CS 213 – Software Methodology
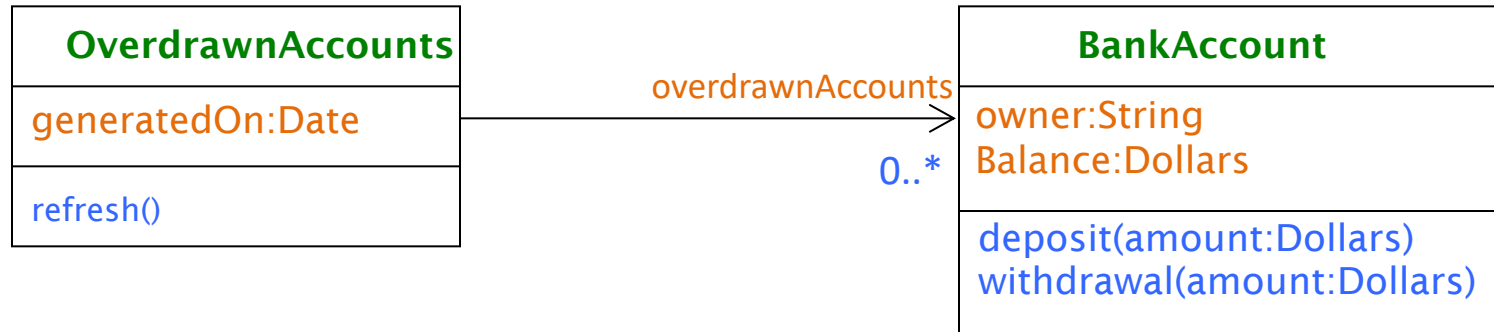# Spring 2023
Sesh Venugopal

Mar 8

UML Class Diagram - III

Unidirectional Association

(From From "UML basics: The class diagram" by Donald Bell)
http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

OverdrawnAccounts
generatedOn:Date
refresh()

overdrawnAccounts →

0..*

BankAccount
owner:String   Balance:Dollars
deposit(amount:Dollars)   withdrawal(amount:Dollars)

OverdrawnAccounts *knows about* BankAccount,
but BankAccount does not know about OverdrawnAccounts => in
the implementation OverdrawnAccounts would have a list of BankAccount
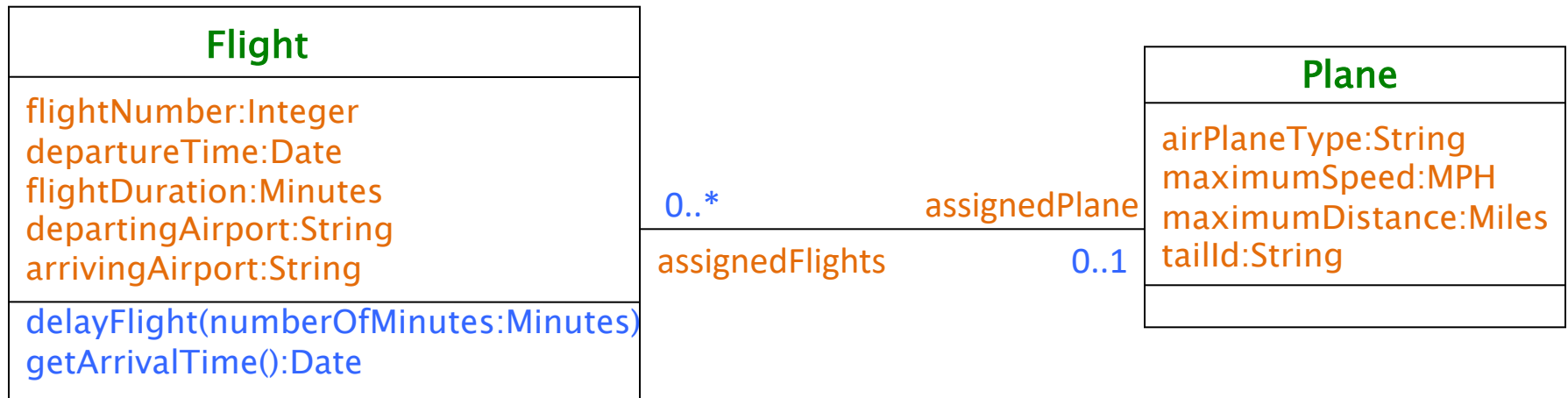instances, but BankAccount would not refer back to OverDrawnAccounts

Reflexive Association

Employee
firstName:String   lastName:String

1

manager

manages   0..*
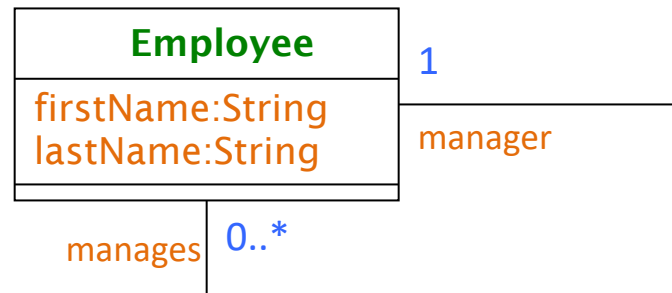
Bidirectional Association and Navigability

From "UML basics: The class diagram" by Donald Bell

http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/
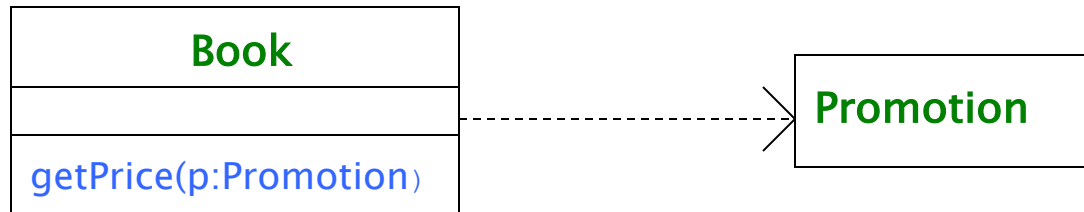

Flight *knows about* Plane, and Plane *knows about* Flight. When implemented, it will be easy to know which Plane is assigned to a Flight, and which Flights a Plane has been assigned to.

In other words, it is possible to **navigate** from Flight to Plane, and vice versa

Dependency

- Class A depends on class B if A <u>uses</u> B in such a way that a change in B will effect A

```
┌─────────────────────────────┐
│           Book              │
├─────────────────────────────┤
│                             │ ┄┄┄┄┄┄┄┄┄>  ┌──────────────┐
├─────────────────────────────┤             │  Promotion   │
│ getPrice(p:Promotion)       │             └──────────────┘
└─────────────────────────────┘
```
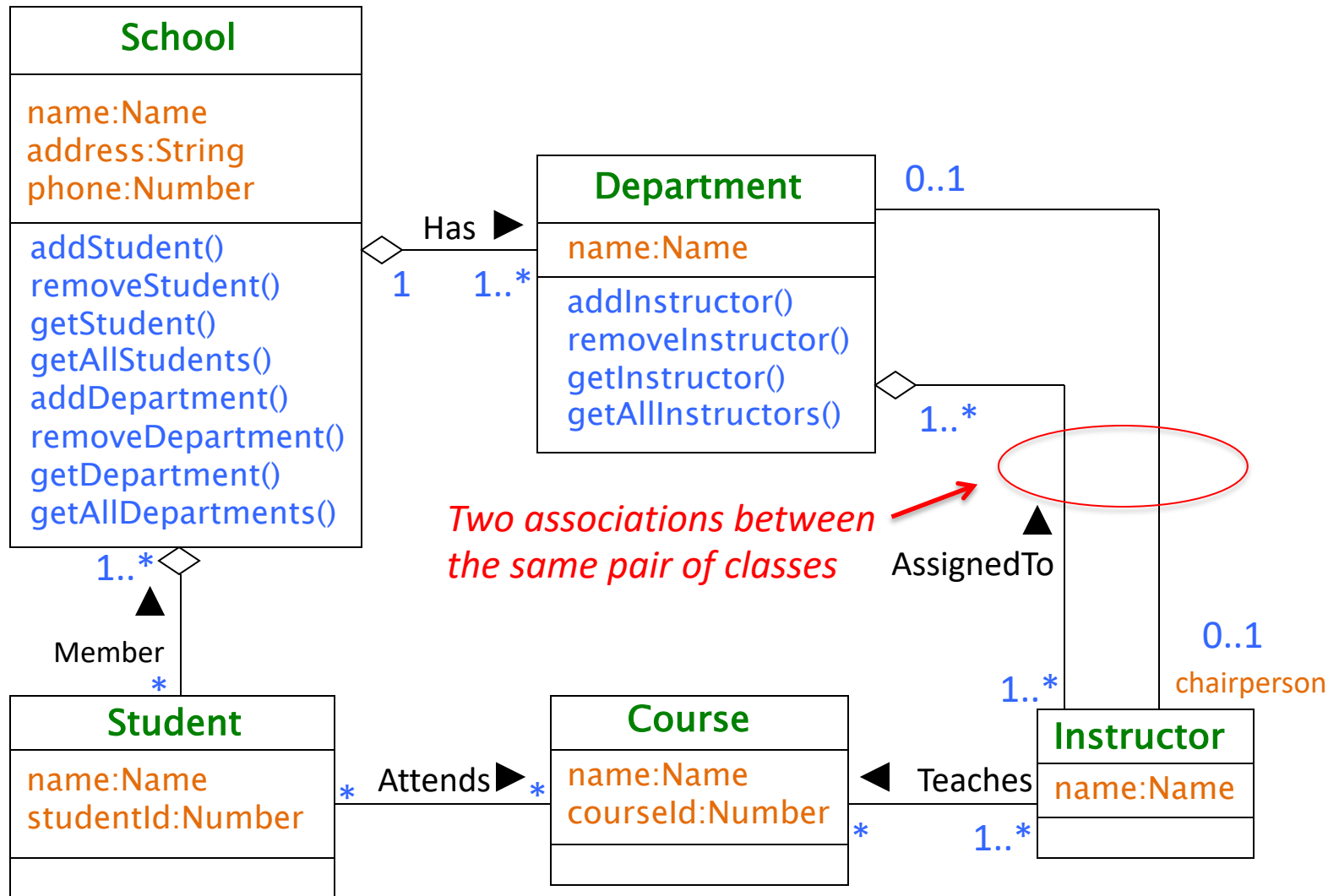
The Book class uses (depends on) on the Promotion class – changing Promotion would affect Book

- Say A depends on B. Typically then, B would appear as a parameter, return type, or local variable in a method of A

- Dependencies are the weakest kind of relationships, and the "dependee" class (e.g. Promotion) is subordinate to the "depender/dependent" class (e.g. Book) In other words, the Book class can be defined meaningfully even without the Promotion class

Modeling a University

From "The Unified Modeling Language User Guide" by Booch, Rumbaugh, and Jacobson

School

name:Name
address:String
phone:Number

addStudent()
removeStudent()
getStudent()
getAllStudents()
addDepartment()
removeDepartment()
getDepartment()
getAllDepartments()

Department

name:Name

addInstructor()
removeInstructor()
getInstructor()
getAllInstructors()

Has ▶   1   1..*   0..1

1..*

*Two associations between
the same pair of classes*

AssignedTo

Member
*

1..*   0..1
chairperson

Student

name:Name
studentId:Number

* Attends ▶ *

Course

name:Name
courseId:Number

◀ Teaches

*   1..*

Instructor

name:Name

Modeling a University
Relationship data

Write the Student and Course classes with just fields (no methods)
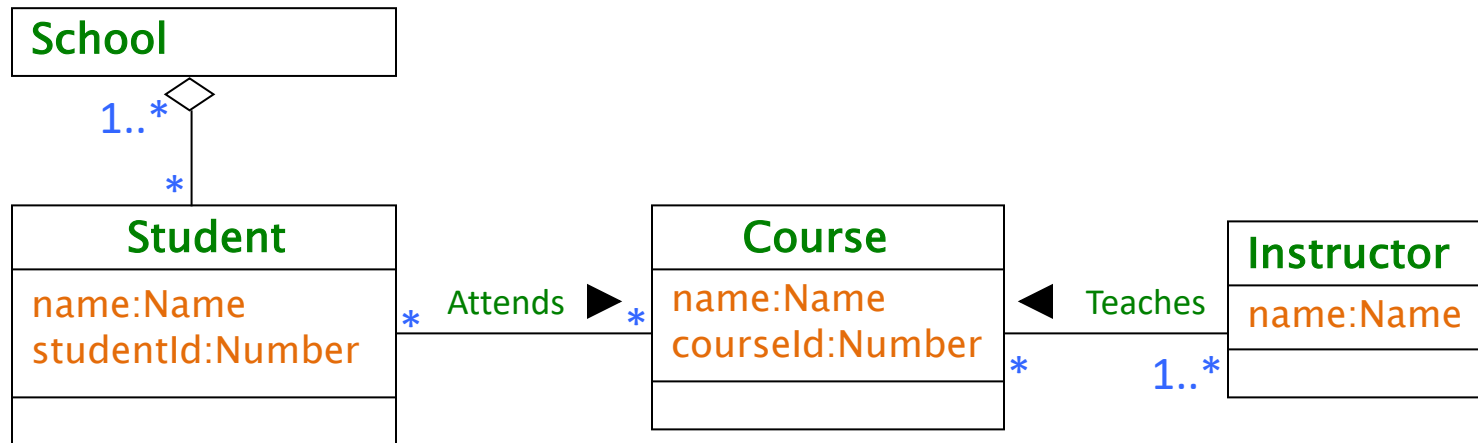
Modeling a University
Relationship data

```
School
     ◇
1..*
     *
```

```
Student
─────────────
name:Name
studentId:Number
─────────────
```

Attends ▶

```
Course
─────────────
name:Name
courseId:Number
─────────────
```

◀ Teaches

```
Instructor
─────────────
name:Name
─────────────
```

*   *   *   1..*

```java
public class Student
{
    Name name;
    Number studentId;

    Course[] courses;
    School[] schools;
    ...
}
```

```java
public class Course {
    Name name;
    Number courseId;

    ArrayList<Student> students;
    Instructor[] instructors;
    ...
}
```
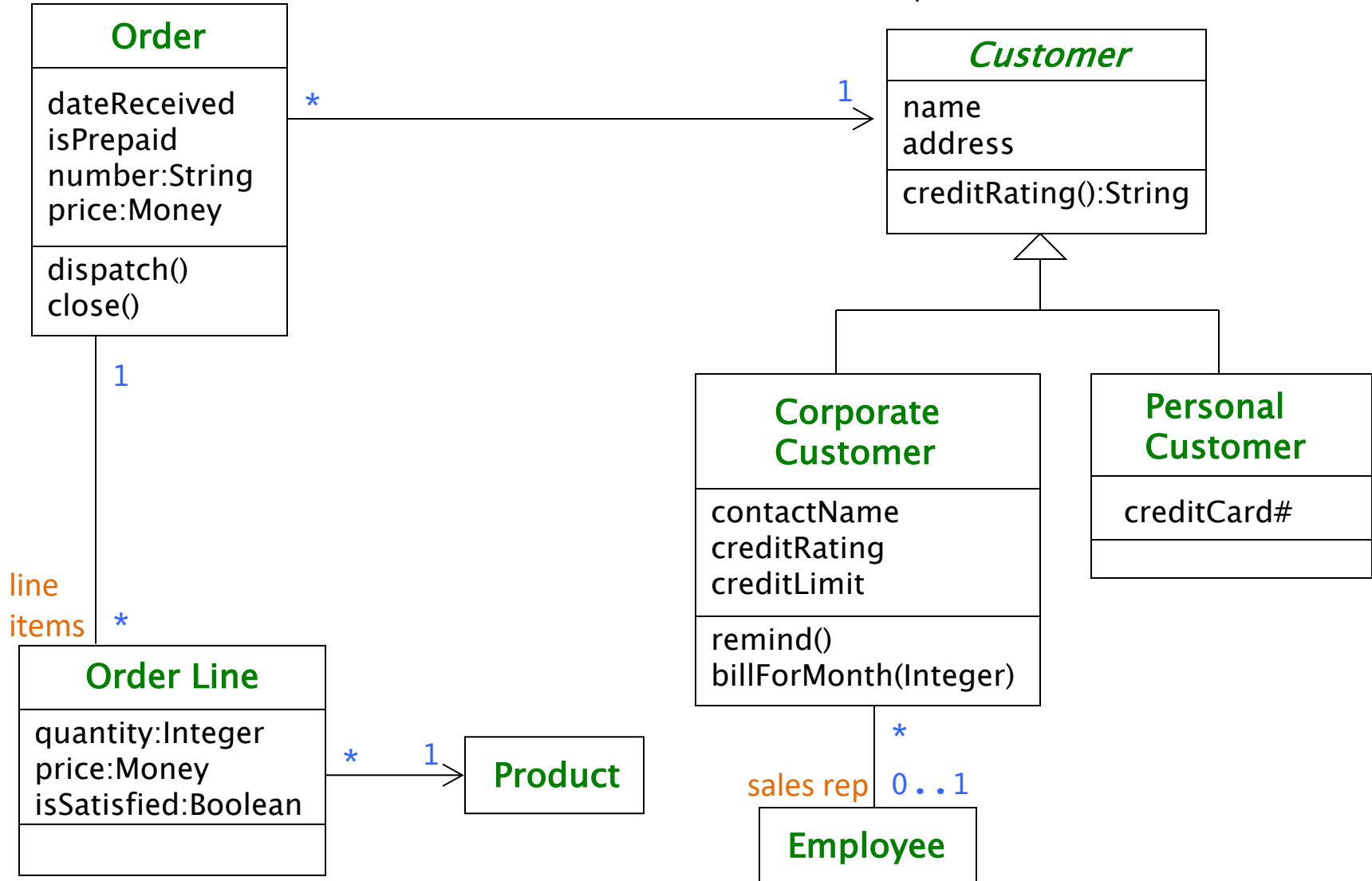
**Data needed to store a relationship between two classes
DO NOT show up as UML attributes in either class, because
they are NOT inherent properties of either.
Instead, they are properties of the relationship between the
classes.**

Modeling Orders for Products

Draw a UML Class Diagram to model orders for products. Customers who place orders can be personal customers or corporate customers. An order could have multiple products (items) in it, and at any point in time the company needs to know which of these items have been shipped.

Modeling Orders for Products

From "UML Distilled" by Martin Fowler with Kendall Scott

Order
- dateReceived
- isPrepaid
- number:String
- price:Money

- dispatch()
- close()

Customer
- name
- address

- creditRating():String

Order Line
- quantity:Integer
- price:Money
- isSatisfied:Boolean

Product

Corporate Customer
- contactName
- creditRating
- creditLimit

- remind()
- billForMonth(Integer)

Personal Customer
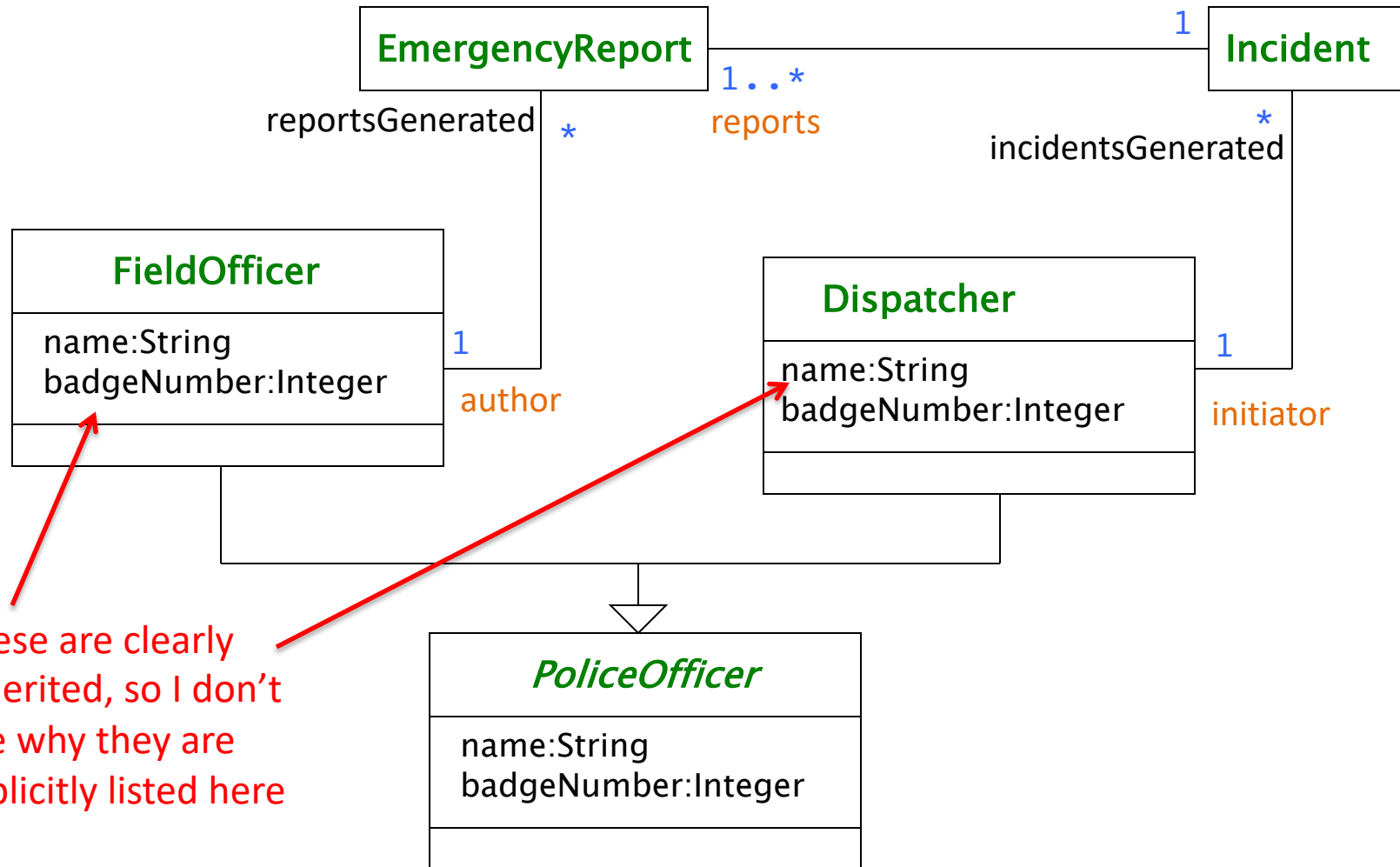- creditCard#

Employee

line items

sales rep

Modeling Police Incidents

Draw a UML Class Diagram to model the handling of incidents by the dispatchers and field officers of a police department. Dispatchers log incidents, and field officers write emergency reports, as many as each incident requires.

Abstract Class/Inheritance/Bidirectional Association

From "Object-Oriented Software Engineering" 2nd ed. by Bruegge and Dutoit

EmergencyReport ——————————— 1 **Incident**

reportsGenerated   1..*   reports   incidentsGenerated   *

FieldOfficer
name:String
badgeNumber:Integer

1   author

Dispatcher
name:String
badgeNumber:Integer

1   initiator

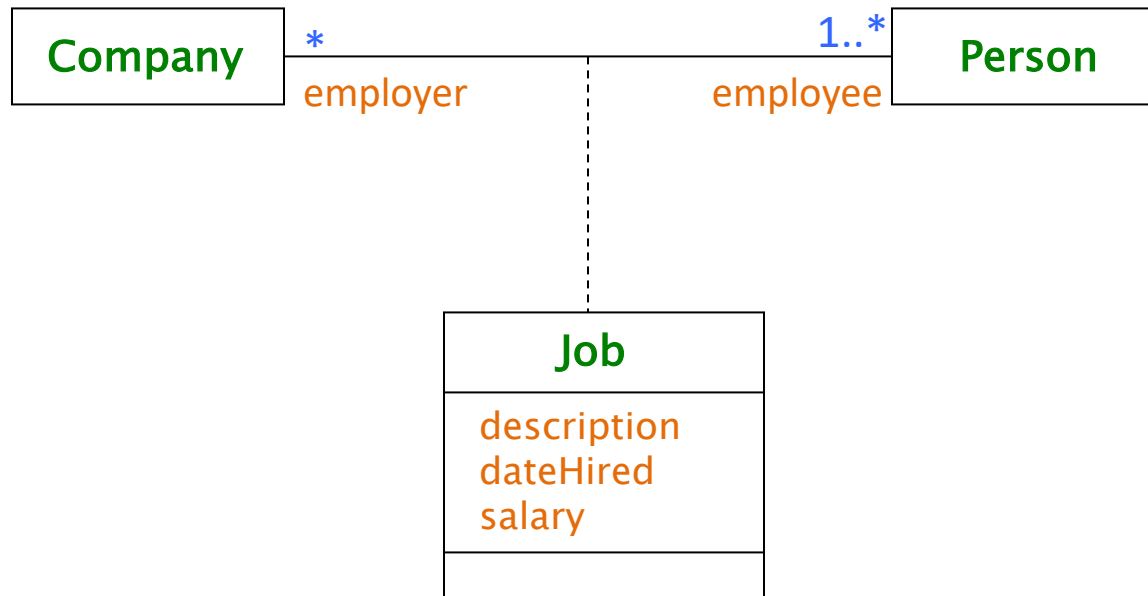*PoliceOfficer*
name:String
badgeNumber:Integer

These are clearly inherited, so I don't see why they are explicitly listed here

Association Class

- At times, an association between two classes itself has properties - an **association class** is used to model these properties.

From "The Unified Modeling Language User Guide" by Booch, Rumbaugh, and Jacobson

```
┌──────────────┐  *                    1..* ┌──────────────┐
│   Company    │──── employer    employee ──│   Person     │
└──────────────┘            │               └──────────────┘
                            ┊
                    ┌───────────────┐
                    │      Job      │
                    ├───────────────┤
                    │ description   │
                    │ dateHired     │
                    │ salary        │
                    ├───────────────┤
                    │               │
                    └───────────────┘
```

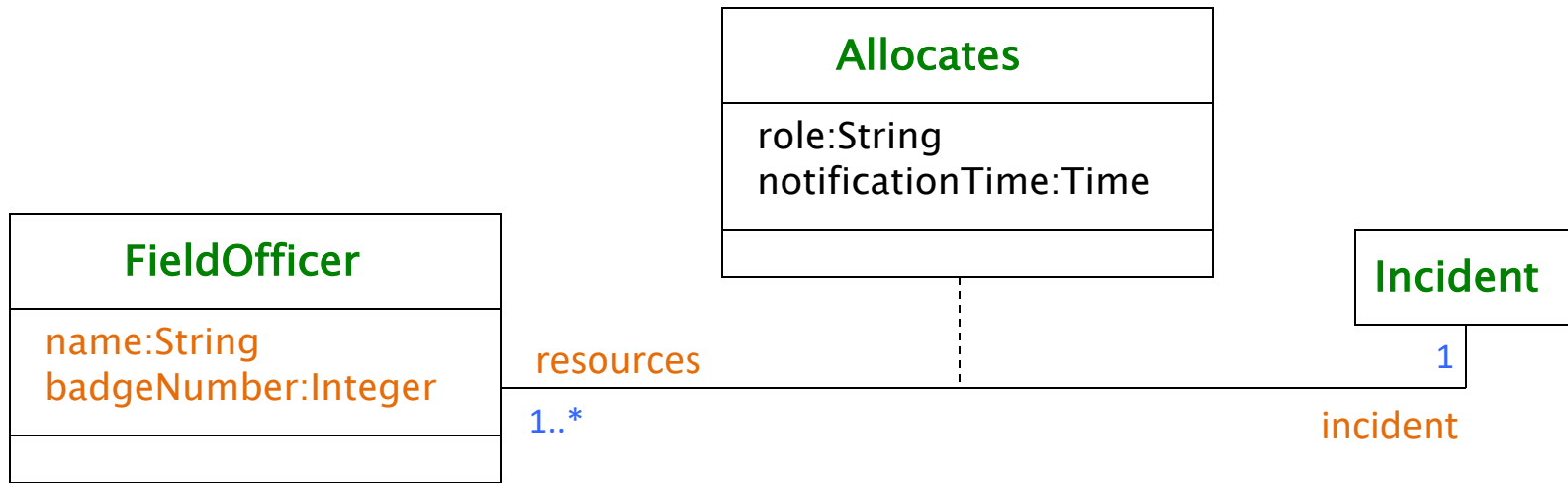# Association Class: Example

From "UML basics: The class diagram" by Donald Bell

http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

Flight

flightNumber:Integer
departureTime:Date
flightDuration:Minutes=60

delayFlight(numberOfMinutes:Minutes)
getArrivalTime():Date

0..*
flights

passengers
0..*

FrequentFlyer

firstName:String
lastName:String
frequentFlyerNumber:String

MileageCredit

baseMiles:Integer
bonusMiles:Integer
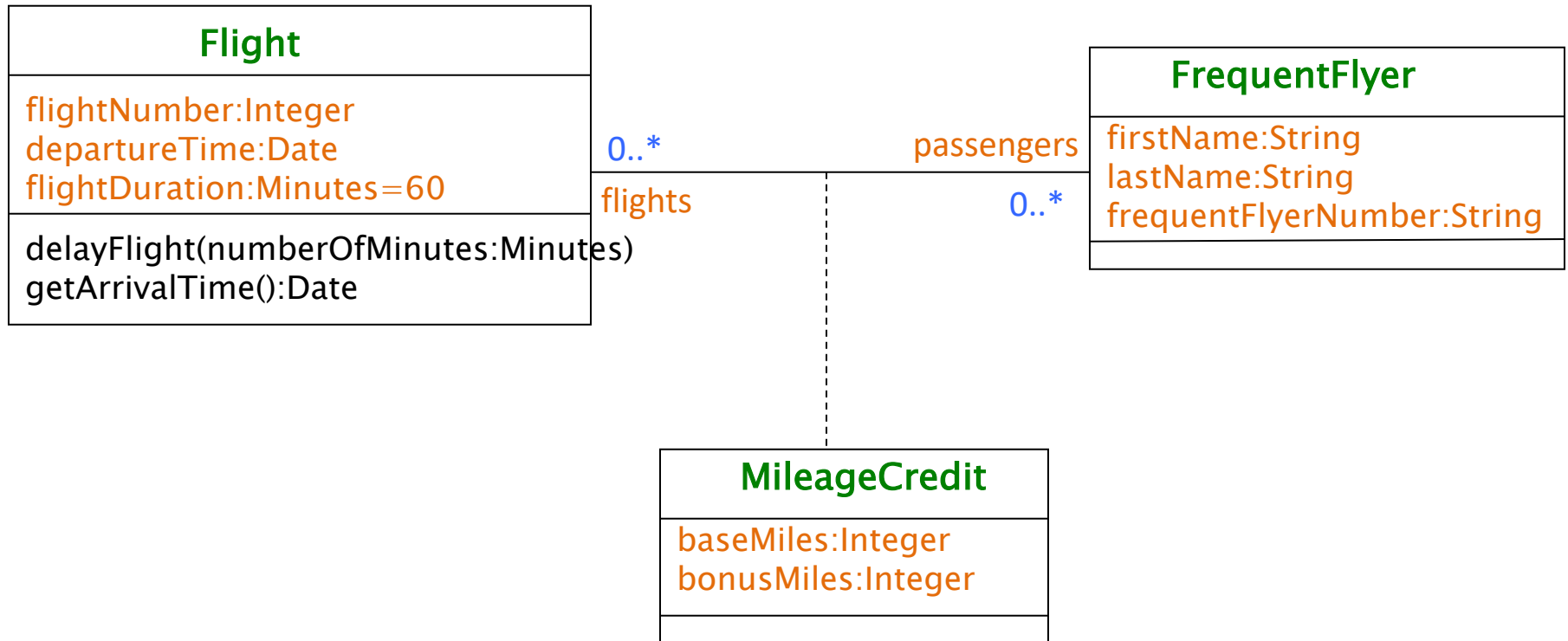
Association Class: Example

From "Object-Oriented Software Engineering" 2nd ed. by Bruegge and Dutoit

Allocates

role:String
notificationTime:Time

FieldOfficer

name:String
badgeNumber:Integer

resources

1..*

Incident

1

incident

Is this equivalent to the above?

Allocates

role:String
notificationTime:Time

1

1

FieldOfficer

name:String
badgeNumber:Integer

incident

1

Incident

1..*

resources