Object Oriented Methodology

Week – *2*, Lecture – *1*

OOM at a Glance

SAURABH SRIVASTAVA

VISITING FACULTY

IIIT LUCKNOW

Domain vs Application Model

Domain Modelling refers to capturing the essence of a real-world domain

- This requires figuring out the Entities, their relationships
- The same model can be used to build multiple applications for that domain
- A Domain Model is created by people who “understand the domain”
- Example: If you have a family business, whatever your parents explained to you to about the enterprise

Application Modelling refers to the model used to build an actual Domain Application

- It is not necessary that all elements from the Domain Model, appear in an Application Model
- Application Model uses the concepts from Domain Model to create a blueprint for the Application
- We will attempt to learn skills of Application Modelling in this course …
- … assuming that someone has already prepared a Domain Model for us
- Example: The sketches you have come up with as part of automating some process of your enterprise

Types of Modelling

Modelling an Application requires understanding multiple aspects about the system

For instance, you need to know about the Entities of a system as well as their relationship

- For example, a Letter is associated with one and only one Postal Request

Also, you need to know the data in the system changes, i.e. what changes are allowed

- For example, a Letter can reach the “sealed” state from the “unsealed” state, but not vice-versa

Another aspect that you need to know about is the interaction semantics between the Entities

- For example, a Postal Request “sends” a Letter from its source, and “delivers” it to its destination

There are three different types of diagrams that capture these aspects of an Application

- We will study all three types one after another

The Class Modelling

Probably the most important aspect of a modelling step is performing the Class Modelling

Class Modelling refers to capturing *static* relationships of an Application through *Class Diagrams*

- Static relationships do not change with, or depend upon, time
- For example, the relationship between a Letter and a Postal Request remains the same – today or tomorrow
- The nodes of a Class Diagram are classes, and the edges represent the nature of their relationship

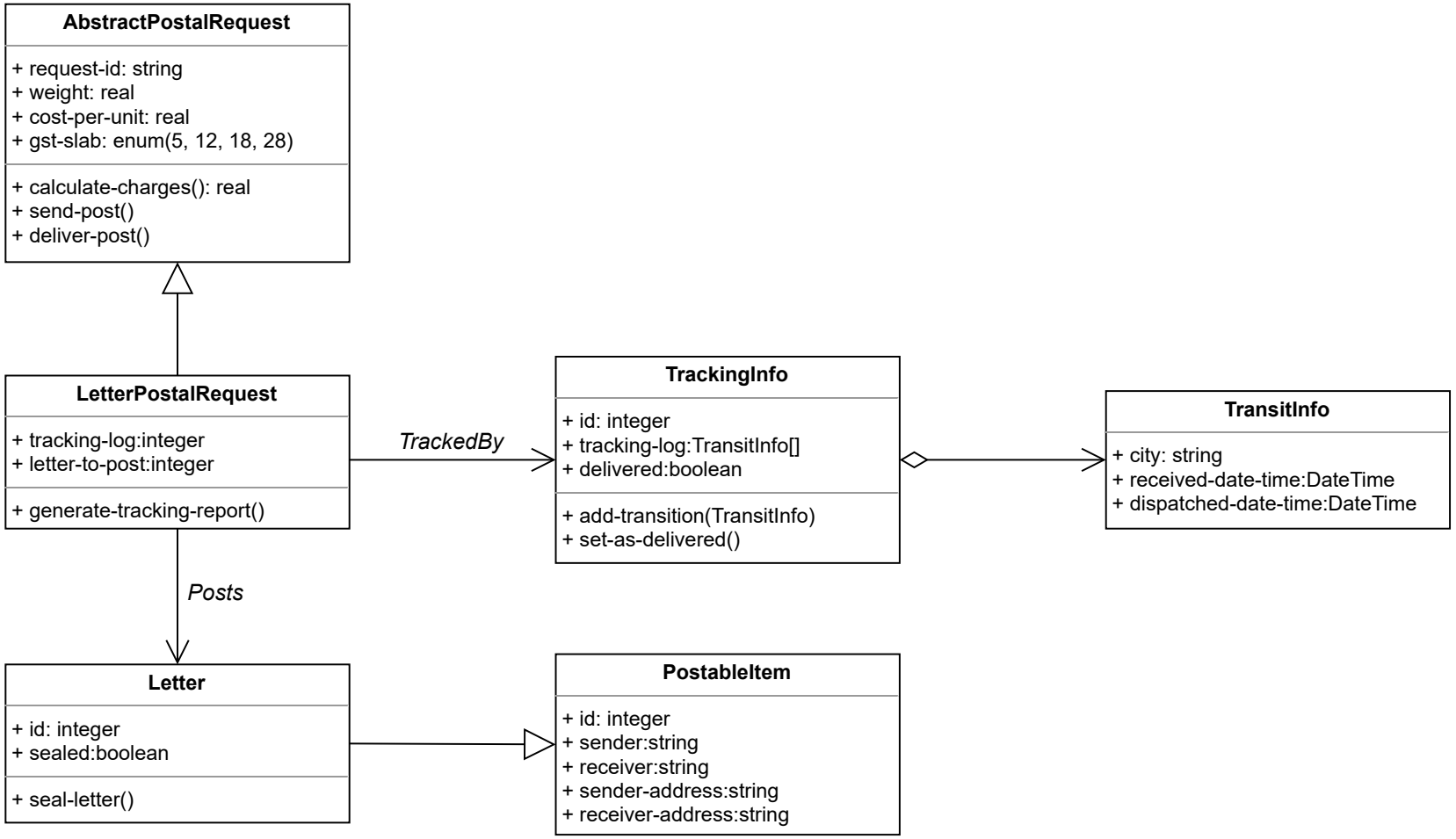
In general, any relationship between two classes is called an *Association*

Two special cases of Association, which are captured often in Class Diagrams are

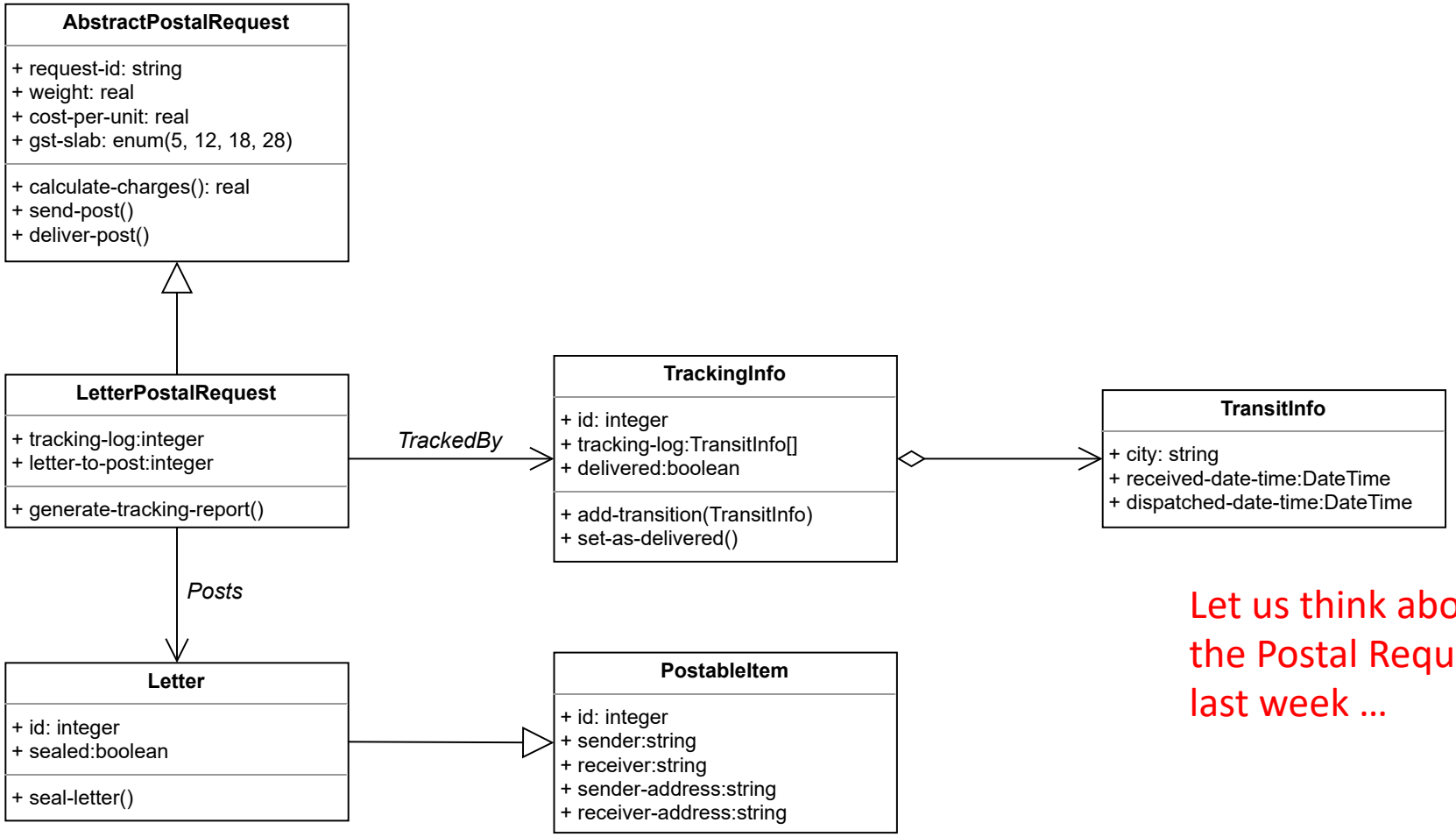
- The *part-of* relationship – when one Entity is a part of another Entity
- The *type-of* relationship – when one Entity is a specialised version of another Entity

I realized later that while *associations* can be used to represent the general relationship between any two classes, the *generalization-specialization* relationship is **not** considered as a type of *association* – it is an independent form of relationship. *Aggregation*, on the other hand, is considered a type of *association*.

Example Class Diagram

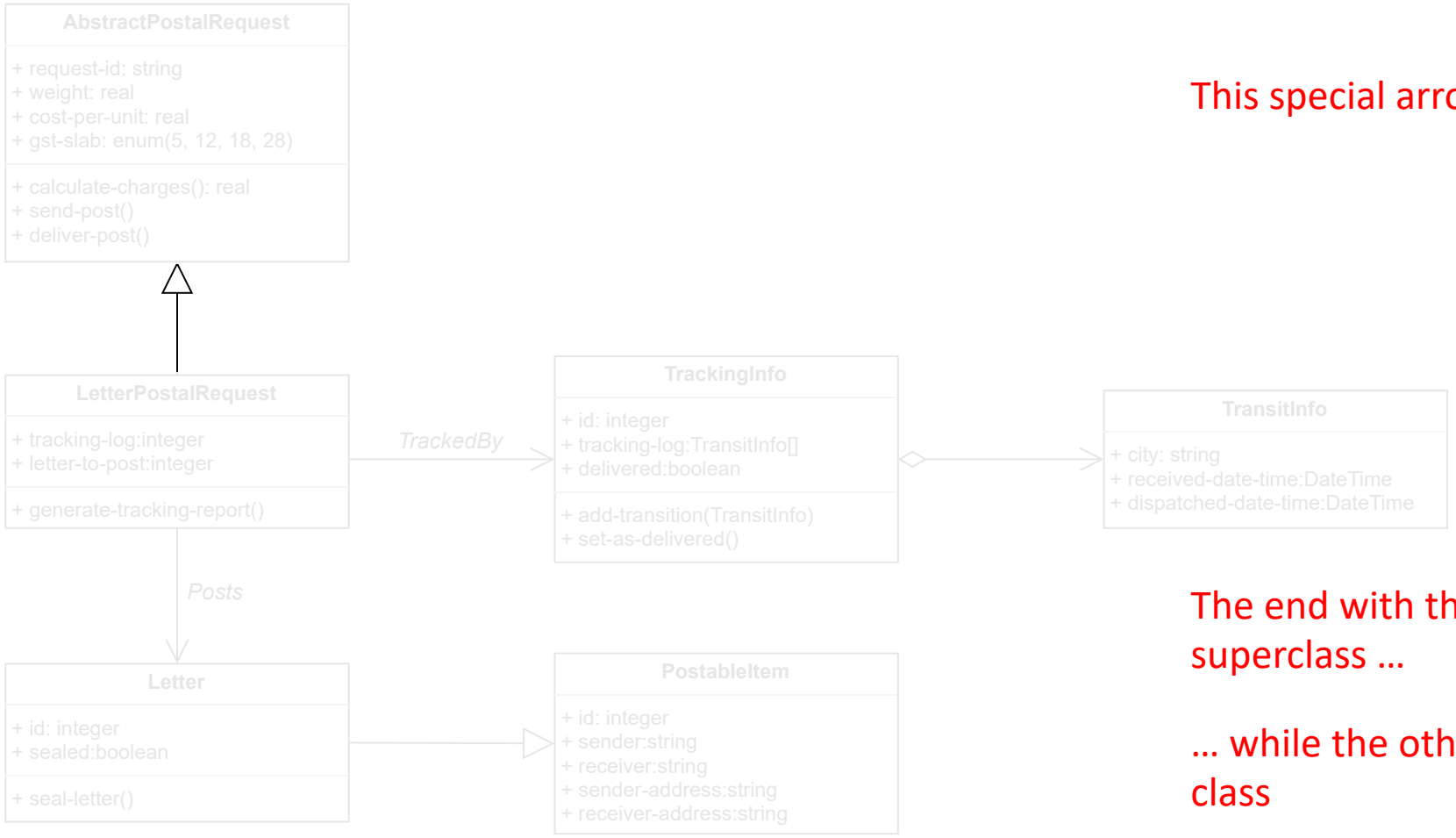


Example Class Diagram



Let us think about a more complex version of the Postal Request Example that discussed in the last week …

Example Class Diagram

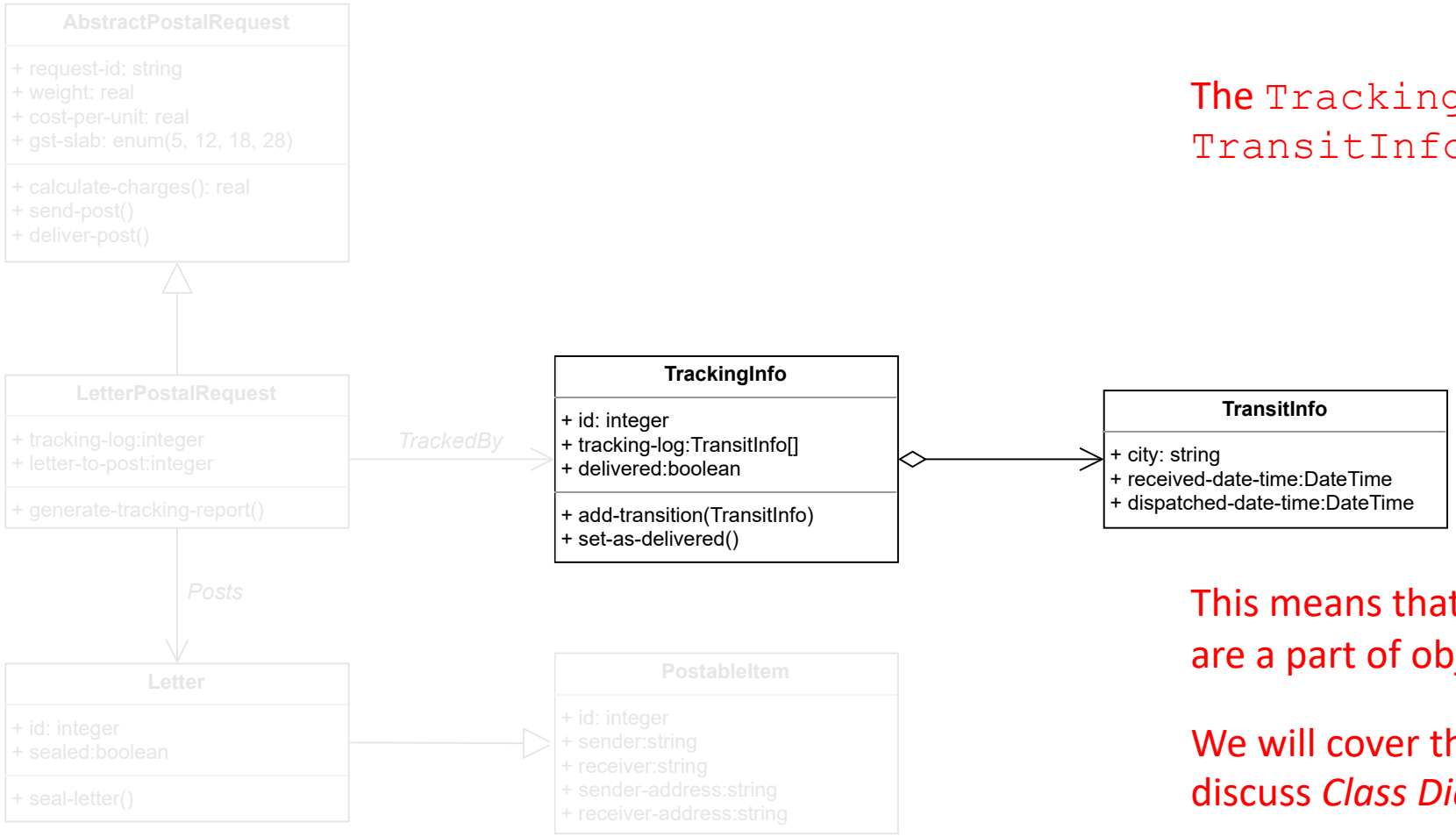


This special arrow indicates this relationship

The end with the hollow triangle represents the superclass …

… while the other end represents the derived class

Example Class Diagram

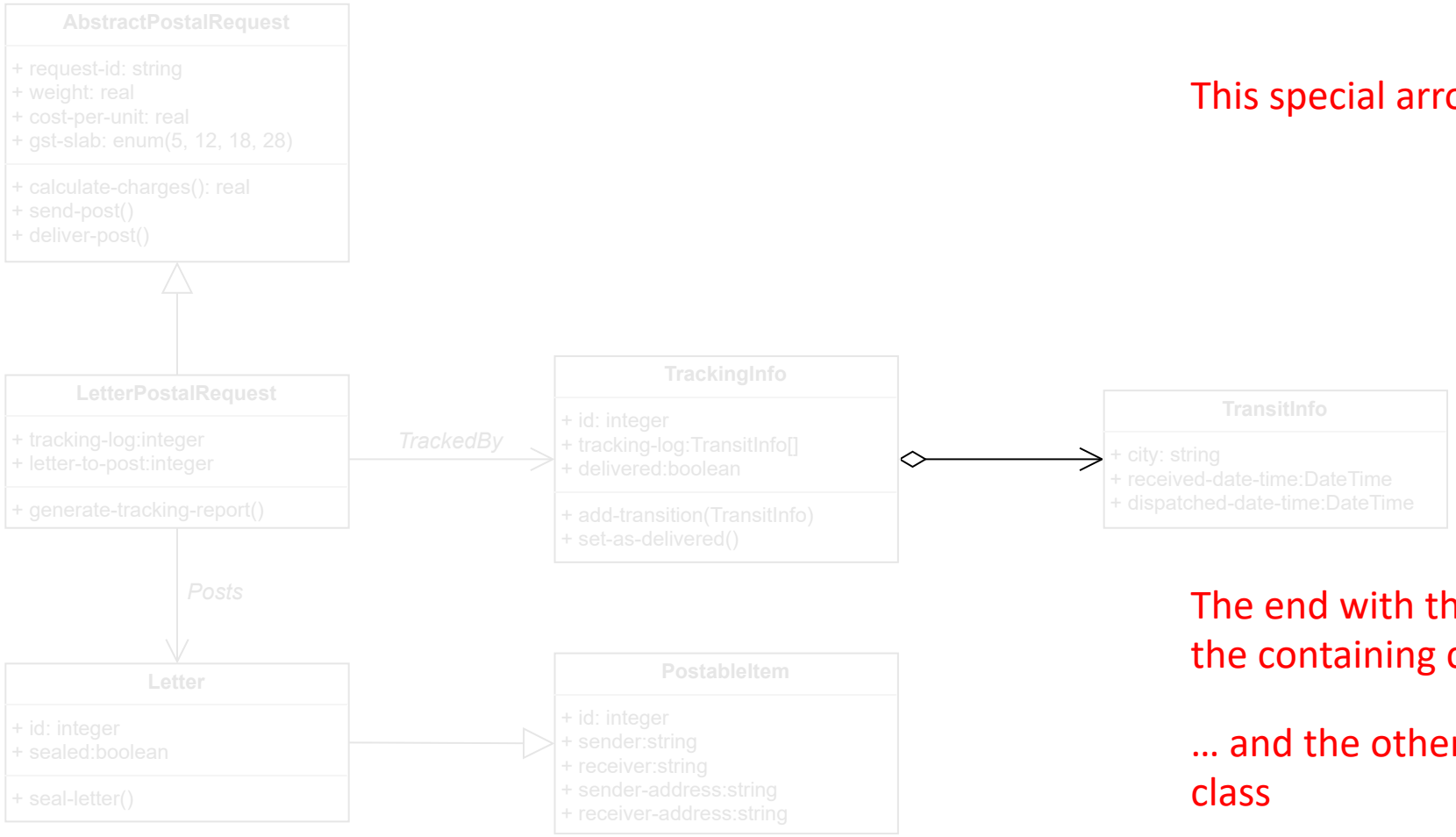


The `TrackingInfo` class *contains* instances of `TransitInfo`

This means that the objects of `TransitInfo` are a part of objects of `TrackingInfo`

We will cover this aspect at length when we discuss *Class Diagrams* and *Aggregation*

Example Class Diagram



This special arrow indicates this relationship

The end with the hollow diamond represents the containing class …

… and the other end represents the contained class

The State Modelling

The Class Modelling takes about how Entities of a system relate to each other

The State and Interaction Modelling involves capturing the runtime information of the application

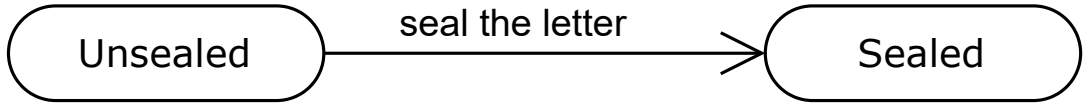
The State Model of an application consists of multiple *State Diagrams*

- One each for every Object type in the system (i.e. for every Class)
- If the number of Objects are too many, State Diagrams may be created only for some

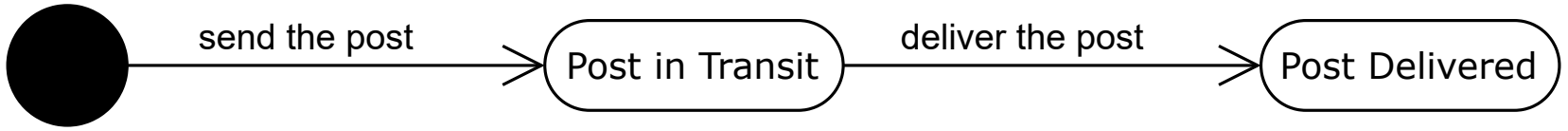
A State Diagram shows the transitions of a typical Object – from one state to another

- The nodes of a State Diagram are Object's States and edges are equivalent to some event
- The Object transitions from one state to another when the event occurs
- For the same Object, we may not show all state transitions and only choose the important ones

Example State Diagram

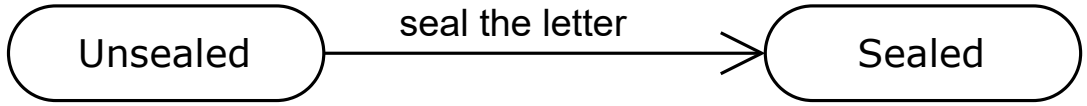


State Diagram for Letter

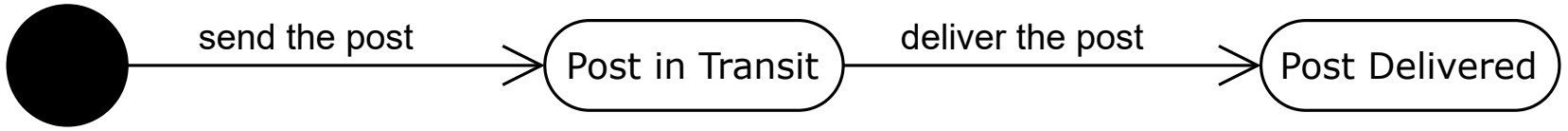


State Diagram for AbstractPostalRequest

Example State Diagram



State Diagram for Letter



State Diagram for AbstractPostalRequest

An Event could be something that happened in the real world …

… or something that is part of a chain reaction started by the same

Example State Diagram

State Diagram for Letter

State Diagram for AbstractPostalRequest

This is the symbol for the *start* state of an object (there is a similar symbol for the *end* too)

We will have a look at State Diagrams in more detail later in the course

The Interaction Modelling

The State Diagrams represent the internal changes of an Object

The Interaction Modelling is a complex task that tries to capture how Objects affect each other

- The overall idea is to capture the dynamicity of the application

Interaction Modelling may require many diagrams of different types

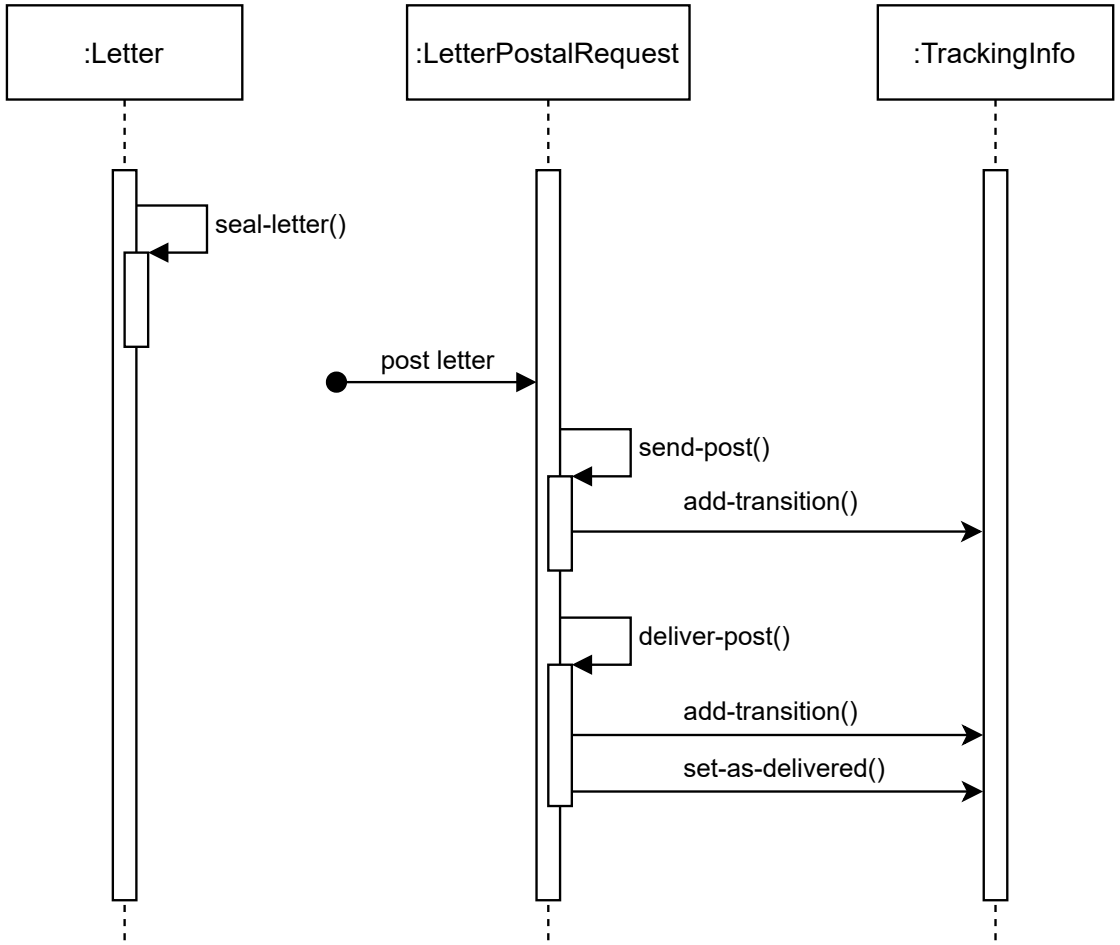
Three types of Interaction Diagrams that we will have a look in the course are

- Use Case Diagrams
- Sequence Diagrams
- Activity Diagrams

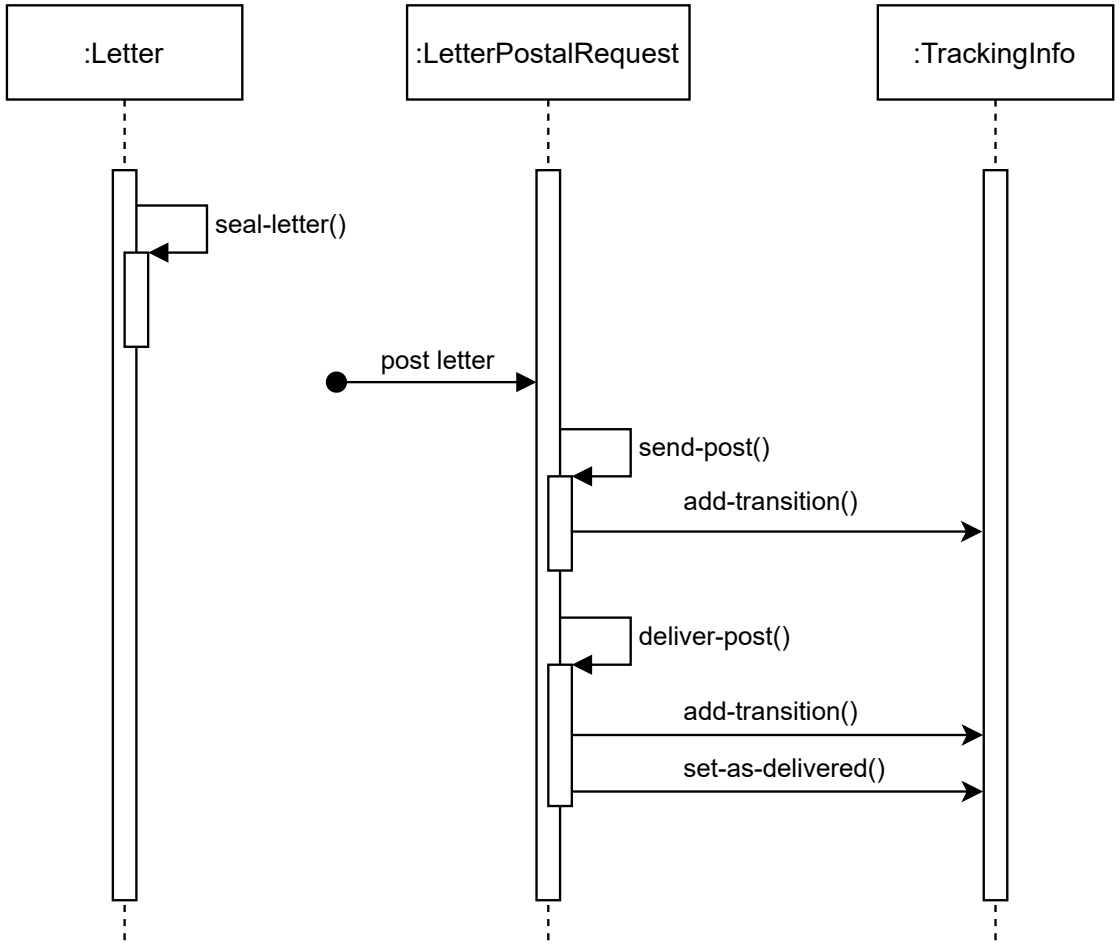
It is not necessary to create all these diagrams

- The goal is to communicate how the Application works while it is running

Example Interaction Diagram



Example Interaction Diagram

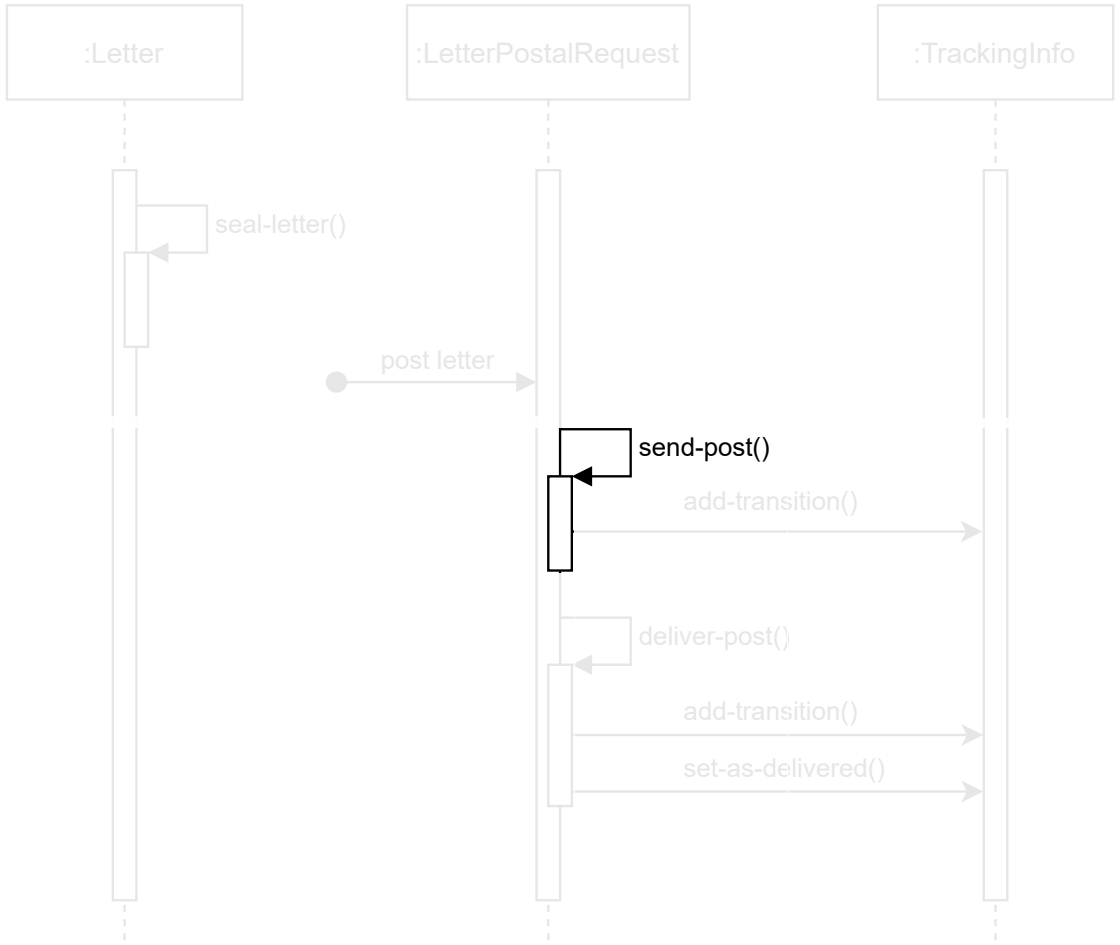


This is a *Sequence Diagram* to show some of the methods that are called when the Postal Example executes

The arrows that are higher in the diagram happen before those that are lower

The distance between the arrows do not matter (it does not represent time differences)

Example Interaction Diagram



This is an example of Event that is internal to an Object, but one that is important for the Application too

This is, in some sense, a loop

We will talk about Sequence Diagrams when we discuss Interaction Modelling in detail

Homework !!

Read Chapter 1 from the book by *Object-Oriented Modeling and Design*, by **Michael R Blaha** and **James R Rumbaugh**