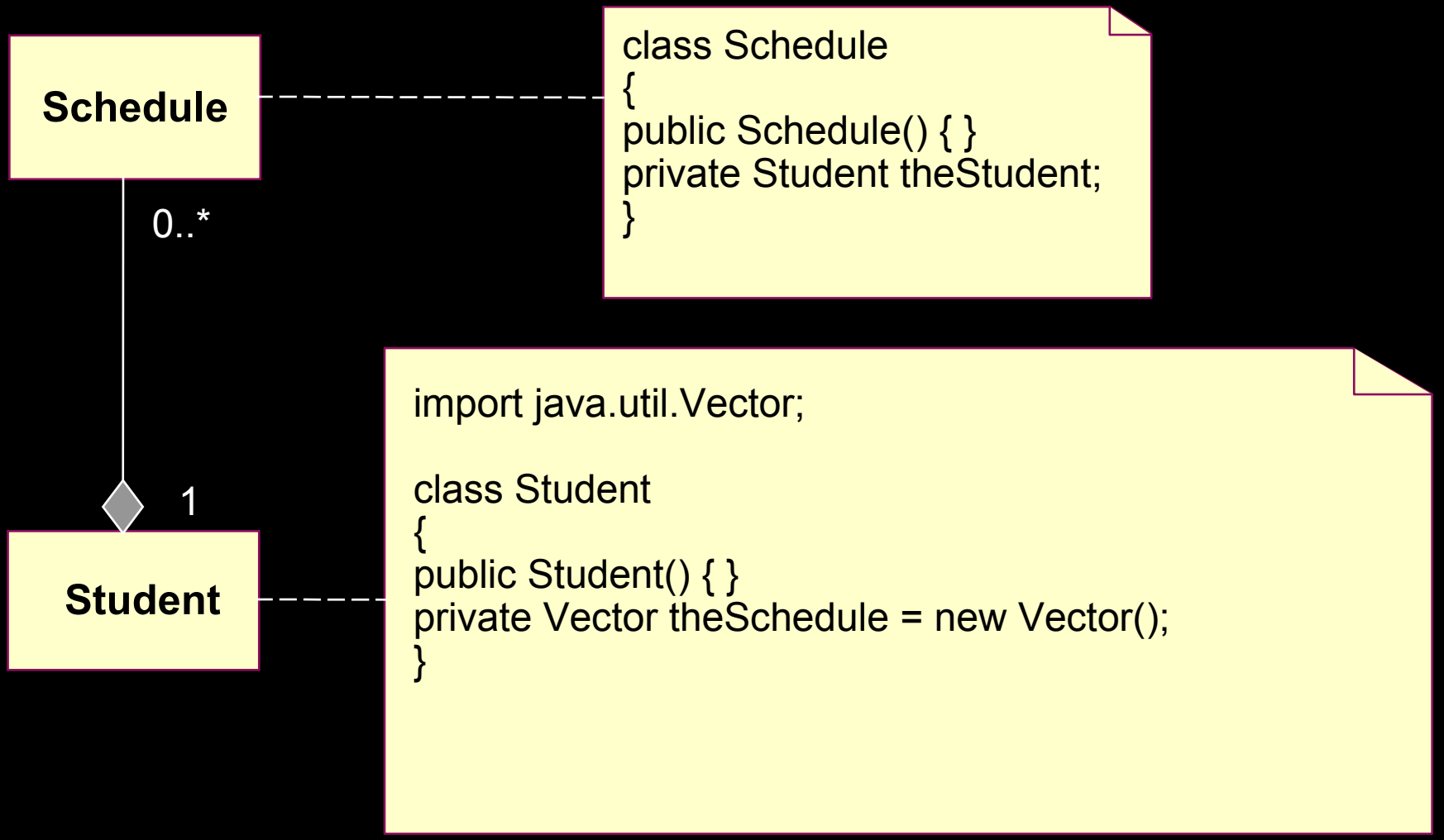



```
import java.util.Vector;

class Course
{
  public Course() {}
// The unbounded multiple association
//   is stored in a vector
  private Vector prerequisites;
}
```

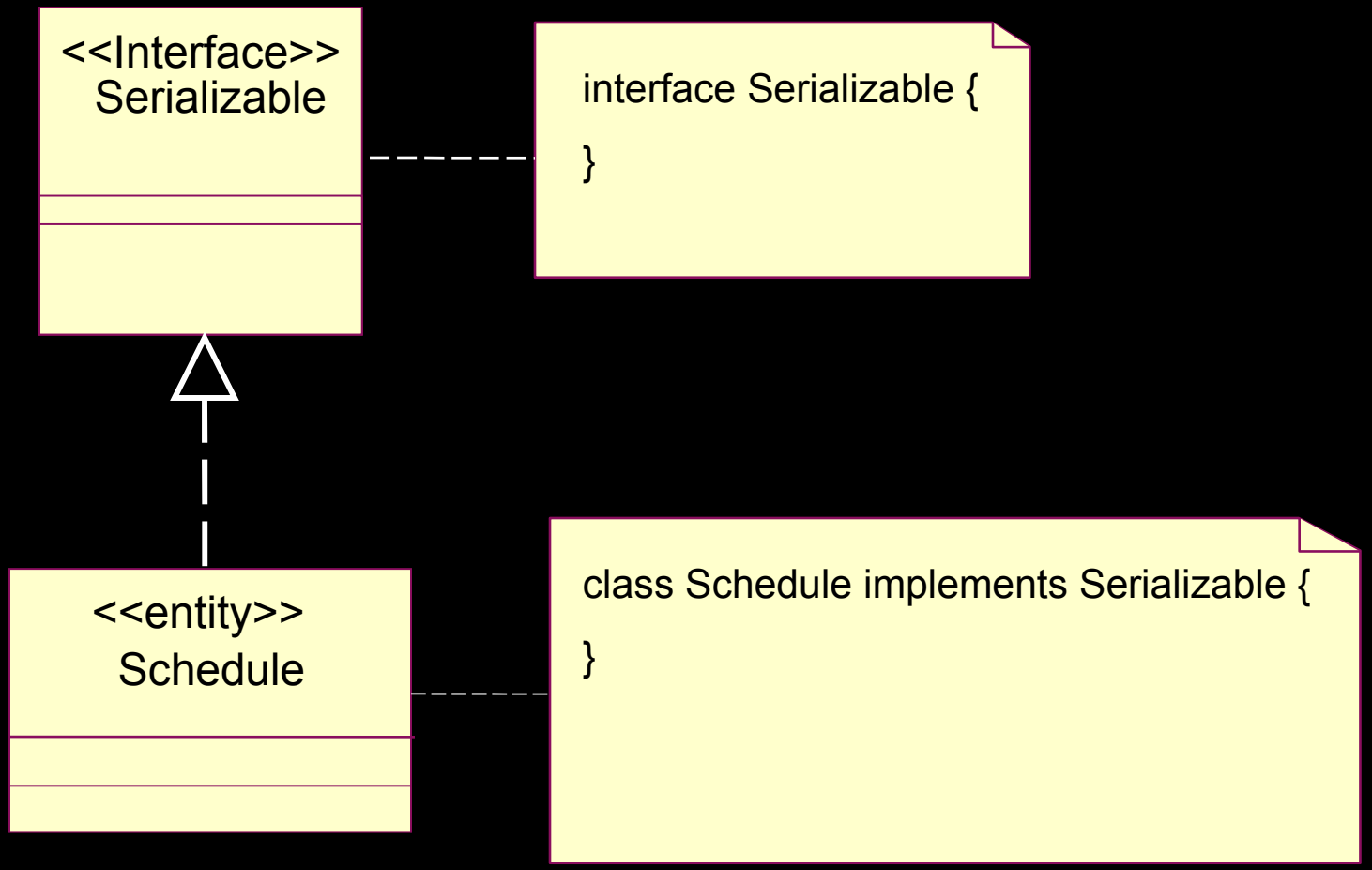
Aggregation



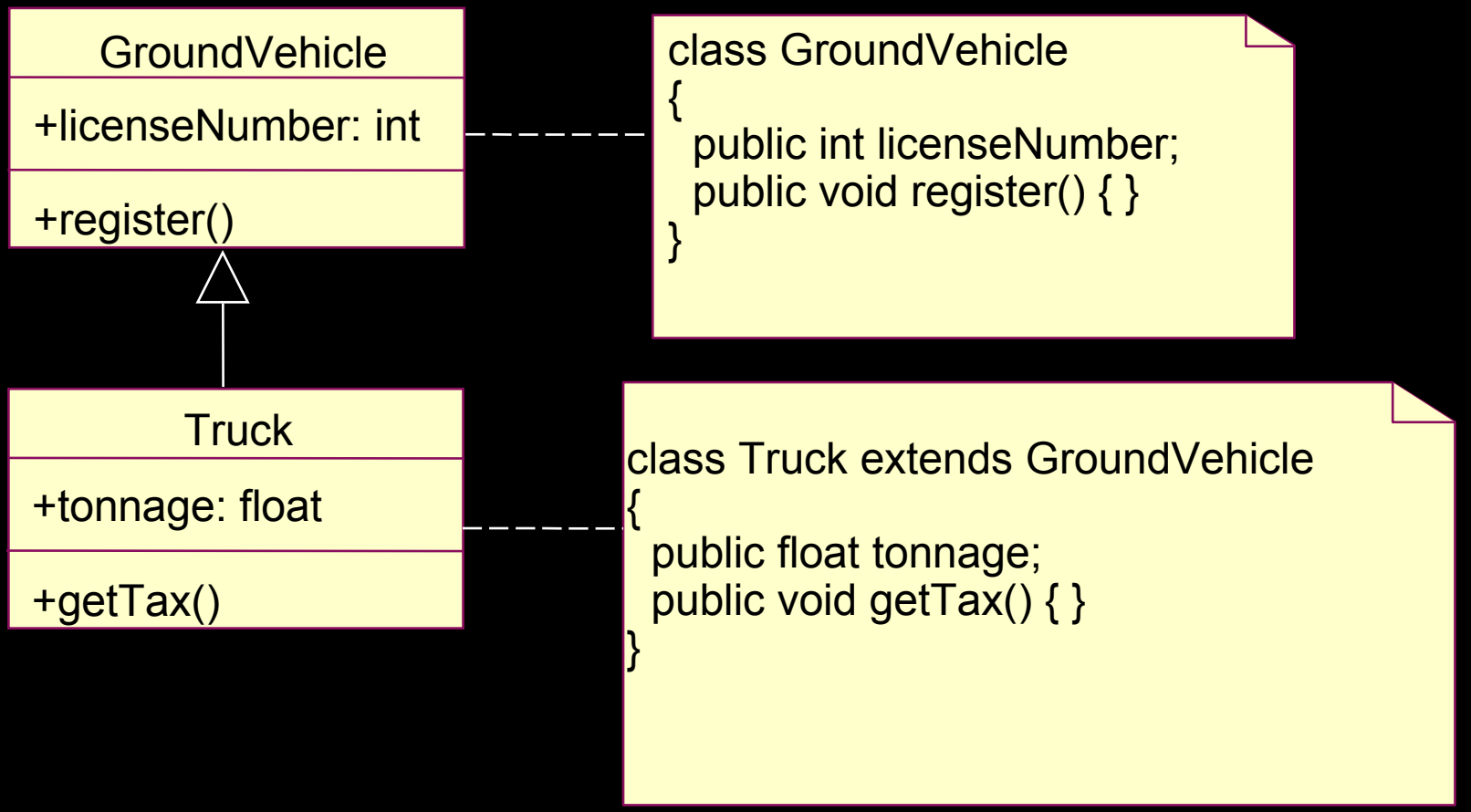
Composition



Interfaces and Realizes Relationships

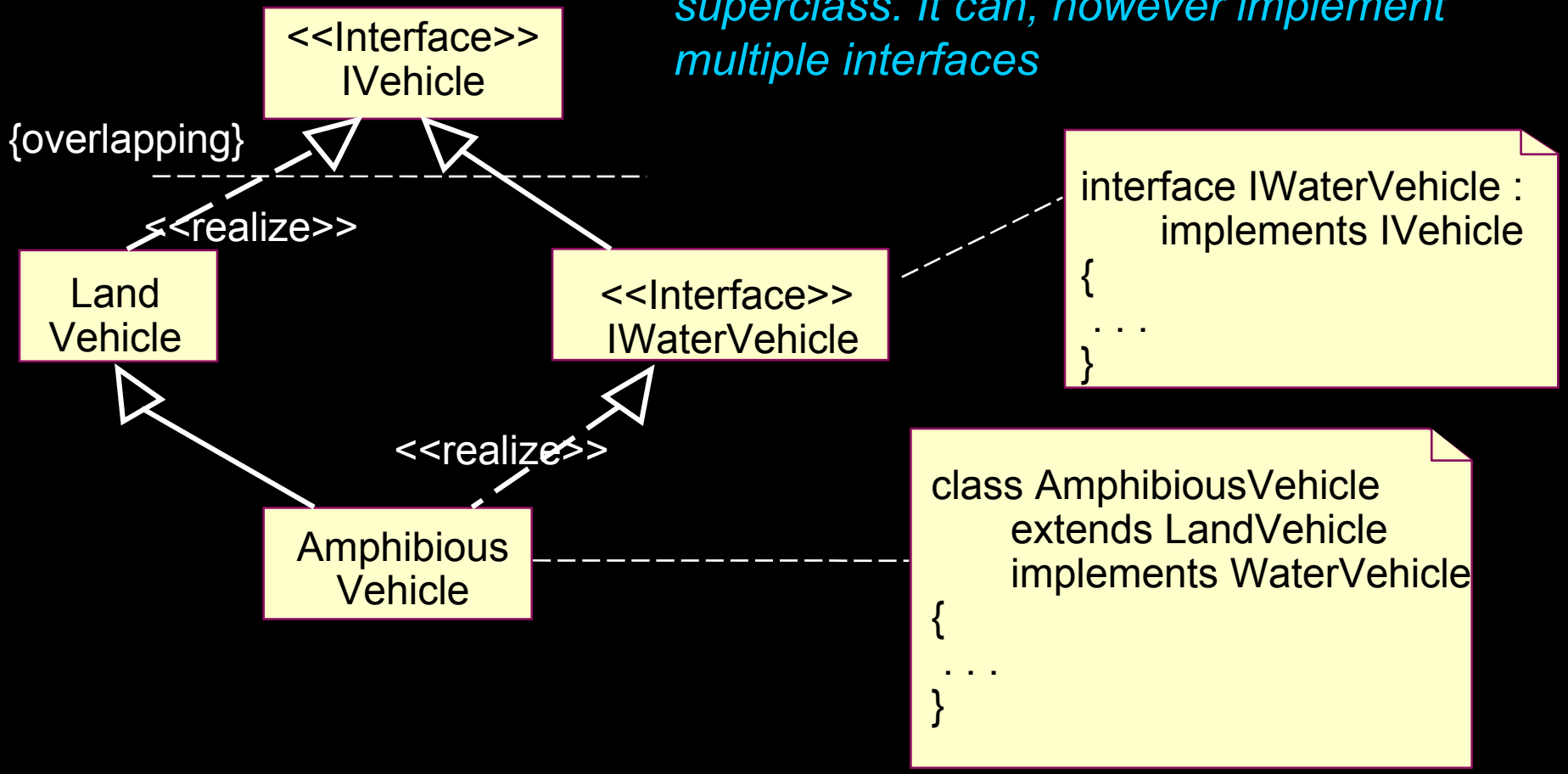


Generalization

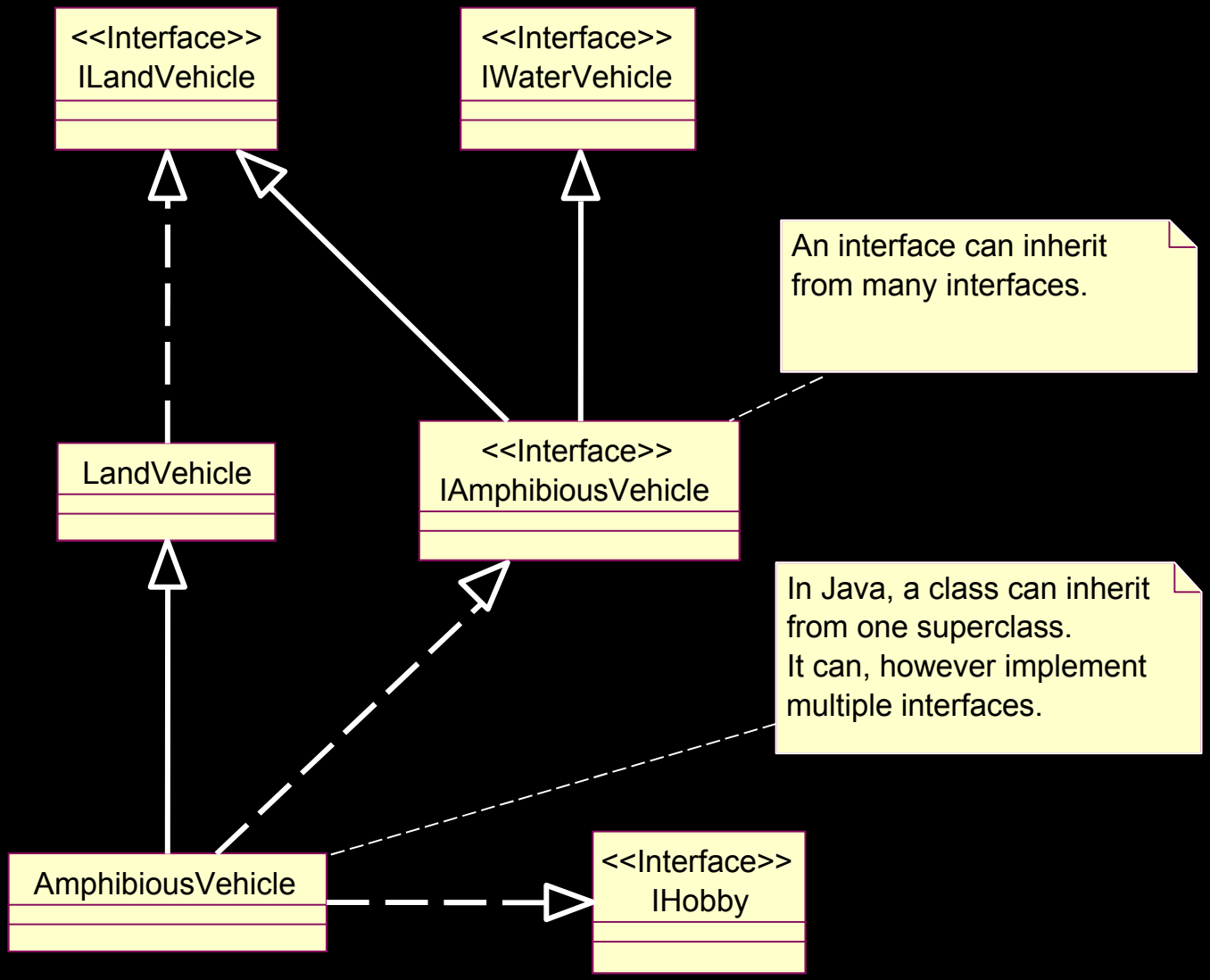


Multiple Inheritance

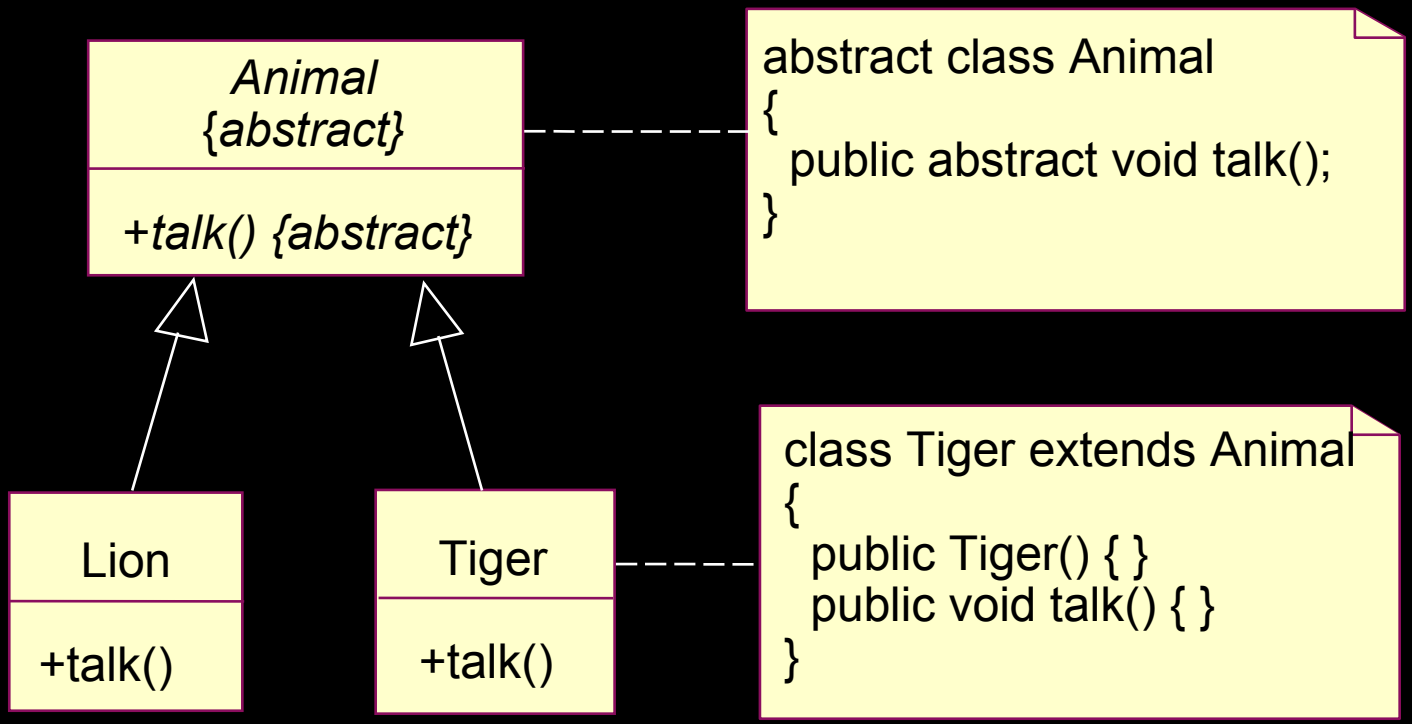
In Java, a class can only inherit from one superclass. It can, however implement multiple interfaces



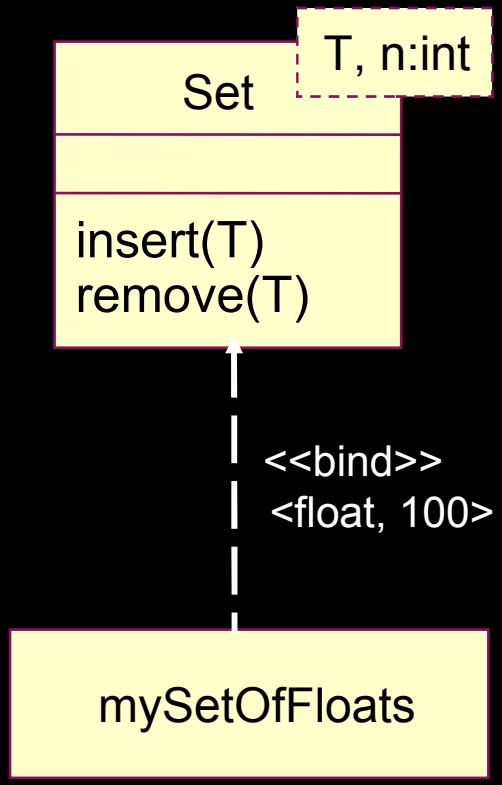
Multiple Inheritance (continued)



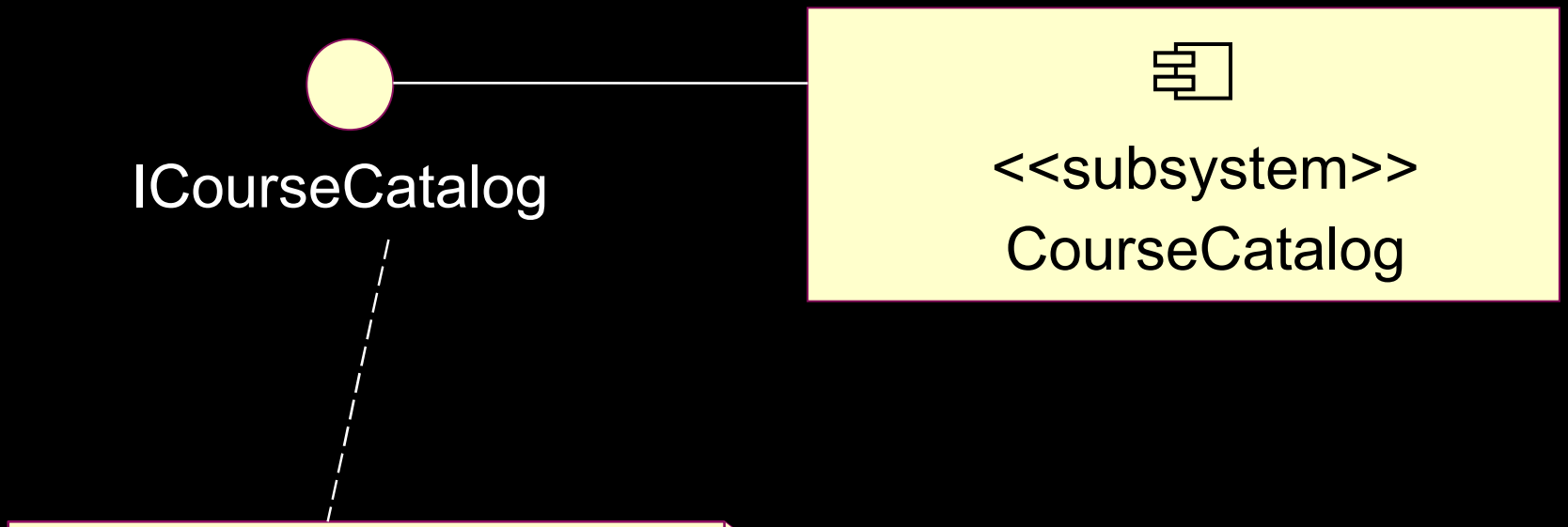
Abstract Class



Parameterized Class



Subsystems




```
package CourseCatalog;


public interface ICourseCatalog {

  public CourseOfferingList
getCourseOfferings();

}
```