MACIASZEK, L.A. (2007):

***Requirements Analysis and System Design*, 3rd ed.**

Addison Wesley, Harlow England

ISBN 978-0-321-44036-5

Chapter 4

Moving from Analysis to Design

Topics

- Advanced class modeling
- Advanced generalization and inheritance modeling
- Advanced aggregation and delegation modeling
- Advanced interaction modeling

1. Advanced class modeling

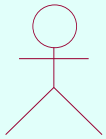
- stereotypes, constraints, derived information, visibility, qualified associations, association class, parameterized class, etc

Extension mechanisms

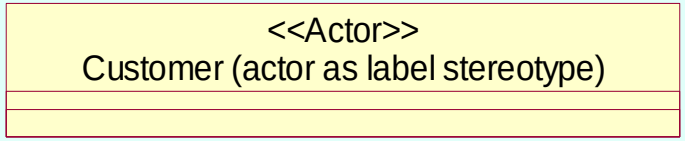
- specify “how specific UML model elements are customized and extended with new semantics by using
  - stereotypes,
  - constraints,
  - tag definitions, and
  - tagged values”
- UML profile
  - extends a reference <u>metamodel</u> (i.e. UML itself)
  - coherent set of extensions, defined for specific purposes

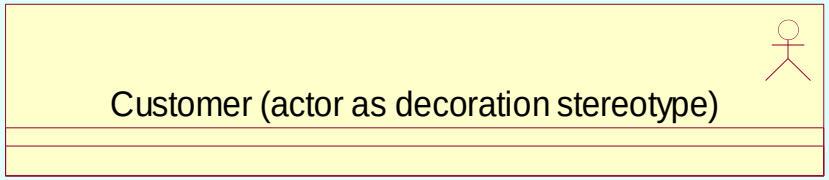
Stereotypes

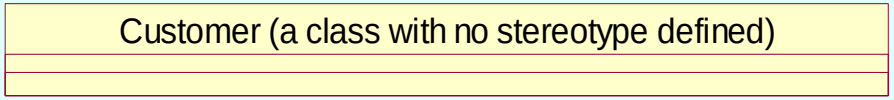
- extends an existing UML modeling element
  - varies the semantics of an existing element (it is not a new model element per se)



Customer (actor as icon stereotype)

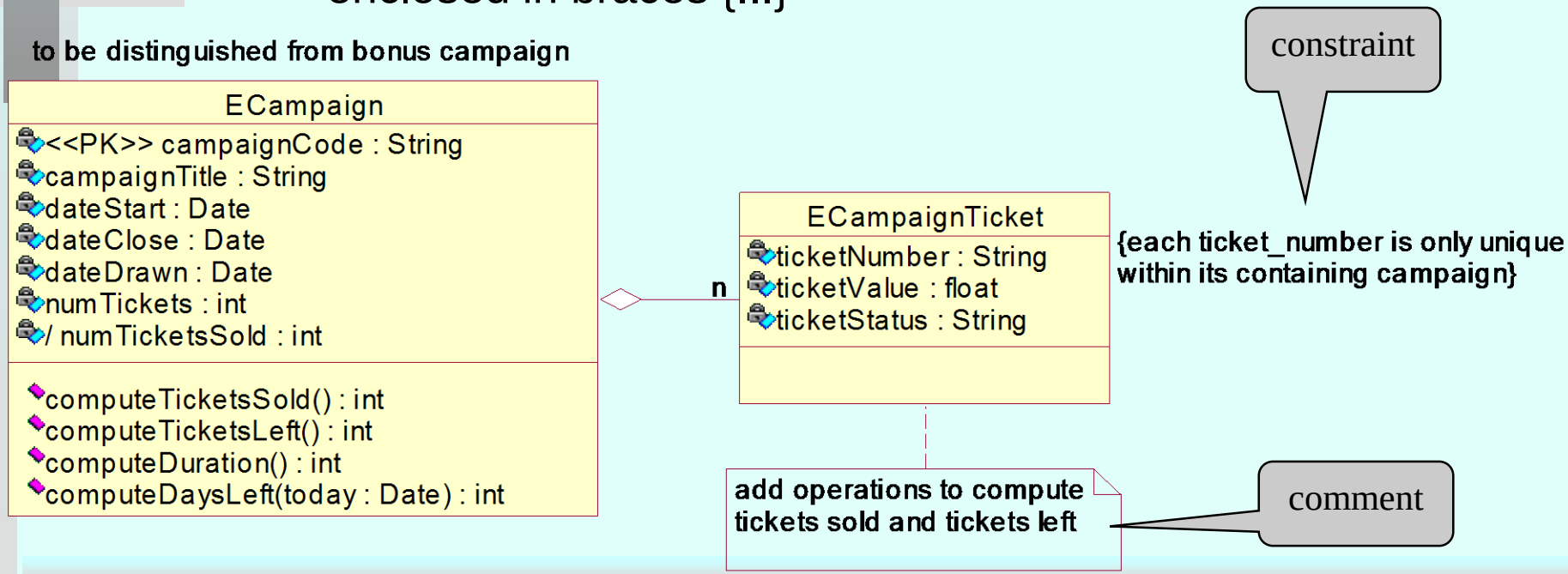




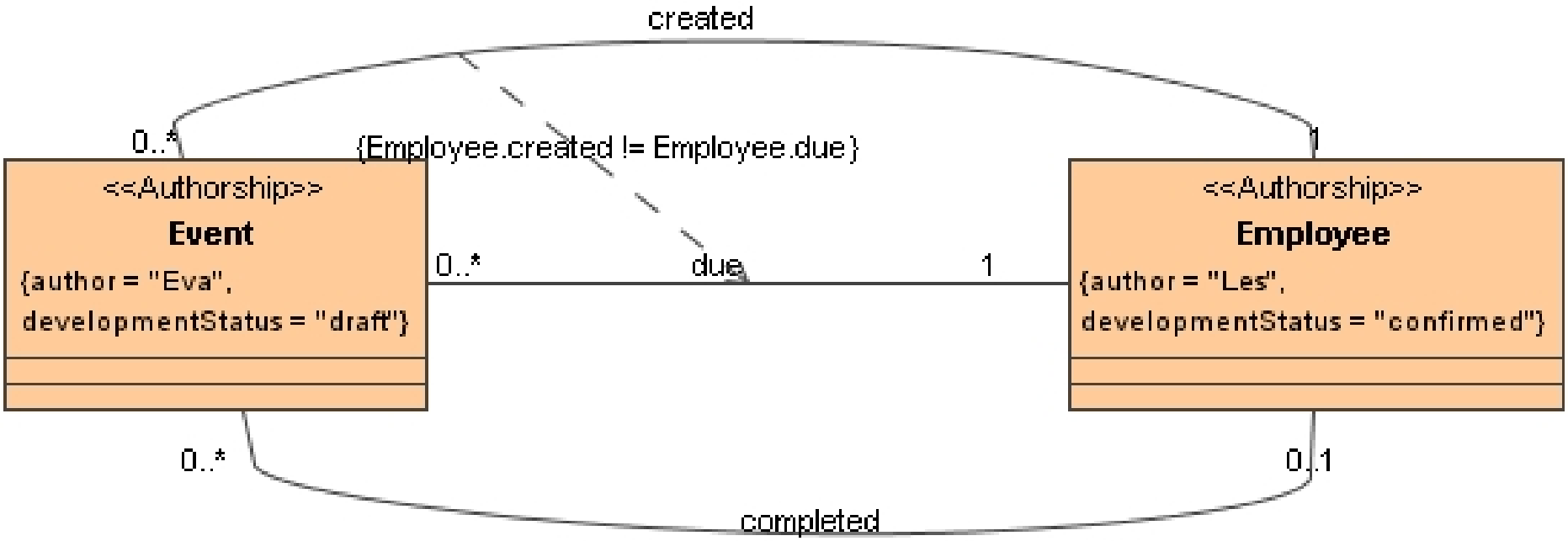


Comments and constraints

- **Comment** - text string attached to a model element
  - can be presented as a UML **note**
- ***Constraint*** - <u>semantic</u> relationship among model elements that specifies conditions and propositions that must be maintained as true
  - enclosed in braces {...}



Constraint on association



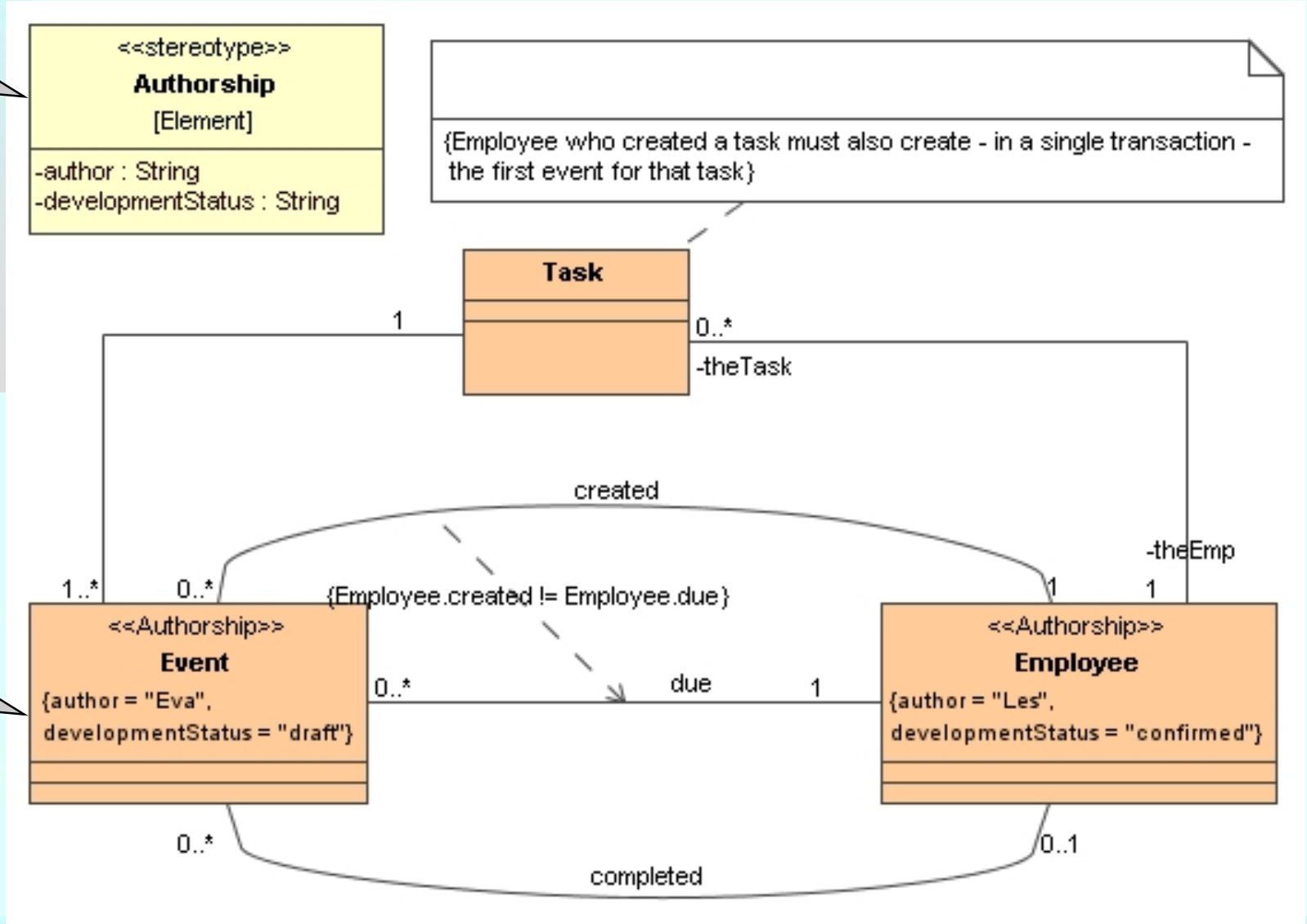
Tags

- **Tag definition** – a property of a stereotype and is shown as an attribute of a class rectangle that contains the stereotype declaration.
- **Tag value** – “a name–value pair that may be attached to a model element that uses a stereotype containing a tag definition.”
  - the keyword is called a **tag**
  - like **constraints**, tag values represent arbitrary textual information and are written inside curly brackets
  - like stereotypes and constraints, a few tags are **predefined** in UML
  - typical use of tags is in providing **project management** information

Tags - example

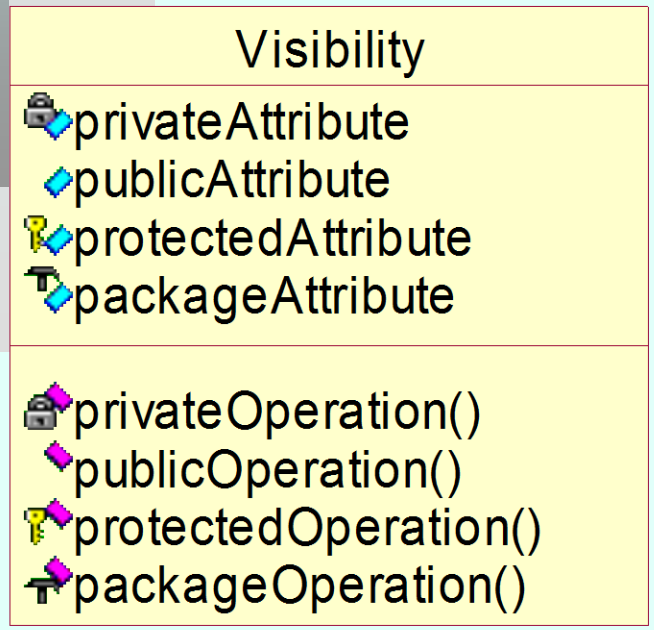
tag definition

tag values



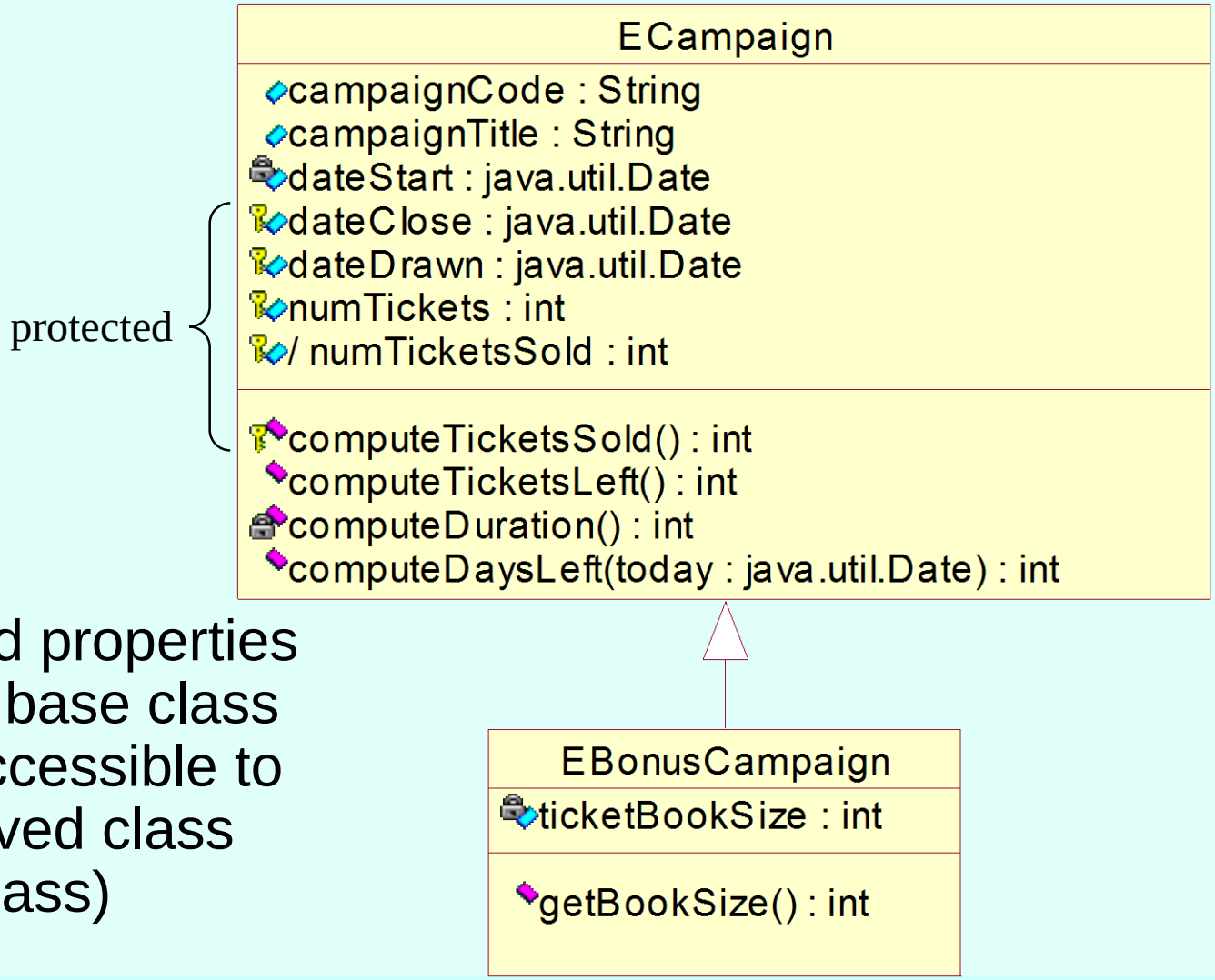
Visibility and encapsulation

+ for public visibility
– for private visibility
for protected visibility
~ for package visibility



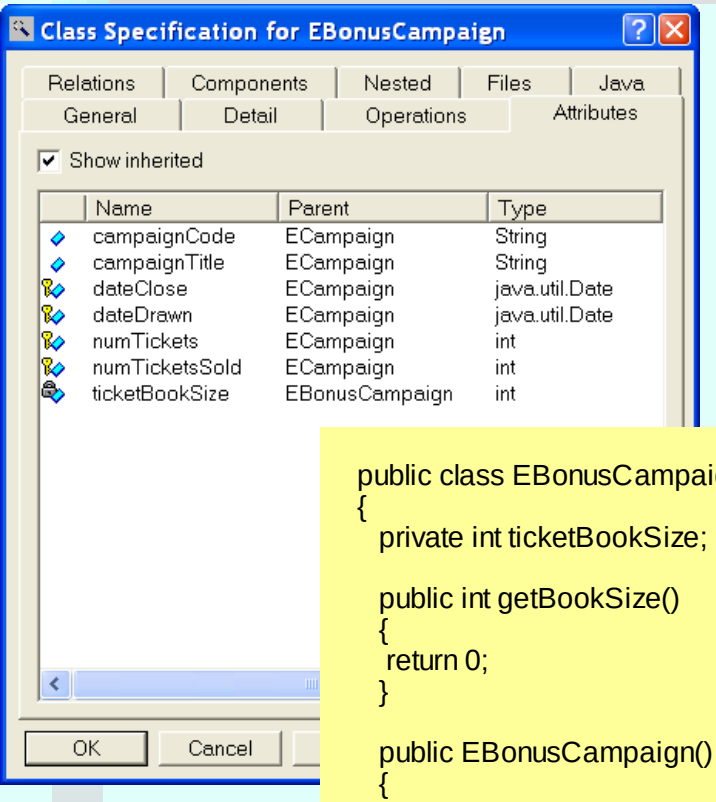

```
public class Visibility
{
   private int privateAttribute;
   public int publicAttribute;
   protected int protectedAttribute;
   int packageAttribute;

   private void privateOperation()
   public void publicOperation()
   protected void protectedOperation()
    void packageOperation()
}
```

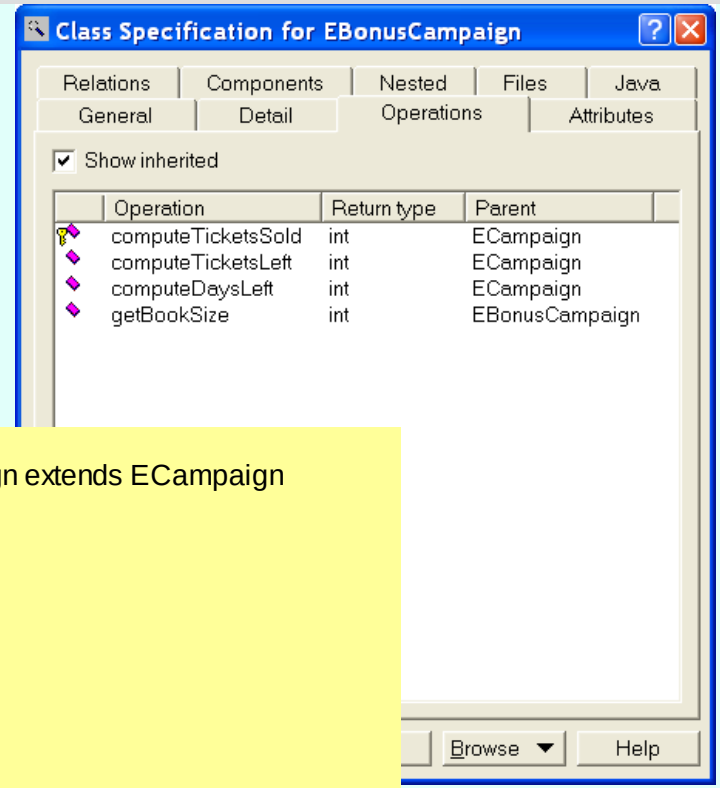
Protected visibility



protected properties in the base class are accessible to a derived class (subclass)

Accessibility of inherited class properties



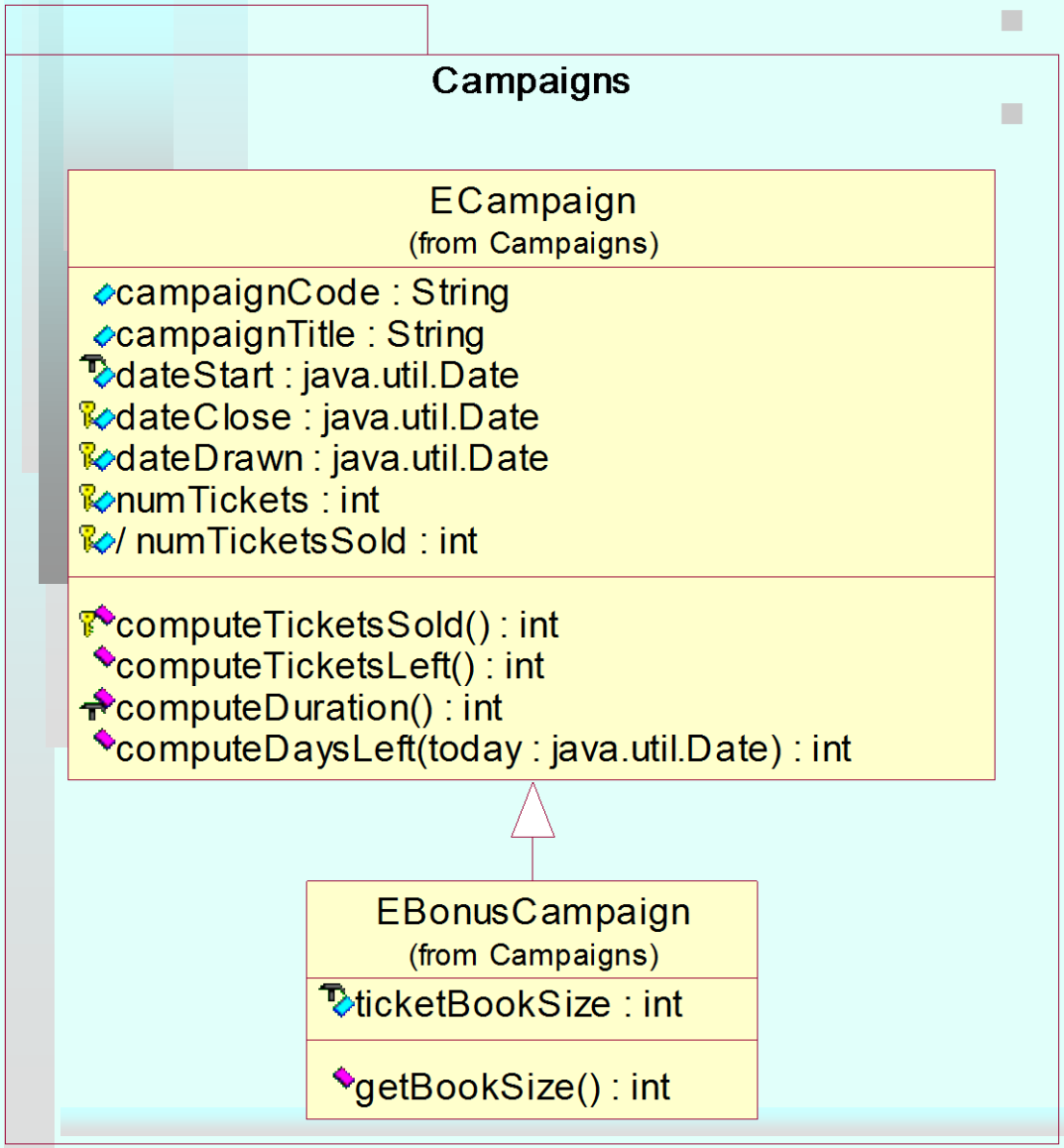



```
public class EBonusCampaign extends ECampaign
{
  private int ticketBookSize;

  public int getBookSize()
  {
   return 0;
  }

  public EBonusCampaign()
  {
   System.out.println("EBonusCampaign constructor);
  }

  public static void main(String[] args)
  {
   EBonusCampaign x = new EBonusCampaign();
   x.computeDuration();
   //The above will fail (x cannot access computeDuration)
  }
}
```

dateStart and computeDuration() of ECampaign not accessible in EBonusCampaign

- inheriting a property does not necessarily mean that the property is accessible with the objects of a derived class
- private properties of the base class remain private to the base and are inaccessible to objects of the derived class

Package visibility



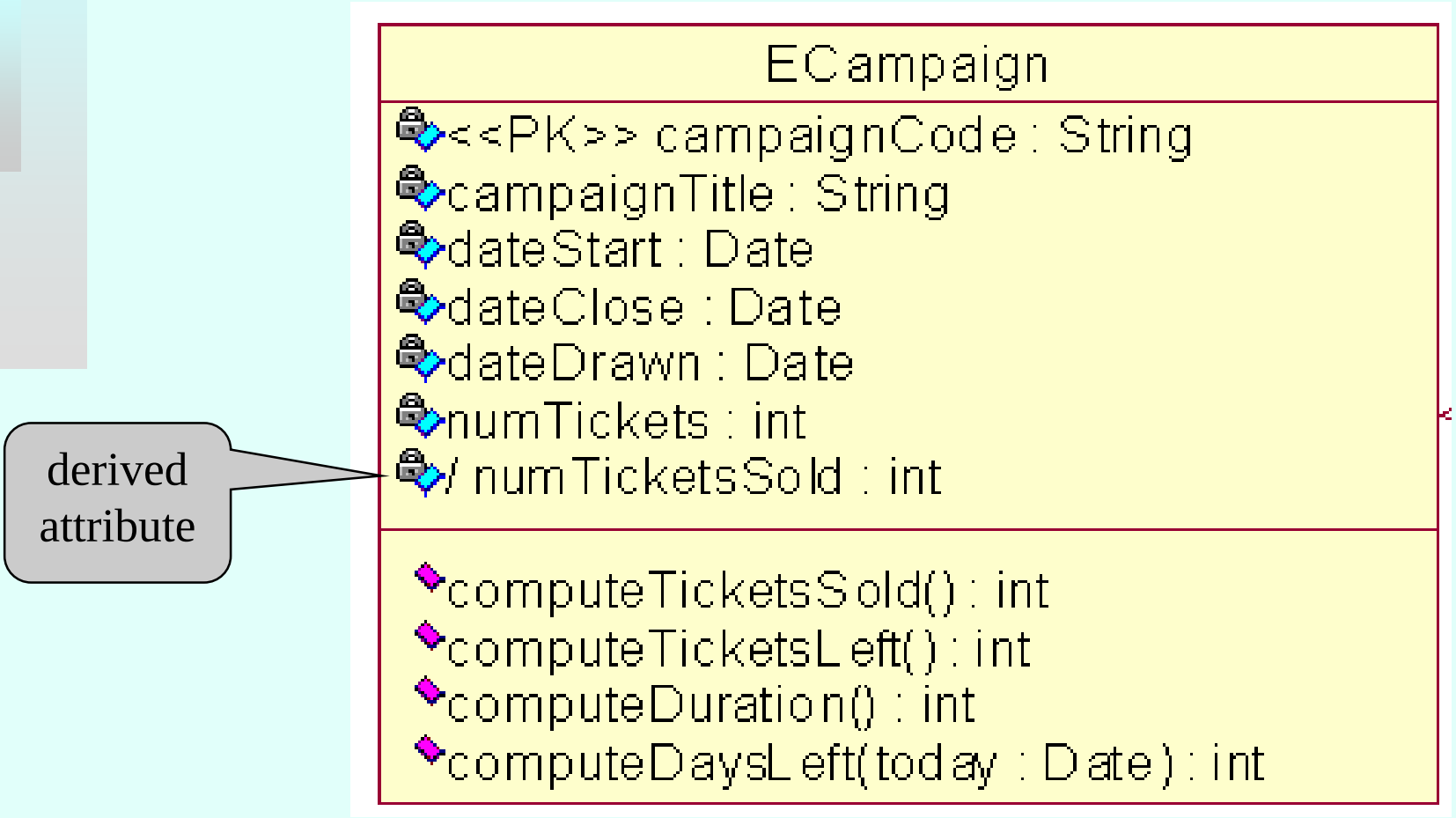
- visible to all other classes in the package
- protected (and public) give package access, but not vice versa
  - derived classes cannot access properties with package visibility if the derived and the base class are in different packages

```
package Campaigns;

class EBonusCampaign extends
ECampaign
{
   int ticketBookSize;

   public int getBookSize()
   {
    return 0;
   }
}
```

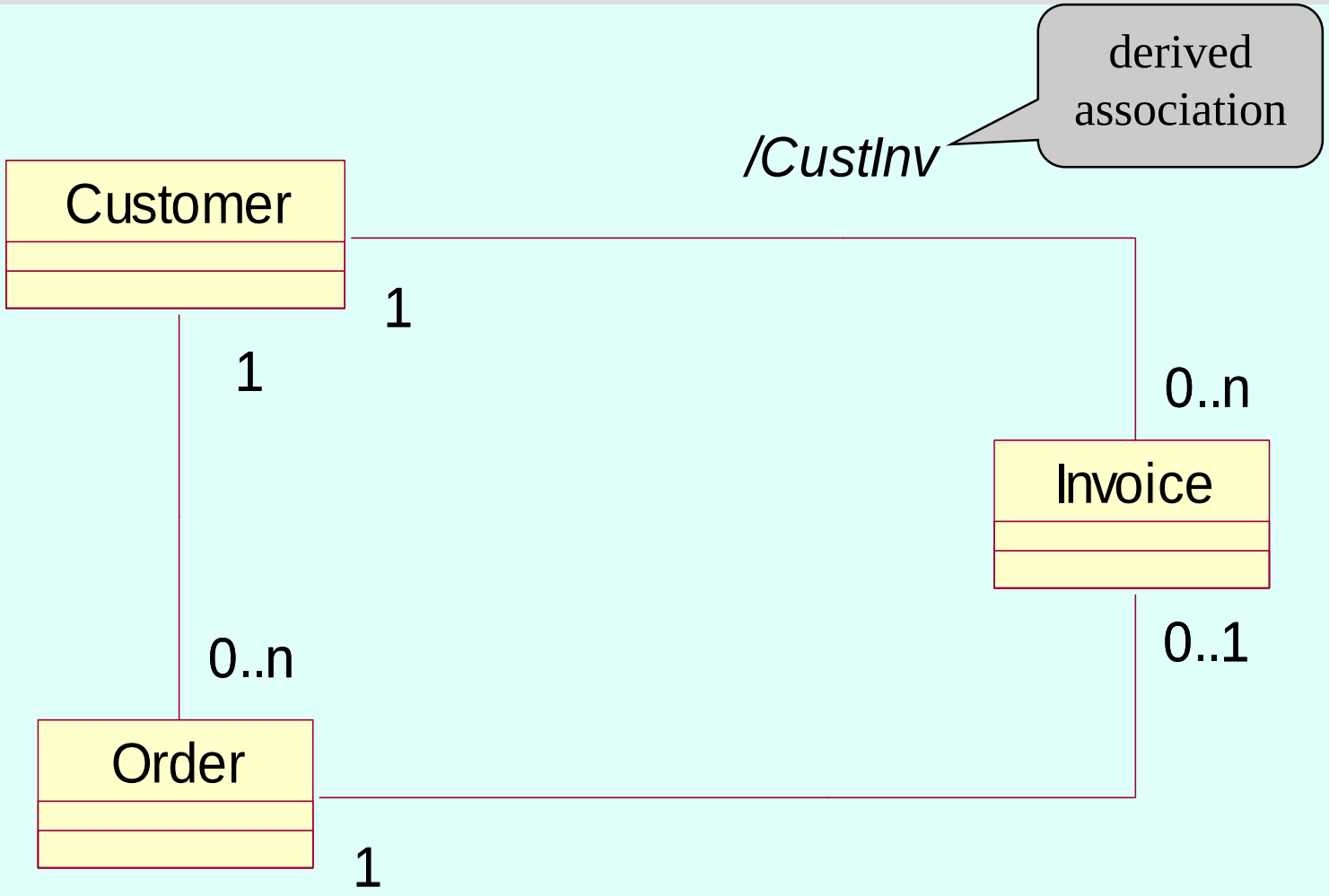
Derived information

- A kind of *constraint* that applies (most frequently) to an attribute or an association.
- Computed from other model elements.
- It does not enrich the semantics of an *analysis model*, it can make the model more readable.
- The knowledge of what information is derived is more important in a *design model*, where optimization of access to information needs to be considered.
- The UML notation for derived information is a slash (/) in front of the name of the derived attribute or association.

Derived attribute



Derived association



Review Quiz 4.1

1. What is the most important extension mechanism of UML?
2. How are role names called in UML 2.0?
3. What is the default visibility in Java (i.e. if the visibility is not specified)?
4. Can reified class replace association class without any loss of semantics?