

Introduction to Object Orientated Analysis and Design (OOAD)

Objektorienterad programutveckling, trådar och datakommunikation (DA343A)

Ben Blamey

Why Diagrams?

- Less precise than code.
- (Can be) more precise than text.
- Communicate ideas about a design.
- Document a implementation or specification.
- Help you to understand, debug, etc.
- This lecture looks at two diagrams:
 - Class Diagrams
 - Activity Diagrams

Object-Orientated Analysis

(from text)

- **Object Analysis - Step 1: Identify Entities**
- **Object analysis – Step 2: Scope**
- **Object analysis – Step 3: Class vs. Instance**
- **Object analysis – Step 4: Normalization**
 - Plural → singular
 - Capital first letter.
- **Object analysis – Step 5: Identify Attributes**
- **Object analysis – Step 6: Generalization (Inheritance)**
- **Object analysis – Step 9: Associations**

Till denna fråga modellerar vi ett system för att driva ett förenklat tågbolag "Skånetåg"

"Skånetåg är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokar sin resa. En passagerare har för- och efternamn, en e-postadress och omhen kräver en handikappanpassade sittplats(en boolean).

I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt."

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

Step 1: Identify Candidate Classes

"Skånetåg" är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokat sin resa. En passagerare har för- och efternamn, en e-postadress och omhen kräver en handikappanpassade sittplats (en boolean).

I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt.

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

Possible Classes:

SkåneTåg
Tågbolag
stationerna
passagerare
resa
biljetter
flyg
tåg
lok
Vagnar

Step 2: Scope

"Skånetåg" är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokat sin resa. En passagerare har för- och efternamn, en e-postadress och omhen kräver en handikappanpassade sittplats (en boolean).

I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt.

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

Classes:

SkåneTåg
Tågbolag
stationerna
passagerare
resa
biljetter
sittplats
~~flyg~~
tåg
lok
Vagnar

Step 3: Normalization

"Skånetåg är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokat sin resa. En passagerare har för- och efternamn, en e-postadress och omhen kräver en handikappanpassade sittplats (en boolean).

I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt.

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

Plural → singular
Capital first letter.

Tågbolag → Tågbolag
stationerna → Station
passagerare → Passenger
resa → Resa
biljetter → Biljett
tåg → Tåg
lok → Lok
Vagnar → Vagn

Step 4: Identify Attributes

"Skånetåg är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokar sin resa. En passagerare har för- och efternamn, en e-postadress och om hen kräver en handikappanpassade sittplats (en boolean).

I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt.

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

TågBolag

Station

Passengare
Förnamn : String Efternamn : String Epostadress : String KravHSP: Boolean

Resa

Biljett
Tid : DateTime Avgång : String Destination: String PrisSEK: Int

FysisktTåg

Lok

Vagn
VagnBokstav

SittPlats
VagnBokstav SittPlatsNummer

Step 5: Generalization

(None this time?)

Passengare
Förnamn : String Efternamn : String Epostaddress : String KravHSP: Boolean

TågBolag

Biljett
Pris: Int

SittPlats
VagnBokstav SittPlatsNummer

Resa
Tid : DateTime Avgång : String Destination: String

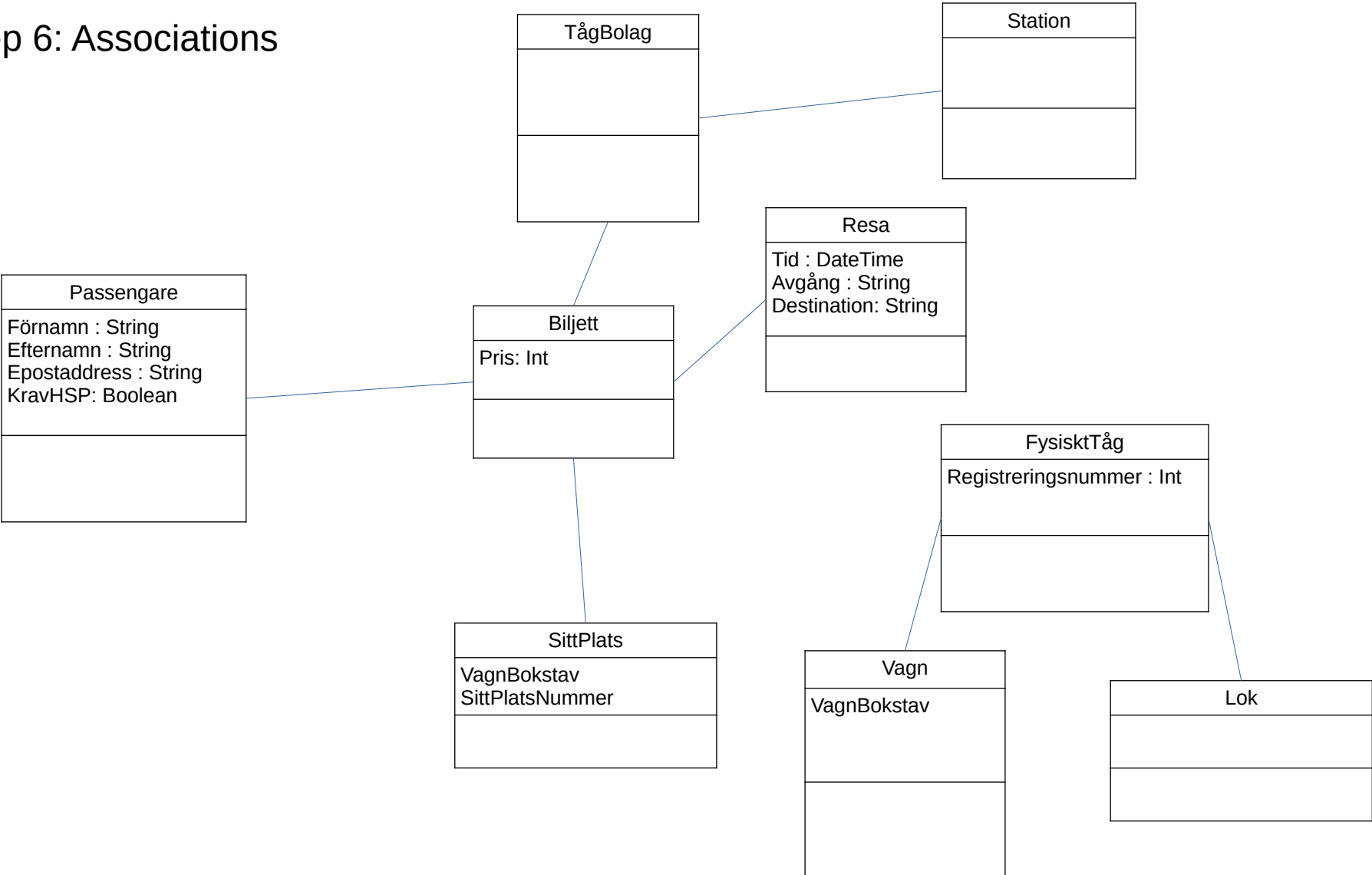
Vagn
VagnBokstav

Station

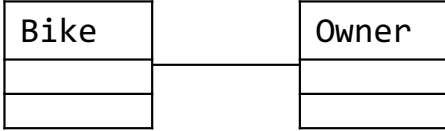
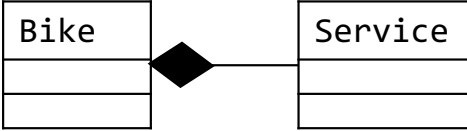
FysisktTåg
Registreringsnummer : Int

Lok

Step 6: Associations

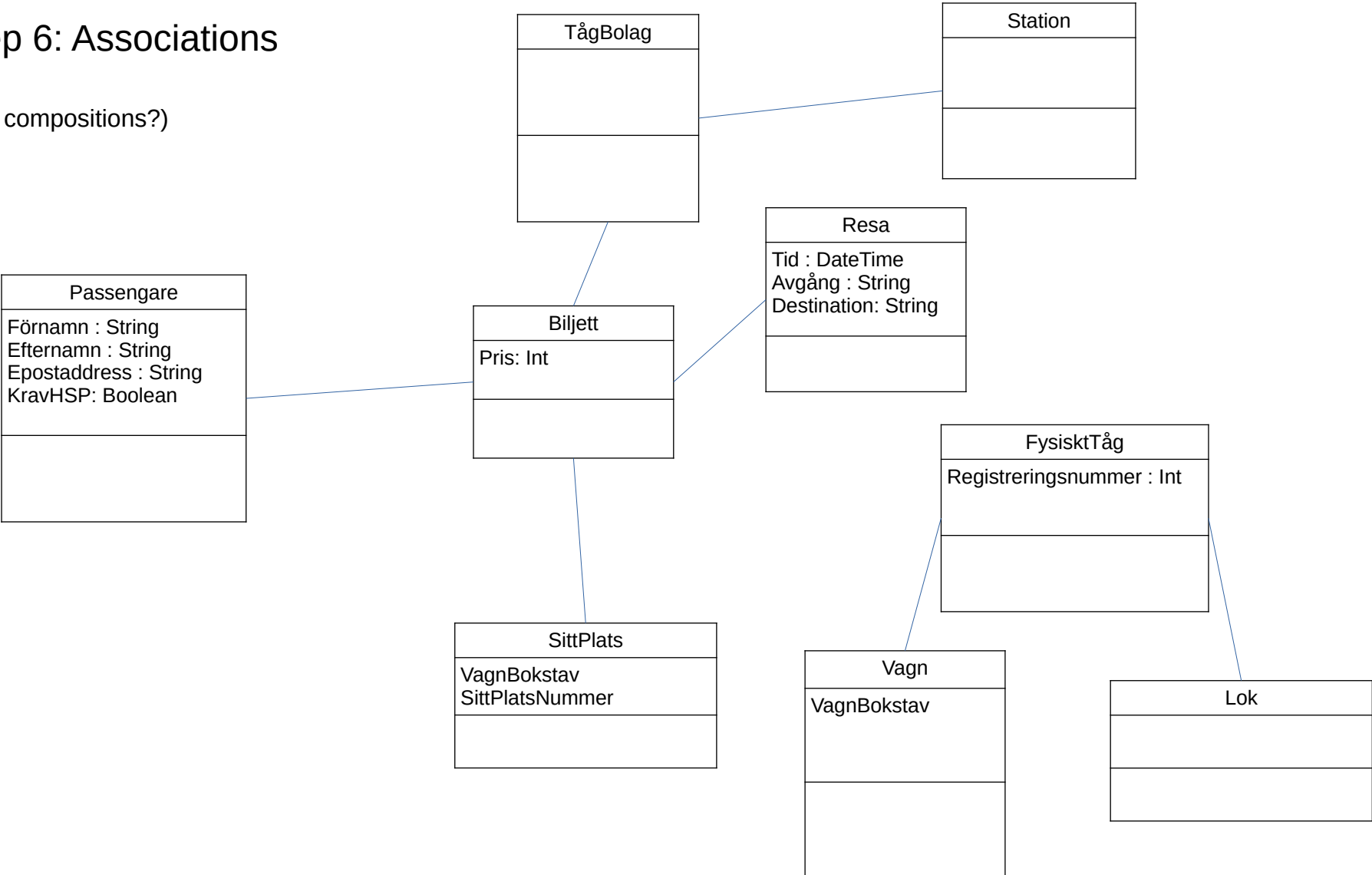


Object Relationships

	Association / Aggregation	Composition
A "Has a" B	✓	✓
Ownership "B cannot live without A"	✗	✓
UML Diagram		
Example	Bike "has an" Owner.	Service "strongly part of" Bike, and "cannot exist without it".

Step 6: Associations

(Any compositions?)



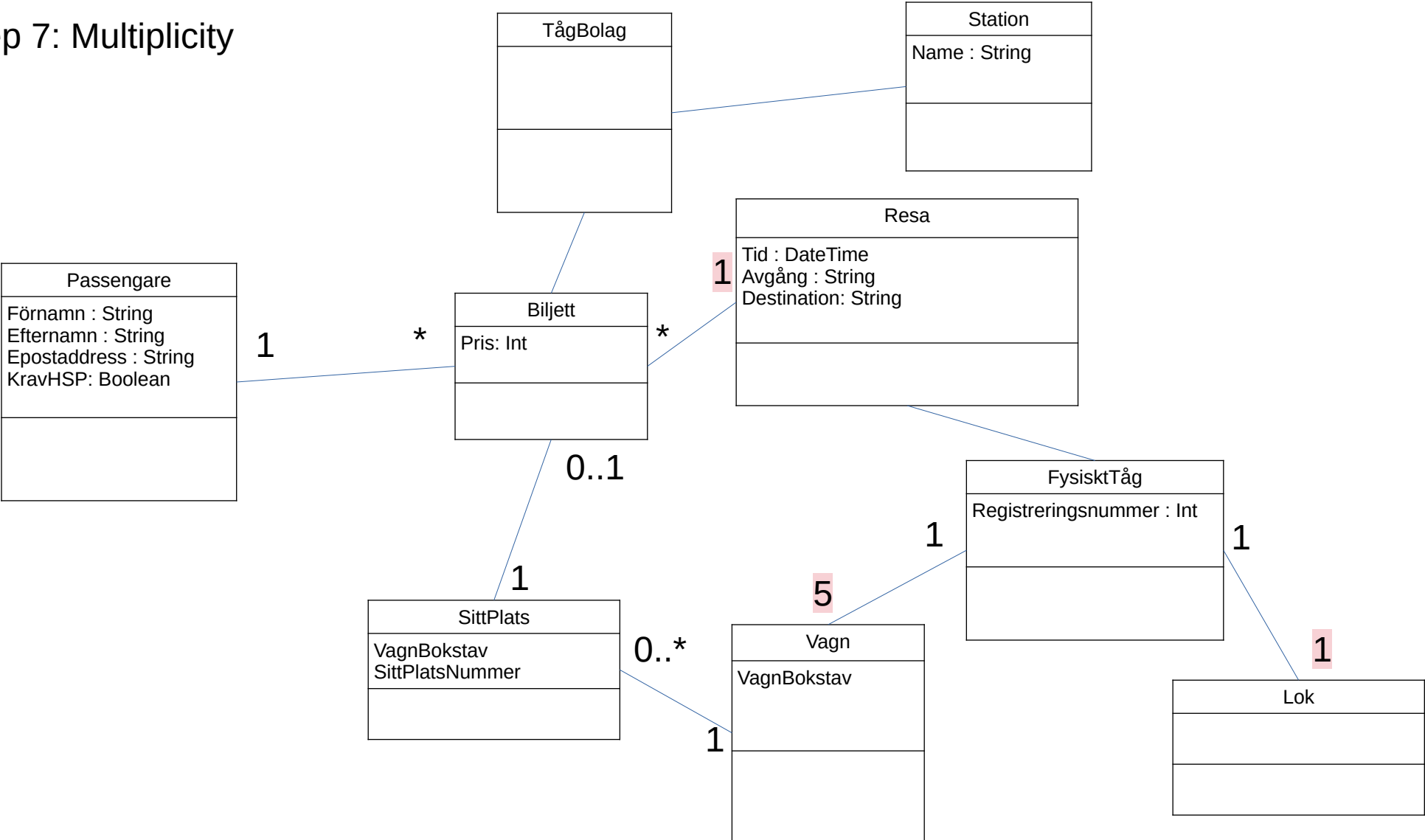
Step 7: Multiplicity

"Skånetåg är ett nytt tågbolag och bedriver initialt direkt tågtrafik endast mellan stationerna Malmö och Lund, i båda riktningarna. Skånetåg kräver att passagerare förbokat sin resa. En passagerare har för- och efternamn, en e-postadress och om hen kräver en handikappanpassade sittplats (en boolean).

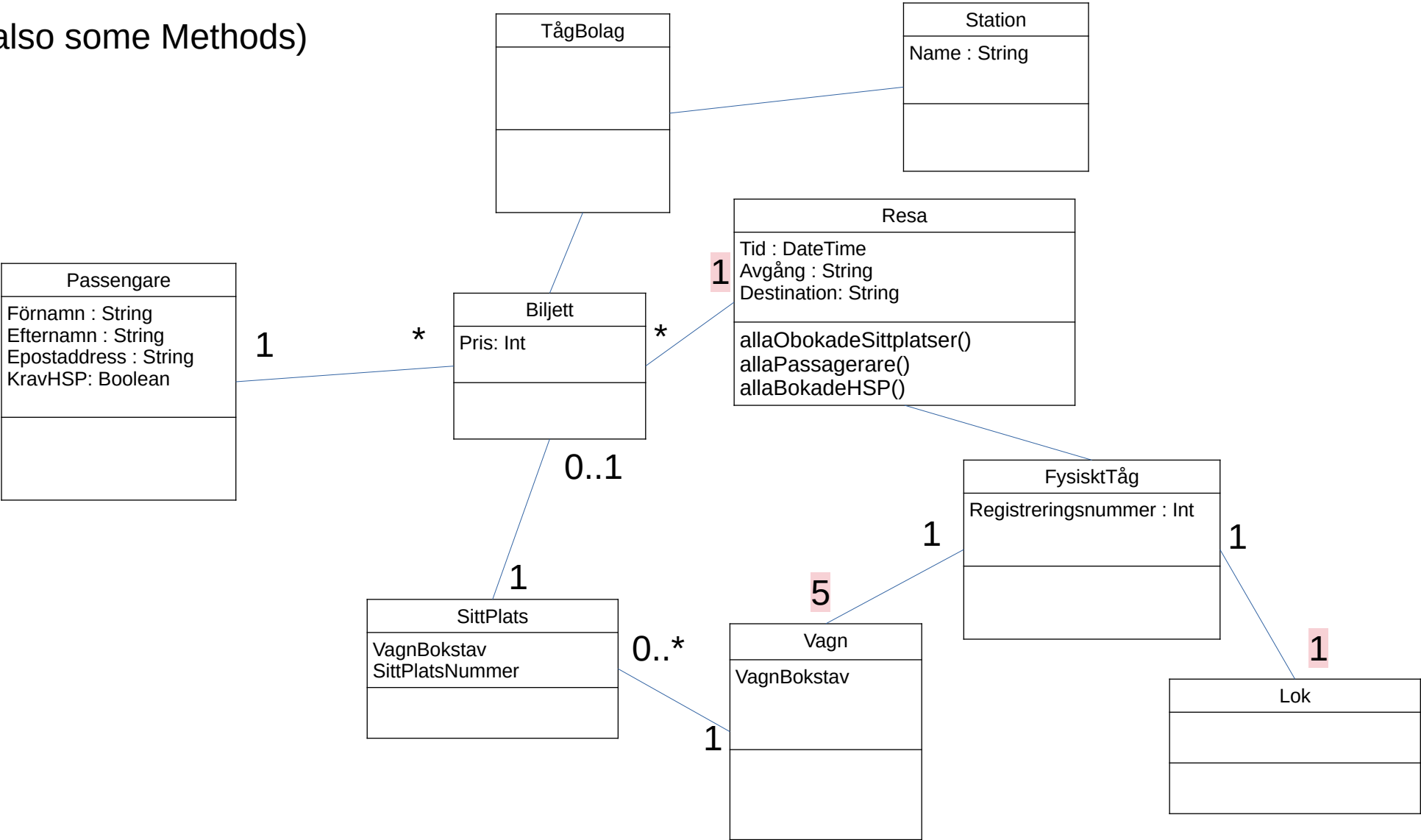
I systemet kan en passagerare flera biljetter (för tidigare, nuvarande och framtida resor) för resor. En biljett är alltid för 1 specifik resa (tid, datum och resans avgångs och destinations station) och har ett pris (i SEK). Modellen innehåller en metod för att på ett sökt tåg hämta alla obokade sittplatser, samt en metod för att hämta en lista över alla passagerare och deras platser. En annan metod hämtar en lista över alla passagerare som är bokade i handikappanpassade sittplatser och om de behöver assistans. (Metodimplementationer visas inte). Ett tåg har generellt lägre CO2-utsläpp än flyg, vilket gör dem mer miljömässigt hållbara. Sätena identifieras med deras vagnbokstav och sittplatsnummer. Ett fysiskt tåg består alltid av 1 lok och vagnar (alltid 5). Varje tåg har ett registreringsnummer och representeras som ett separat objekt.

Extrahera den relevanta informationen från texten och rita ett UML-klassdiagram (class diagram) för systemet, baserat på informationen ovan. Alla datum/tider kan representeras av instanser av typ "DateTime"

Step 7: Multiplicity

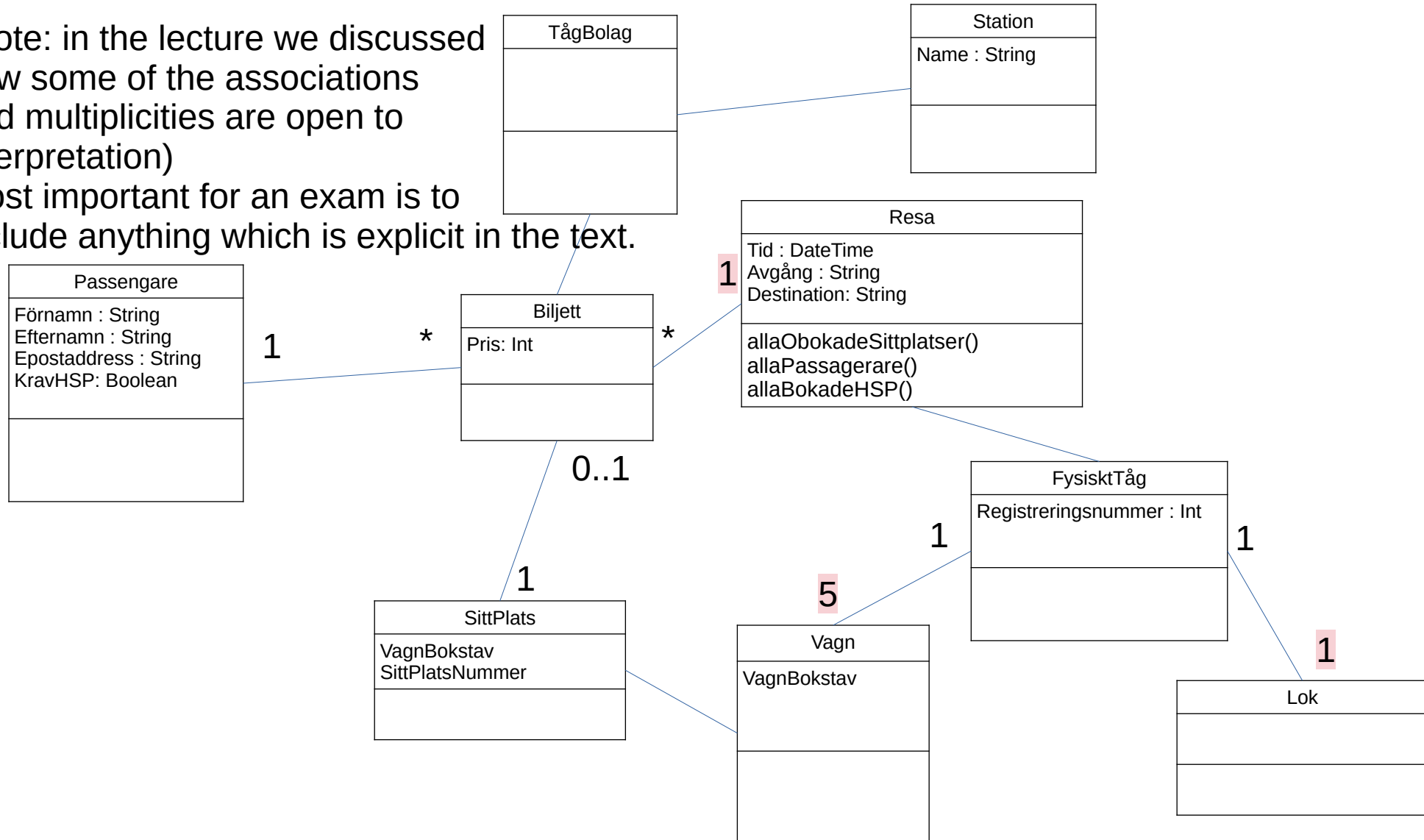


(...also some Methods)



(Note: in the lecture we discussed how some of the associations and multiplicities are open to interpretation)

Most important for an exam is to include anything which is explicit in the text.

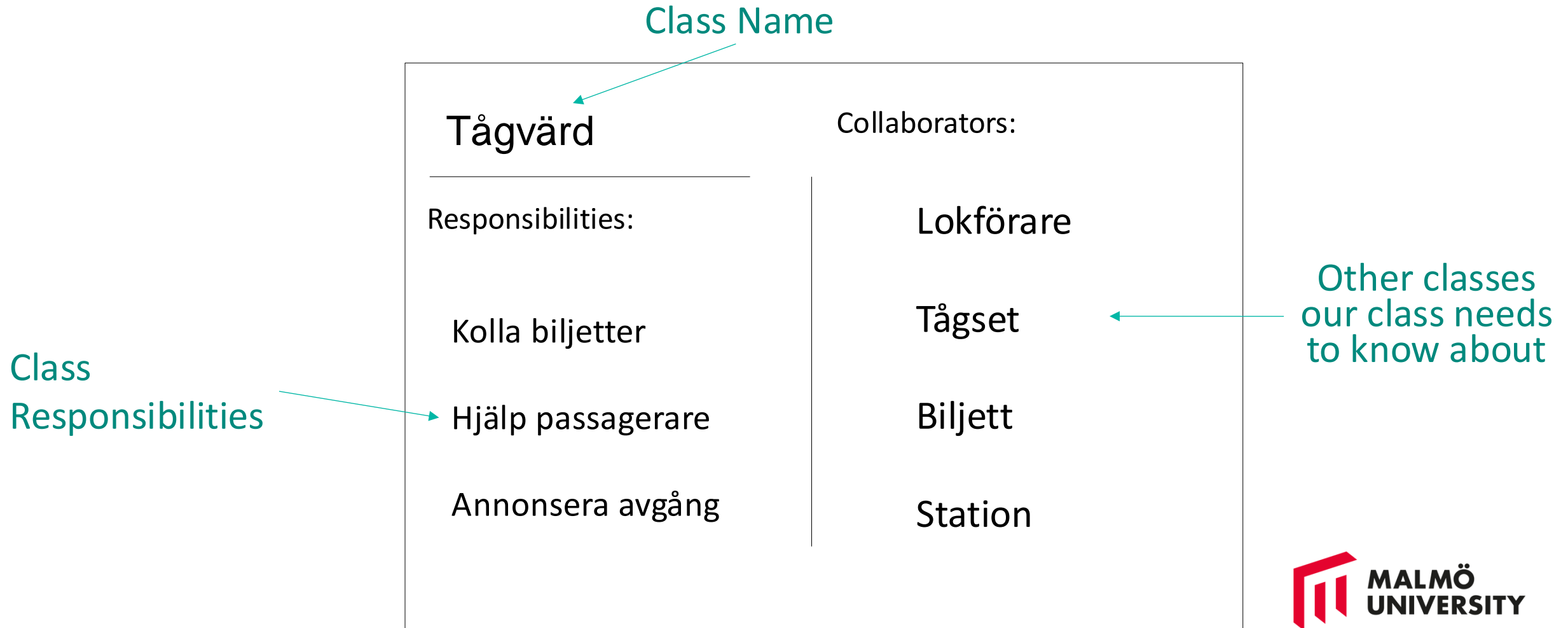


What about Methods?

- In the example, we lack information about what the different classes should or can do.
- We have only described the state and relationships (like a database).
- We can also look at another way of modelling.....

Class-Responsibility Collaborator (CRC) Card

CRC (Class-Responsibility Collaborator)



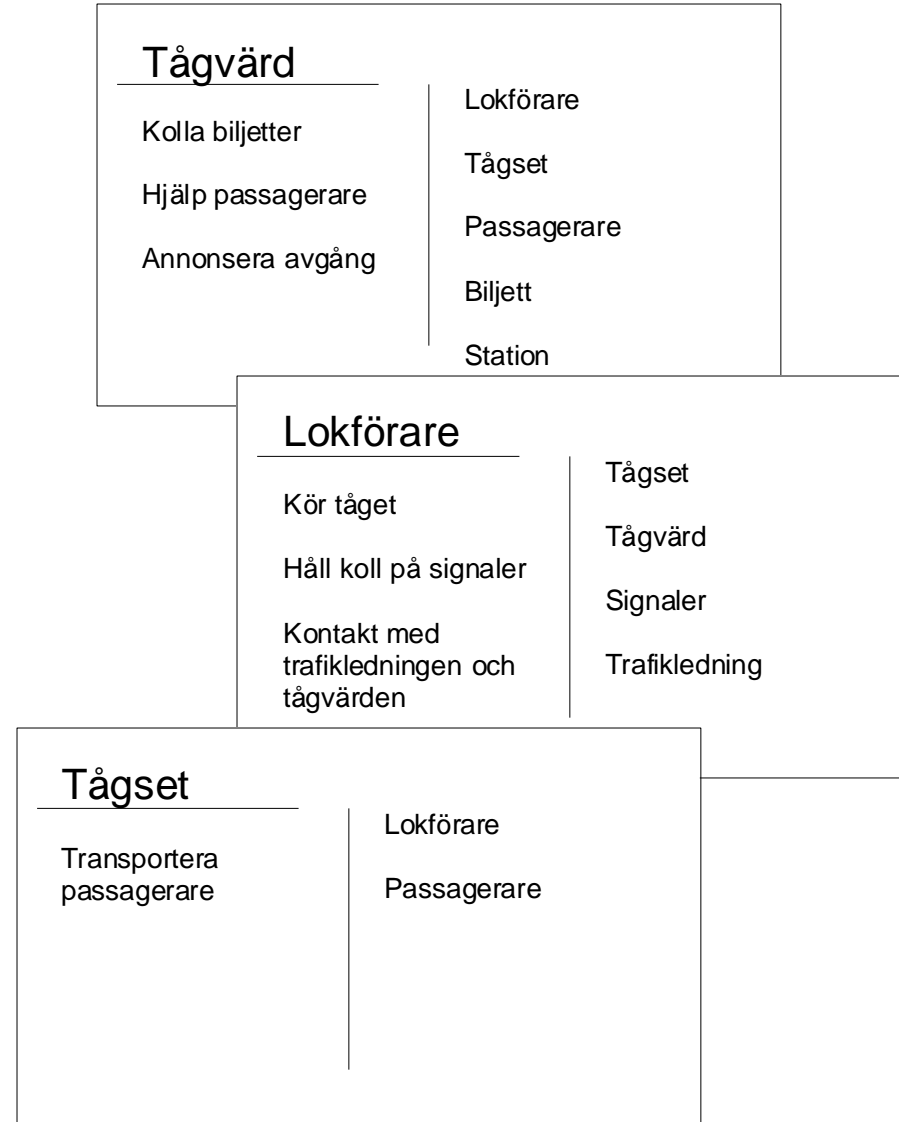
CRC

Johan commutes with Pågatåg to work. When he gets on the train, he shows his ticket to the train attendant, who checks that it is valid. If not, Johan will be fined.

Before the train sets leave the station, the train attendant announces departure by blowing his whistle. They also help the passengers on board. In the event of delays, it is the train attendant who tells them what is happening, and therefore has contact with the train driver.

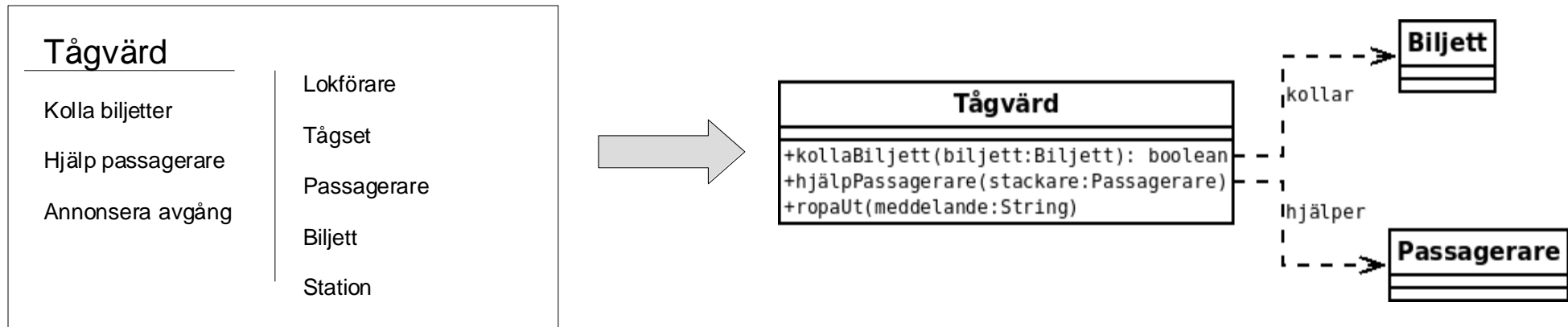
It is the train driver's job to drive the train. That responsibility includes keeping track of signals and maintaining contact with traffic management.

You already know this.



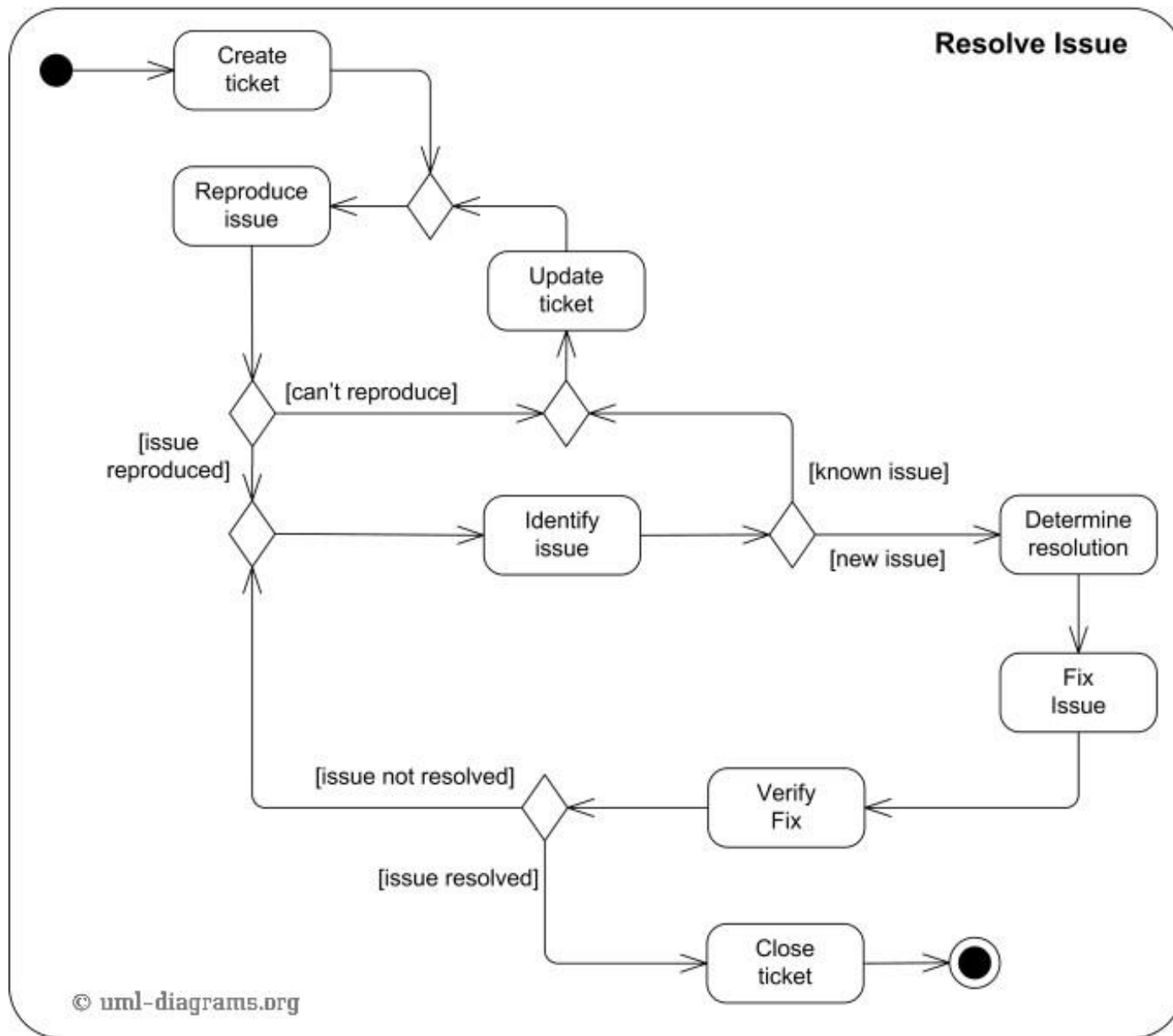
Methods?

The responsibilities of the CRC cards are good candidate methods. For example:



Break?

Activity Diagrams



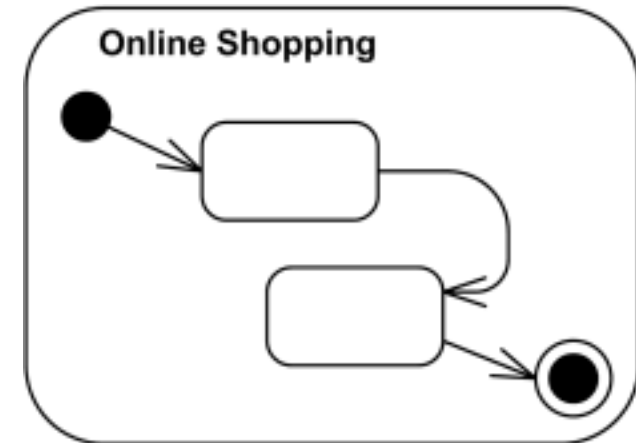
Activity diagram

These show the flow of control in a system or subsystem.

Each diagram shows what a single use case looks like in the system, but can refer to other use cases.

Activity (Aktivitet)

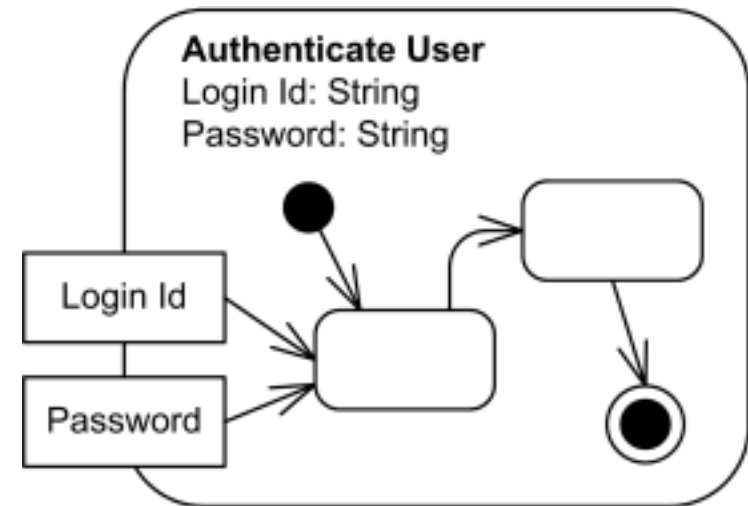
An activity is a flow of events. It is identified by a name, which often corresponds to a use case in a Use Case diagram.



Parameters

We can set **parameters** to an activity. These are modelled as rectangles at the edge of the activity.

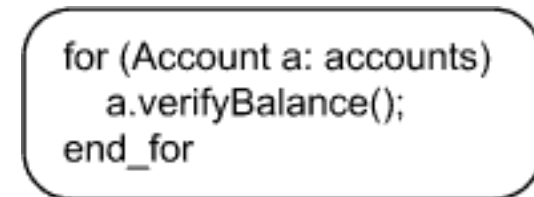
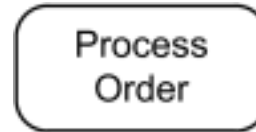
Parameters are inputs that are needed to perform a given activity.



Actions (Handlingar)

The basic component of an activity graph is the action (or action). An action is an atomic step in an activity, which means that it cannot be broken down into smaller parts.

Actions are named in the same way as use cases, but avoid reusing the names within a given system.

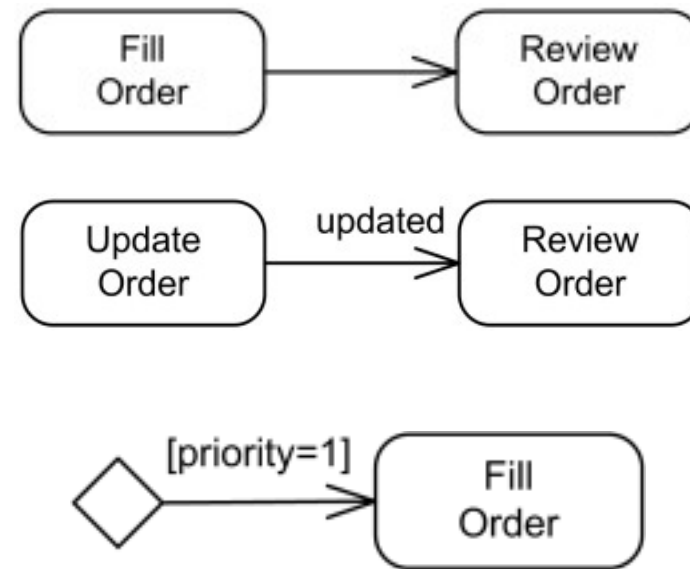


An action described in a language given for the application.

Edges (Bågar)

We connect actions with arrows called edges.

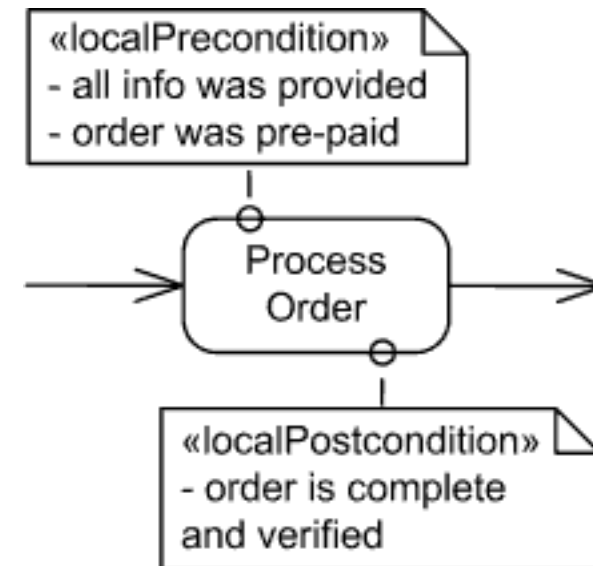
The edges are directional and may be marked with a name or a guard above the arrowhead.



Local conditions

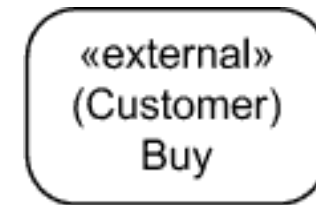
We can provide an event with **local preconditions** that must be met for the event to take place. In the same way, we can also specify the conditions required for an event to be counted as completed.

These are modelled as comments with the stereotypes `localPrecondition` and `localPostcondition`.



External Actions (Yttre handlingar)

External actions are actions that, strictly speaking, do not belong to the current chart. They are often, but not always, specified in another diagram in the system description.

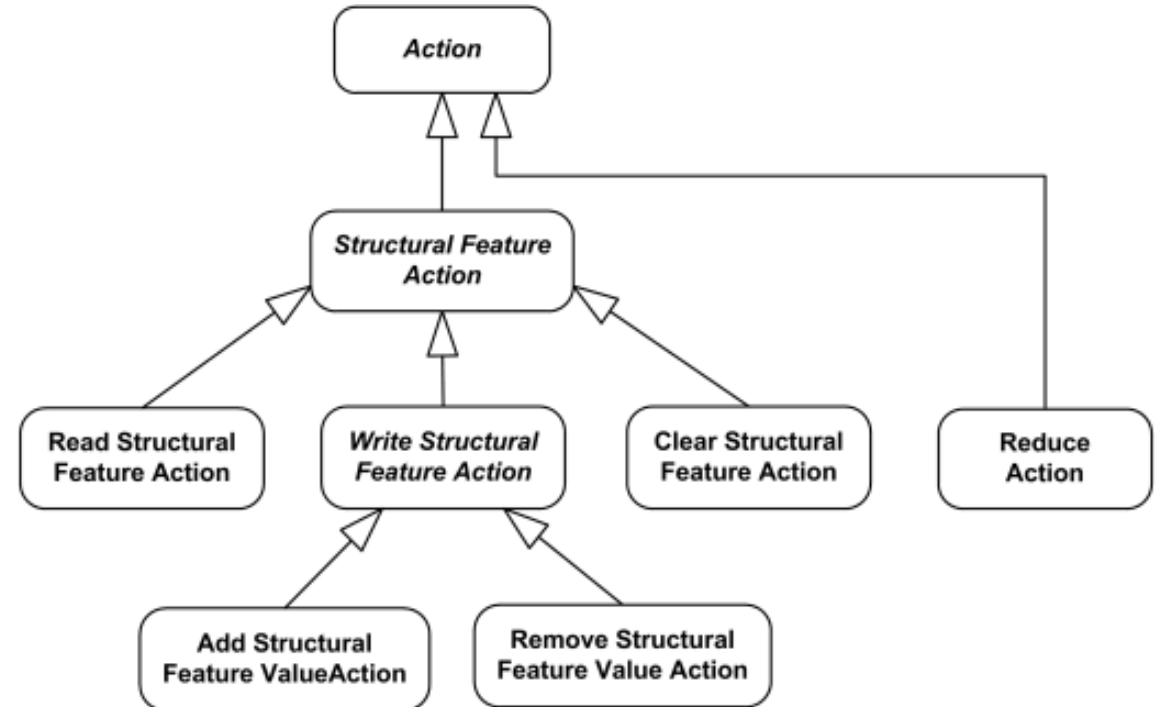


Outward actions are marked with the stereotype external.

Generalizations

We can **generalize** actions by modelling them in the same way as inheritance in class diagrams.

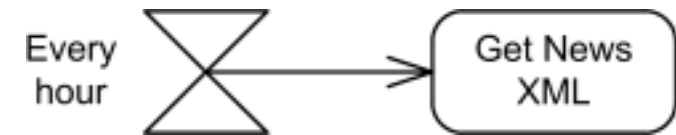
Abstract actions can be modelled by italicizing the name.



Periodic operations

Periodic actions are actions that take place at regular intervals and therefore are not necessarily initiated at the initiative of a user.

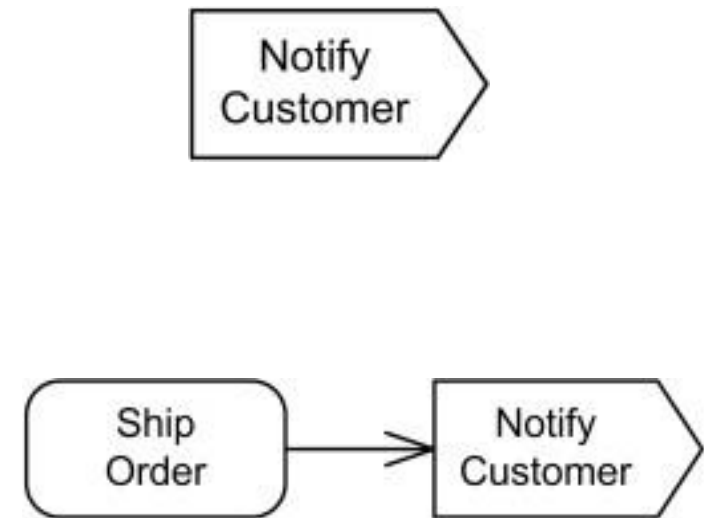
These actions are modelled with a stylized hourglass.



Signaleringar, skicka signal

Signaling is used to connect different workflows. A common view of this is that our flow has side effects that are modelled elsewhere.

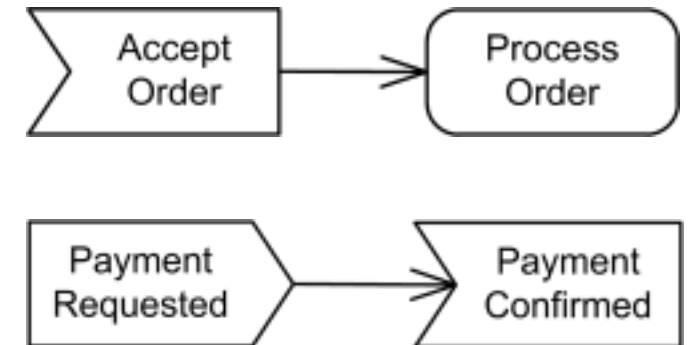
We show that an event sends a signal with a **pointed box**.



Receiving Signals

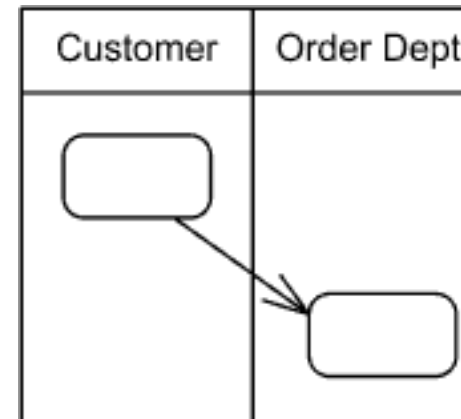
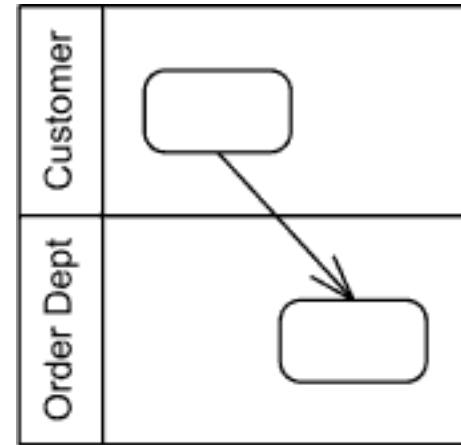
In the same way that we send signals, we can also receive signals in our models. We show this with a "bitten off" rectangle.

We can also model jumps in and out of a graph by using a sent signal and a received signal next to each other.



Partitioner / Swimlanes

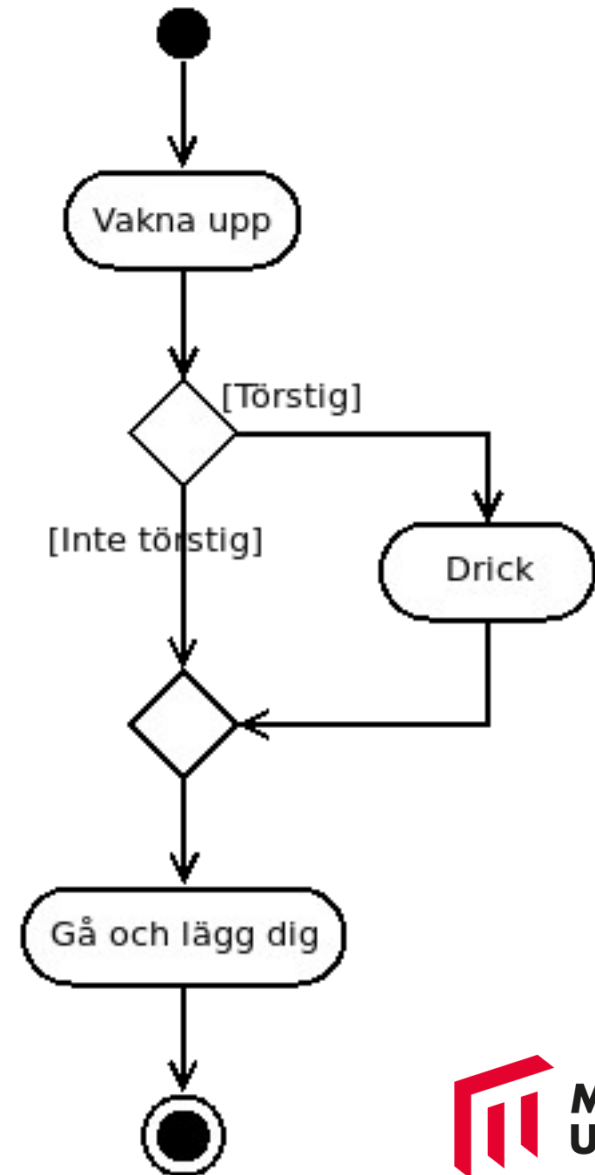
We can clarify where events belong by using partitions. These are modelled as swim lanes, which can be both horizontal and vertical.



Control Nodes

To be able to model more complex chains of events than purely linear ones, we use **control nodes**.

These are used to model, for example, if statements, loops, and so on.

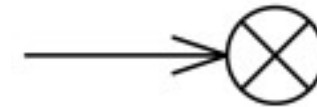


Start/Final

The **start node** shows, where a flow starts.

The flow **final node** shows that a partial flow is terminating. It can be used to model partial flows in a larger flow that requires, for example, a signalling or periodic event in order to be completed.

The activity **final node** shows, just like in a state diagram, where a flow ends.

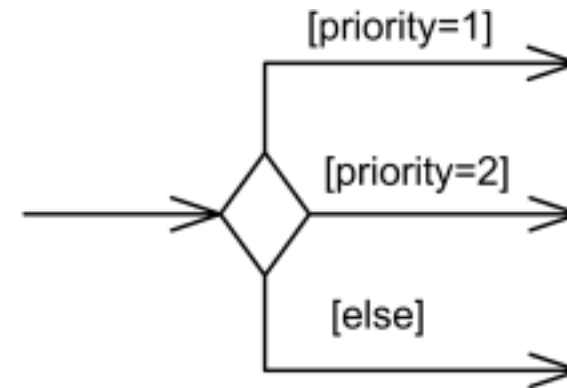
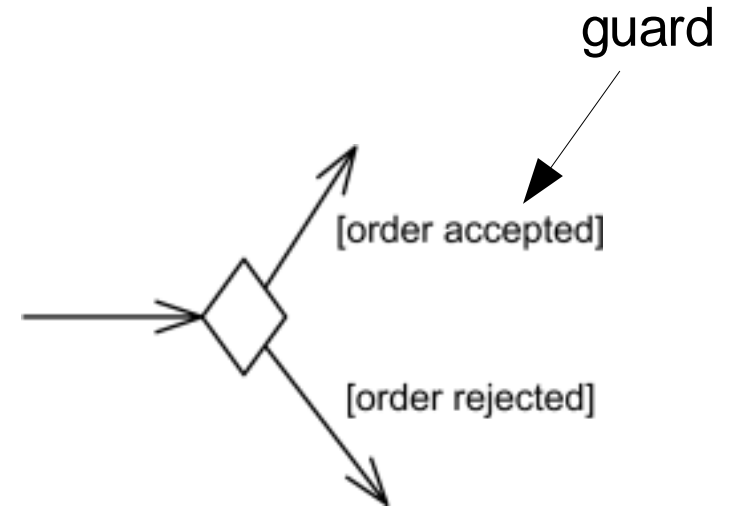


Decision Nodes

Decision nodes are used to model if statements and similar control structures, such as select/switch statements.

Decision nodes can have two or more branches.

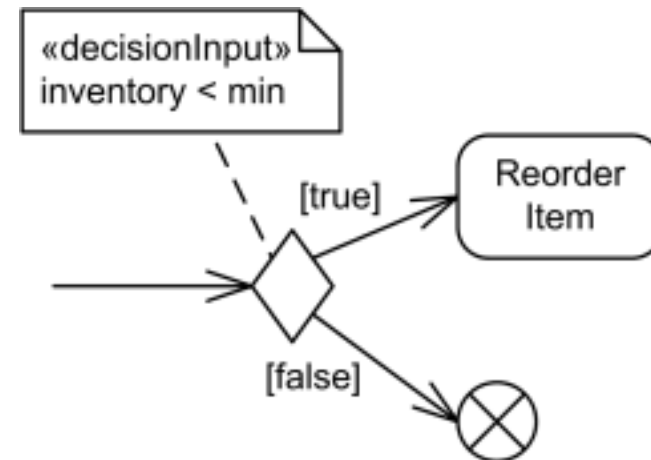
Each branch is described by a guard. The guards are evaluated to determine which branch to follow.



Decision Nodes, Input

We can model inputs (conditions) in a decision node with a comment connected to the decision node with a dotted line.

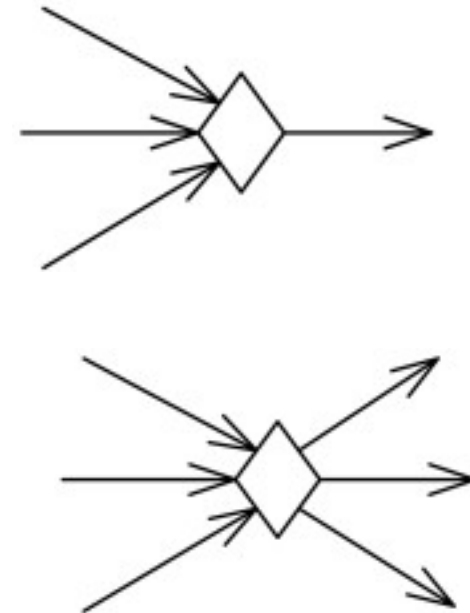
The input is marked with the stereotype `decisionInput`.



Merge Nodes

Merge nodes are used to model two or more paths from a decision node reaching the same place in the flow.

These nodes can also be decision nodes for the next step in the flow.



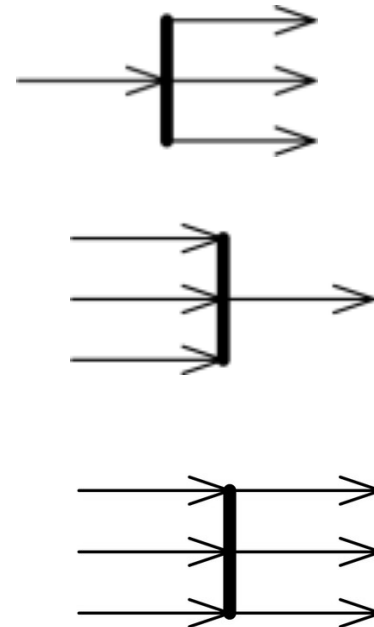
Fork nodes and Join Nodes

(Avgreningar och anslutningar)

Forks show that two or more sub-flows occur in parallel. These sub-flows are independent of each other and are grouped together in a join node.

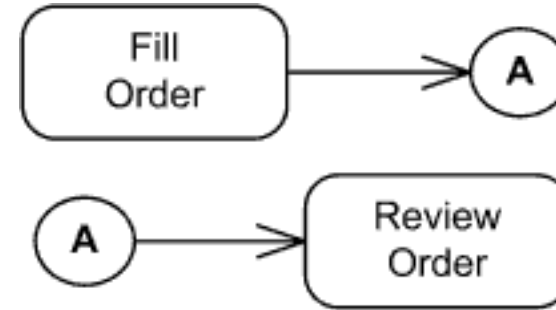
A join node synchronizes multiple flows.
(waits for them all to finish).

As with merge points, a point can also constitute a new branch.



Connectors

The connector connects two parts of the flow. It has no other meaning than that you don't have to draw lines across the entire chart.

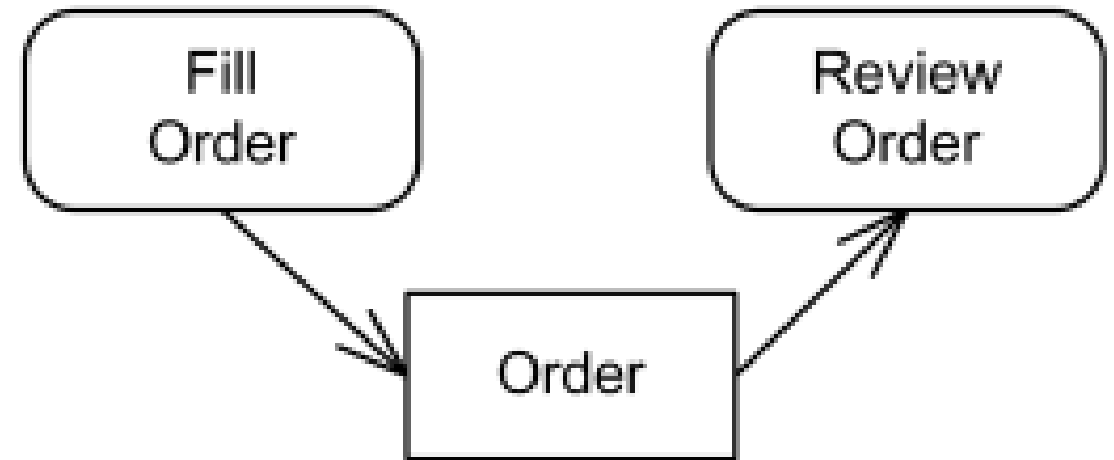


Equivalent to...



Object flows

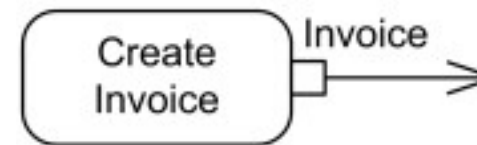
An object flow shows that the data sent from one action to another is an object. These are modelled with named rectangles as in an object diagram (we will go through this in the next lecture).



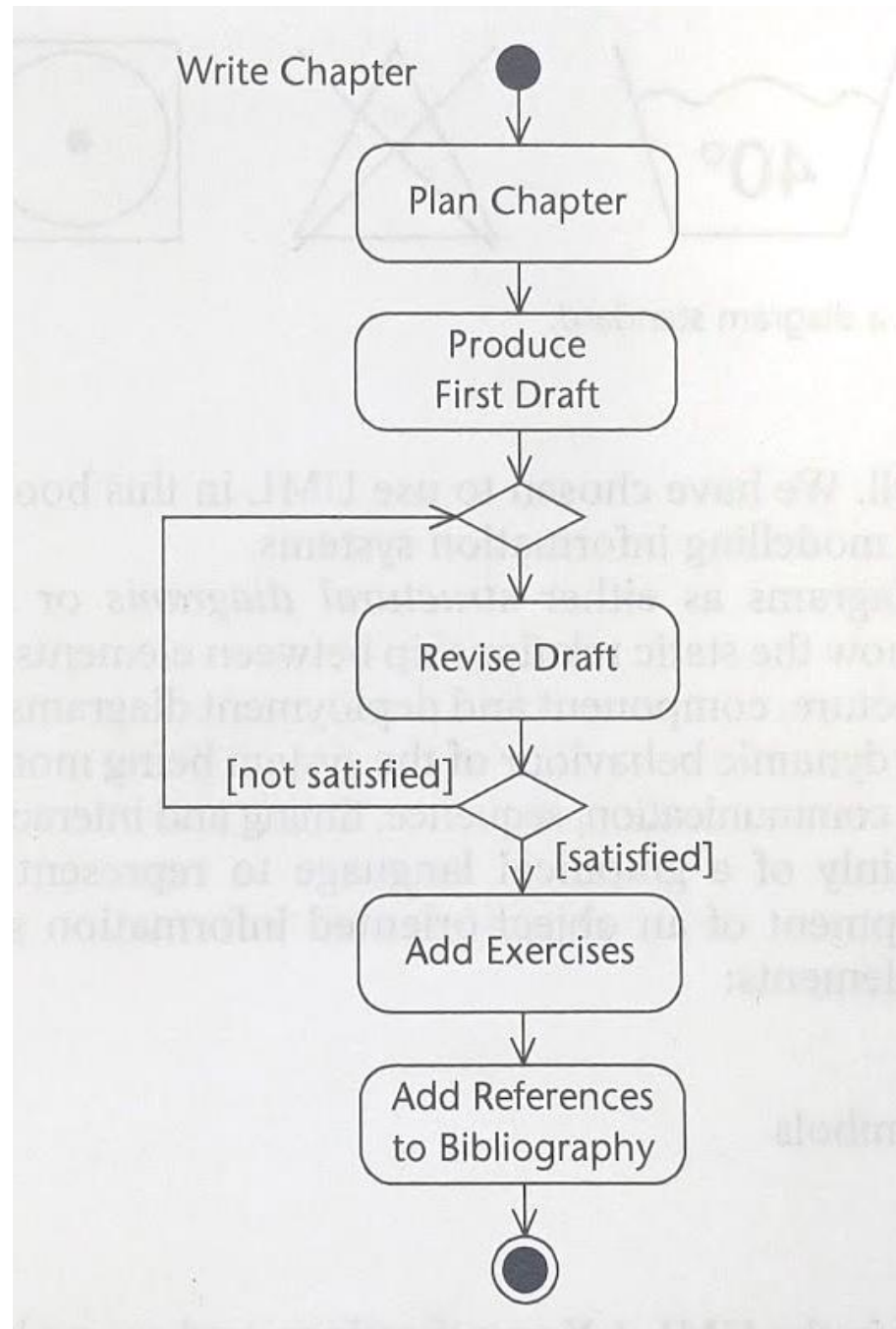
Pins

A **Pin** shows the type of item that is expected or created by an **action**.

They can be used to replace object flows and make a diagram a little less cluttered.



Example 1



Example 2

