MSc programme (induction week) – Department of Informatics

INTRODUCTION TO UML

Some of this material is based on

Bernd Bruegge and Allen H. Dutoit (2009) 'Object-Oriented Software Engineering: Using UML, Patterns, and Java', Pearson, 3rd edition.

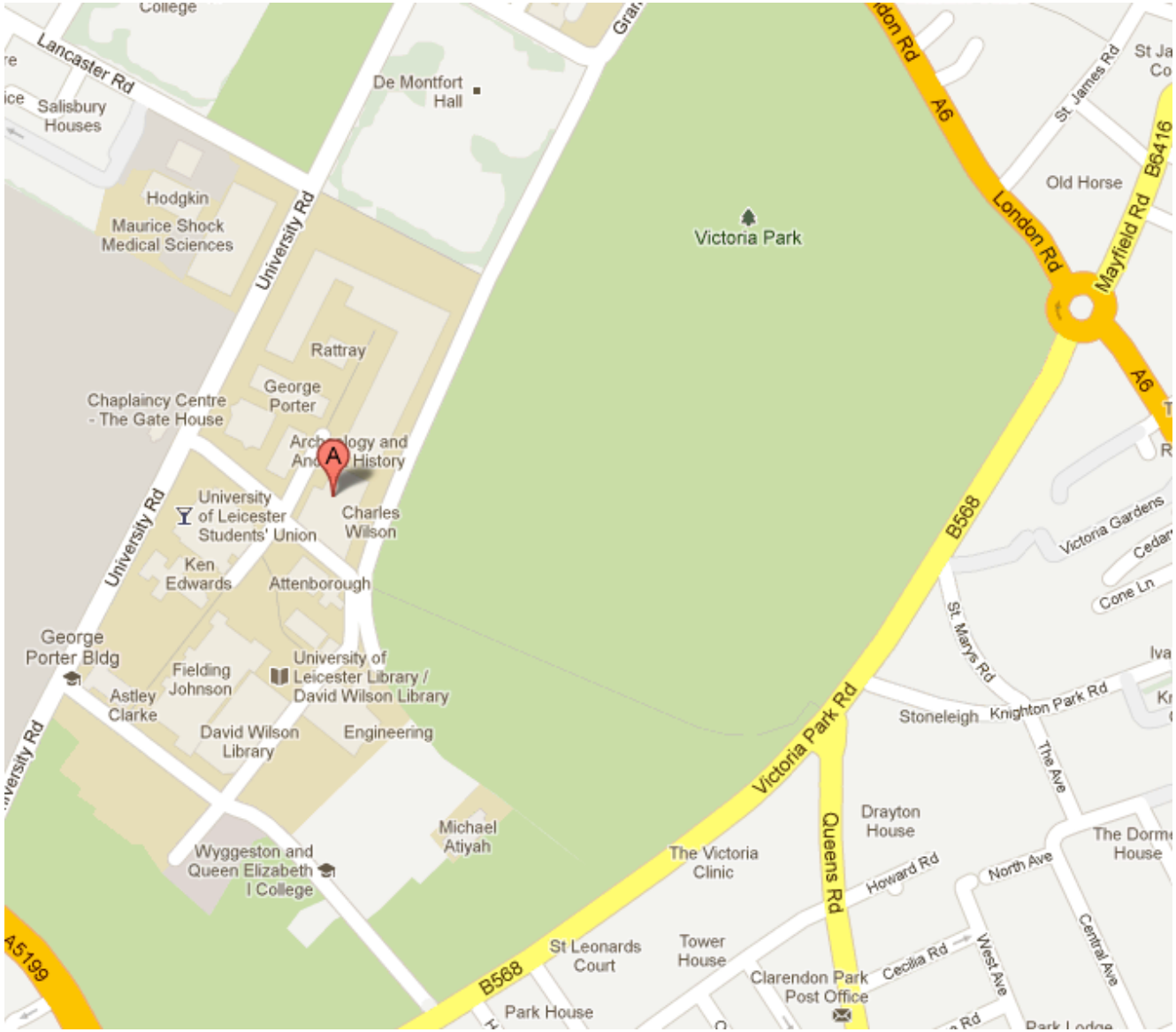
Overview: modelling with UML

- What is modelling?
- What is UML?
- Use case diagrams
- Class diagrams
- Sequence diagrams
- Activity diagrams

What is modelling?

- Modelling consists of building an abstraction of reality.
- Abstractions are simplifications because:
  - **They ignore irrelevant details and**
  - **They only represent the relevant details.**
- What is *relevant* or *irrelevant* depends on the purpose of the model.

Example: street map



Why model software?

- Software is getting increasingly more complex:
  - **Samples of complex software.**
  - **A single programmer cannot manage this amount of code in its entirety.**
- Code is not easily understandable by developers who did not write it.
- We need simpler representations for complex systems:
  - **Modelling is a means for dealing with complexity.**

What should be done first? Coding or Modelling?

- It all depends….
- Forward Engineering
  - **Creation of code from a model**
  - **Start with modelling**
  - **Greenfield projects**
- Reverse Engineering
  - **Creation of a model from existing code**
  - **Interface or reengineering projects**
- Roundtrip Engineering
  - **Move constantly between forward and reverse engineering**
  - **Reengineering projects**
  - **Useful when requirements, technology and schedule are changing frequently.**

What is UML? Unified Modelling Language

- Convergence of different notations used in object-oriented methods, mainly
  - **OMT (James Rumbaugh and colleagues), OOSE (Ivar Jacobson), Booch (Grady Booch)**
- They also developed the Rational Unified Process, which became the Unified Process in 1999

Origins

- OO programming languages
- OO analysis and design techniques
  - **business modelling**
  - **analysis of requirements**
  - **design of software systems**
- UML: industry standard that merges the best features of different notations

What UML is not

- UML is not a programming language per se
- UML is not a software modelling tool
- UML is not a method, methodology or software development process

Why UML?

- De facto standard for OO modelling
- Unified modelling language
- UML provides extension mechanisms

UML overview

- Use case diagrams
  - **Describe the functional behaviour of the system as seen by the user.**
- Class diagrams
  - **Describe the static structure of the system: objects, attributes, associations.**
- Sequence diagrams
  - **Describe the dynamic behaviour between objects of the system.**
- Activity diagrams
  - **Describe the dynamic behaviour of a system, in particular the workflow.**

USE CASE DIAGRAM

Arriva Bus Home

Artur

https://www.arrivabus.co.uk/?gclid=CKmw3M3asc8CFcYV0wod434DMw

ARRIVA
a DB company

Search

Choose a region Please select... Go

Where we operate | Travel help and accessibility | About us | Bus tickets | Working with Arriva | Contact Us

Find services and timetables

Service number/town/postcode

Use my location Search

or

Journey planner

From: Bus stop/road name

To: Bus stop/road name

Clear search Go

Now you can buy your tickets with PayPal

PayPal

Find out more

Travel tickets

Buy now

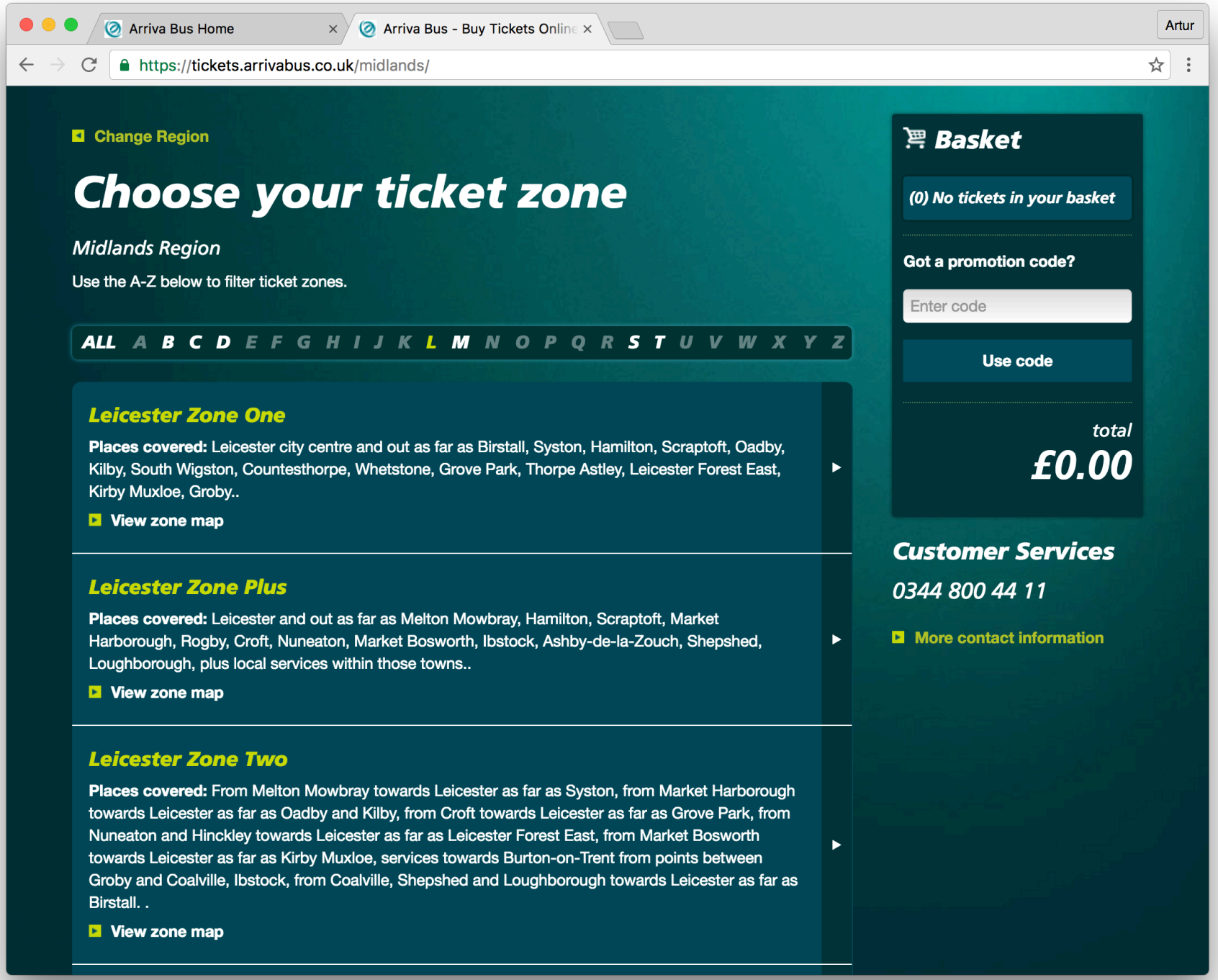
Sign up to our newsletter

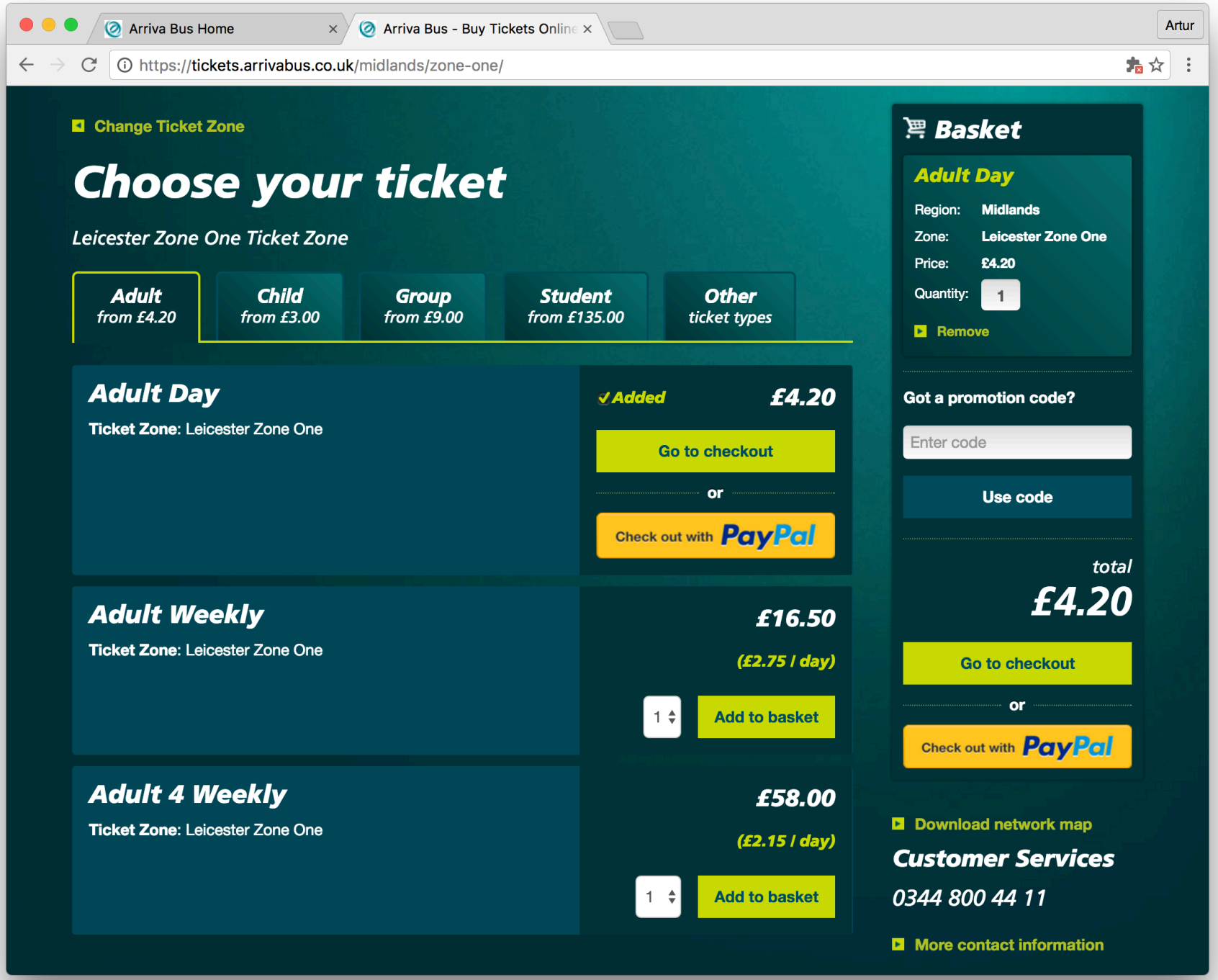
Sign up now

Welcome

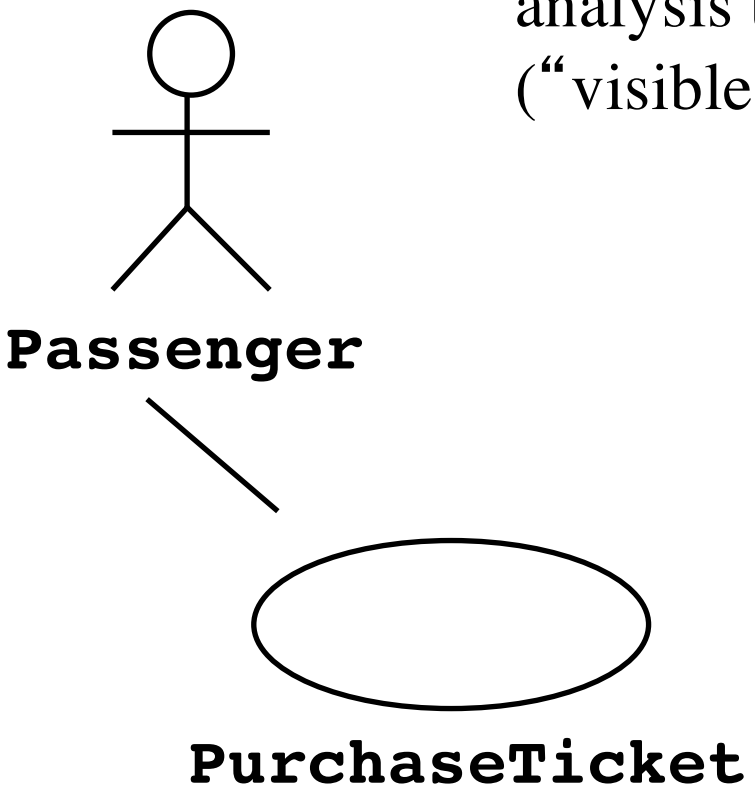
to Arriva buses

Use our website to plan your journey, buy tickets, look at timetables and catch up on our latest news. Browse employment opportunities with Arriva and find out how to get in touch with Arriva through a variety of channels.


Arriva Bus Home
Arriva Bus - Buy Tickets Online
Artur
https://tickets.arrivabus.co.uk/midlands/
Change Region
Choose your ticket zone
Midlands Region
Use the A-Z below to filter ticket zones.
ALL A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Leicester Zone One
Places covered: Leicester city centre and out as far as Birstall, Syston, Hamilton, Scraptoft, Oadby, Kilby, South Wigston, Countesthorpe, Whetstone, Grove Park, Thorpe Astley, Leicester Forest East, Kirby Muxloe, Groby..
View zone map
Leicester Zone Plus
Places covered: Leicester and out as far as Melton Mowbray, Hamilton, Scraptoft, Market Harborough, Rogby, Croft, Nuneaton, Market Bosworth, Ibstock, Ashby-de-la-Zouch, Shepshed, Loughborough, plus local services within those towns..
View zone map
Leicester Zone Two
Places covered: From Melton Mowbray towards Leicester as far as Syston, from Market Harborough towards Leicester as far as Oadby and Kilby, from Croft towards Leicester as far as Grove Park, from Nuneaton and Hinckley towards Leicester as far as Leicester Forest East, from Market Bosworth towards Leicester as far as Kirby Muxloe, services towards Burton-on-Trent from points between Groby and Coalville, Ibstock, from Coalville, Shepshed and Loughborough towards Leicester as far as Birstall. .
View zone map
Basket
(0) No tickets in your basket
Got a promotion code?
Enter code
Use code
total
£0.00
Customer Services
0344 800 44 11
More contact information


Arriva Bus Home
Arriva Bus - Buy Tickets Online
Artur
https://tickets.arrivabus.co.uk/midlands/zone-one/
Change Ticket Zone
Choose your ticket
Leicester Zone One Ticket Zone
Adult
from £4.20
Child
from £3.00
Group
from £9.00
Student
from £135.00
Other
ticket types
Adult Day
Ticket Zone: Leicester Zone One
✓Added
£4.20
Go to checkout
or
Check out with PayPal
Adult Weekly
Ticket Zone: Leicester Zone One
£16.50
(£2.75 / day)
1
Add to basket
Adult 4 Weekly
Ticket Zone: Leicester Zone One
£58.00
(£2.15 / day)
1
Add to basket
Basket
Adult Day
Region: Midlands
Zone: Leicester Zone One
Price: £4.20
Quantity: 1
Remove
Got a promotion code?
Enter code
Use code
total
£4.20
Go to checkout
or
Check out with PayPal
Download network map
Customer Services
0344 800 44 11
More contact information

UML Use Case Diagrams



Used during requirements elicitation and analysis to represent external behaviour ("visible from the outside of the system")

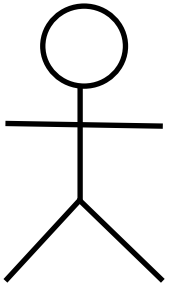
An ***Actor*** represents a role, that is, a type of user of the system

A ***use case*** represents a class of functionality provided by the system

Use case model:

The set of all use cases that completely describe the functionality of the system.

Actors



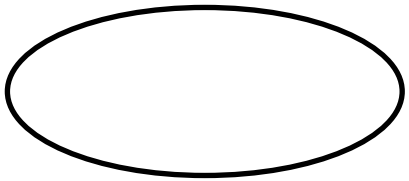
Passenger

- An actor is a model for an external entity which interacts (communicates) with the system:
  - **User**
  - **External system (Another system)**
  - **Physical environment (e.g. Weather)**
- An actor has a unique name and an optional description
- Examples:
  - **Passenger: A person in the train**
  - **GPS satellite: An external system that provides the system with GPS coordinates.**

Optional Description

Name

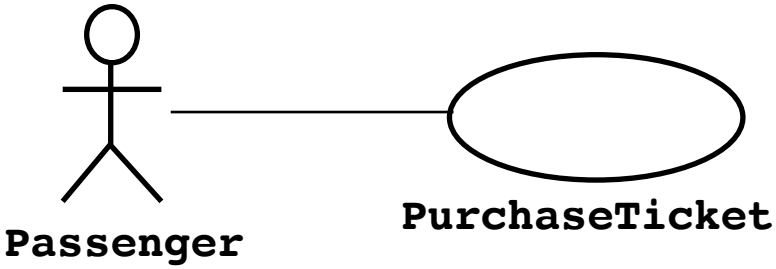
Use Case



`PurchaseTicket`

- A use case represents a class of functionality provided by the system
- Use cases can be described textually, with a focus on the event flow between actor and system
- The textual use case description consists of 6 parts:
  1. **Unique name**
  2. **Participating actors**
  3. **Entry conditions**
  4. **Exit conditions**
  5. **Flow of events**
  6. **Special requirements.**

Textual Use Case Description Example



1. Name: `Purchase ticket`

2. Participating actor: `Passenger`

3. Entry condition:

- `Passenger` stands in front of ticket distributor
- `Passenger` has sufficient money to purchase ticket

4. Exit condition:

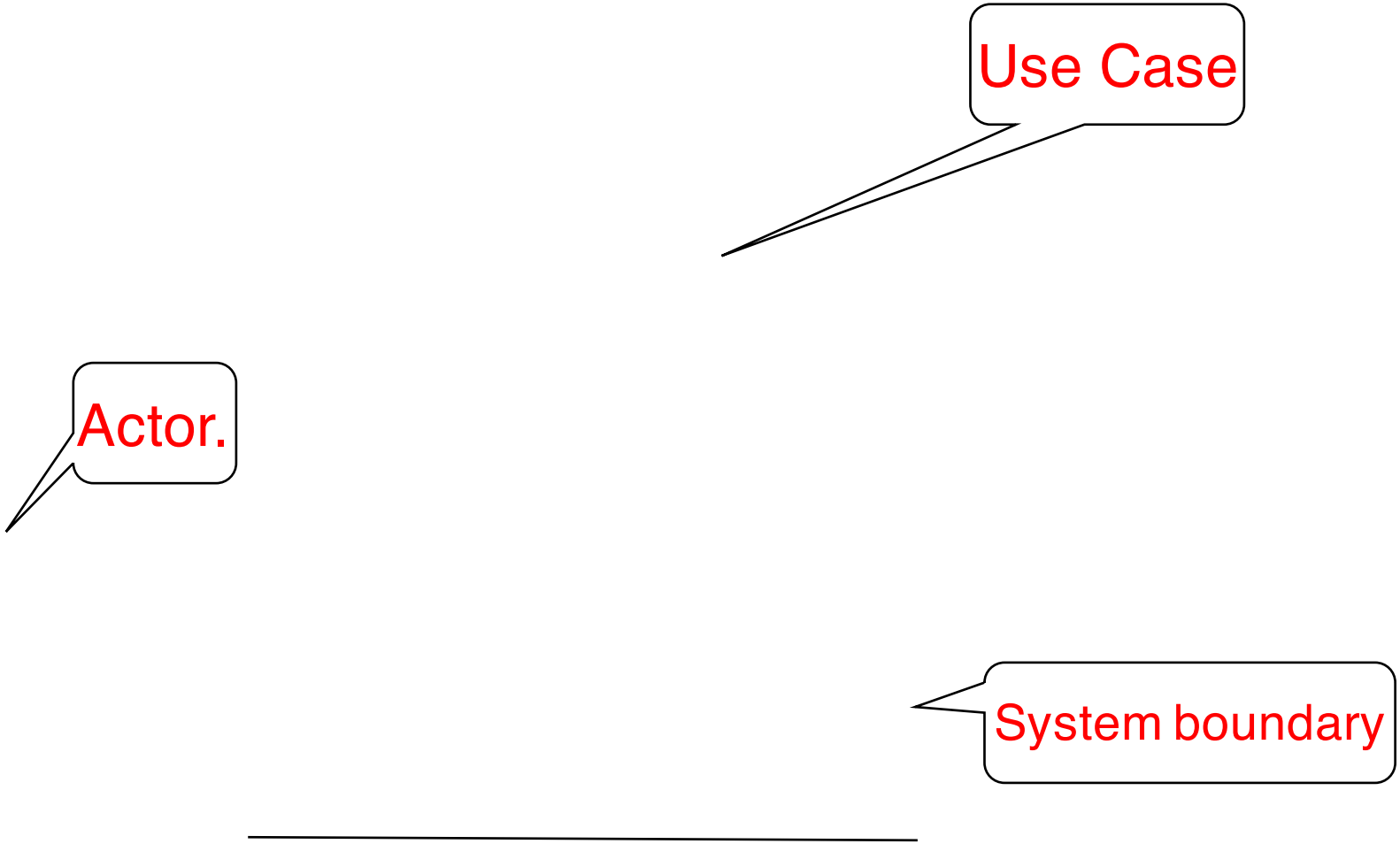
- `Passenger` has ticket

5. Flow of events:

1. **`Passenger` selects the number of zones to be traveled**
2. **Ticket Distributor displays the amount due**
3. **`Passenger` inserts money, at least the amount due**
4. **Ticket Distributor returns change**
5. **Ticket Distributor issues ticket**

6. Special requirements: *None.*

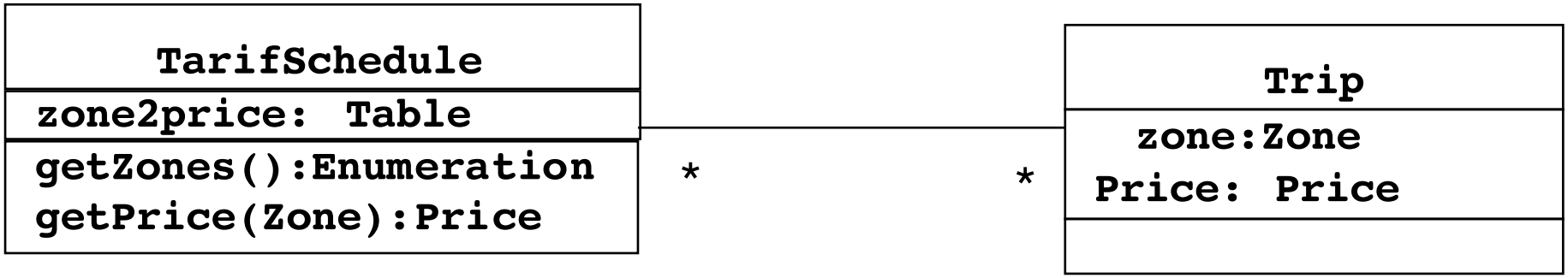
Use Case Models should be packaged



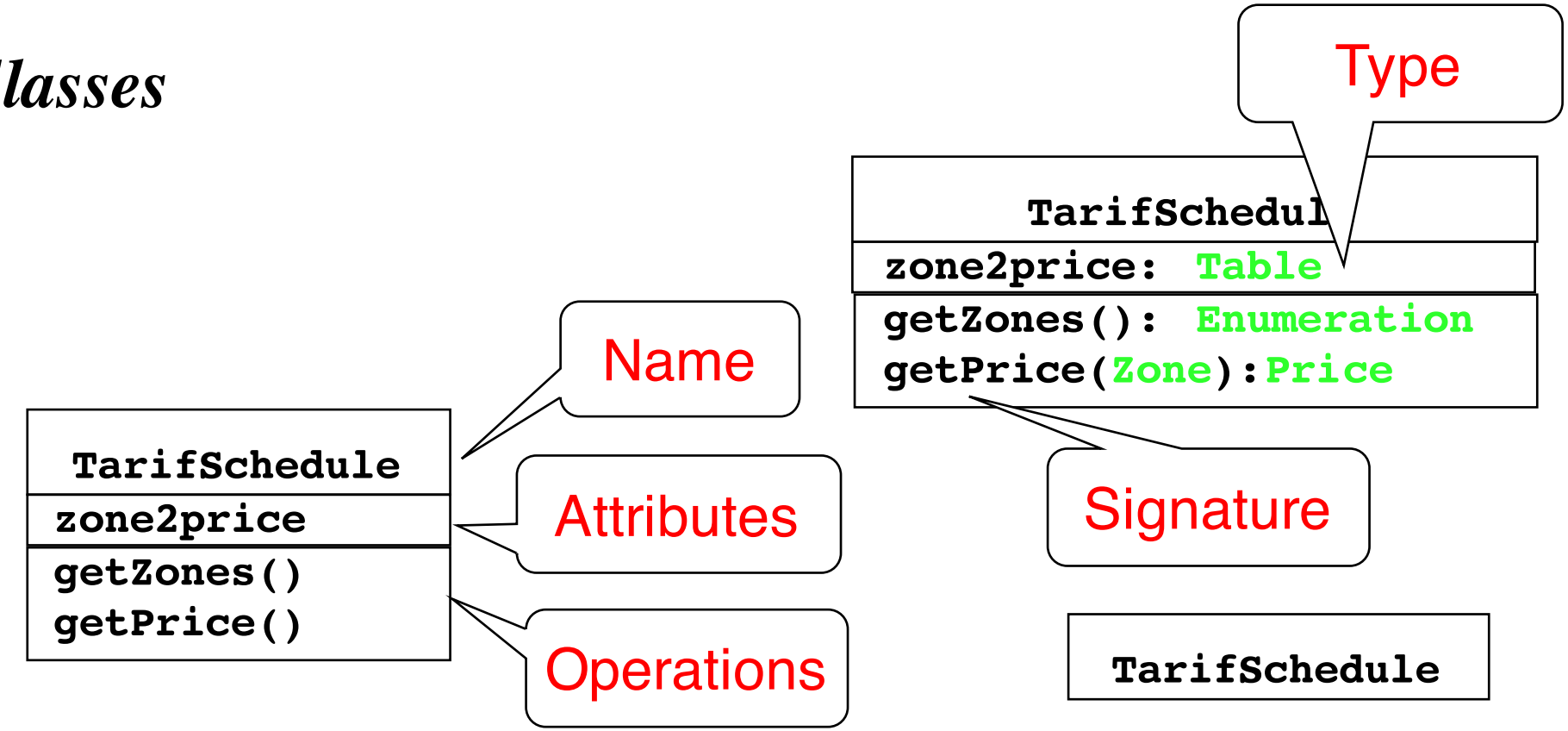
CLASS DIAGRAM

Class Diagrams

- Class diagrams represent the structure of the system
- Used
  - **during requirements analysis to model application domain concepts**
  - **during system design to model subsystems**
  - **during object design to specify the detailed behaviour and attributes of classes.**



Classes

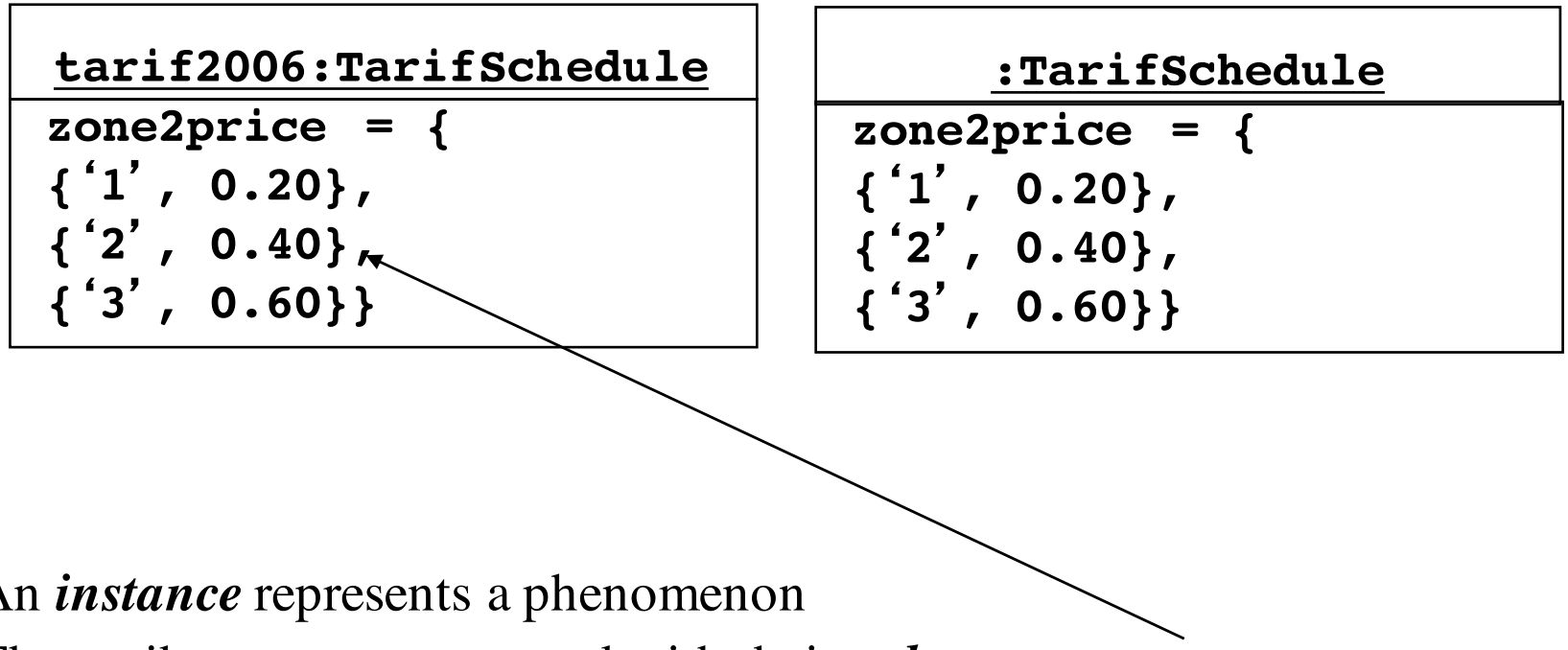


- A ***class*** represents a concept
- A class encapsulates state ***(attributes)*** and behaviour ***(operations)***

  Each attribute has a ***type***
  Each operation has a ***signature***

The class name is the only mandatory information

Instances

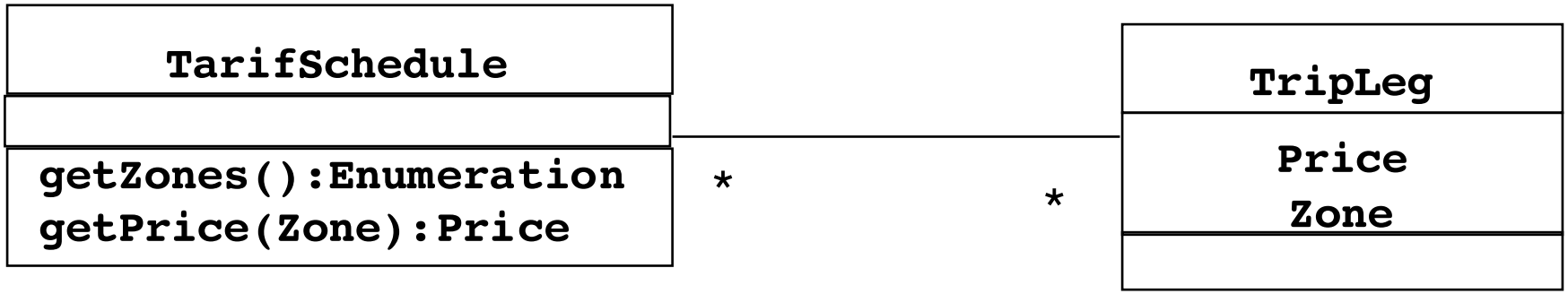


- An ***instance*** represents a phenomenon
- The attributes are represented with their ***values***
- The name of an instance is underlined
- The name can contain only the class name of the instance (anonymous instance)

Actor vs Class vs Object

- **Actor**
  - **An entity outside the system to be modelled, interacting with the system (“Passenger”)**
- **Class**
  - **An abstraction modelling an entity in the application or solution domain**
  - **The class is part of the system model (“User”, “Ticket distributor”, “Server”)**
- **Object**
  - **A specific instance of a class (“Joe, the passenger who is purchasing a ticket from the ticket distributor”).**

Associations

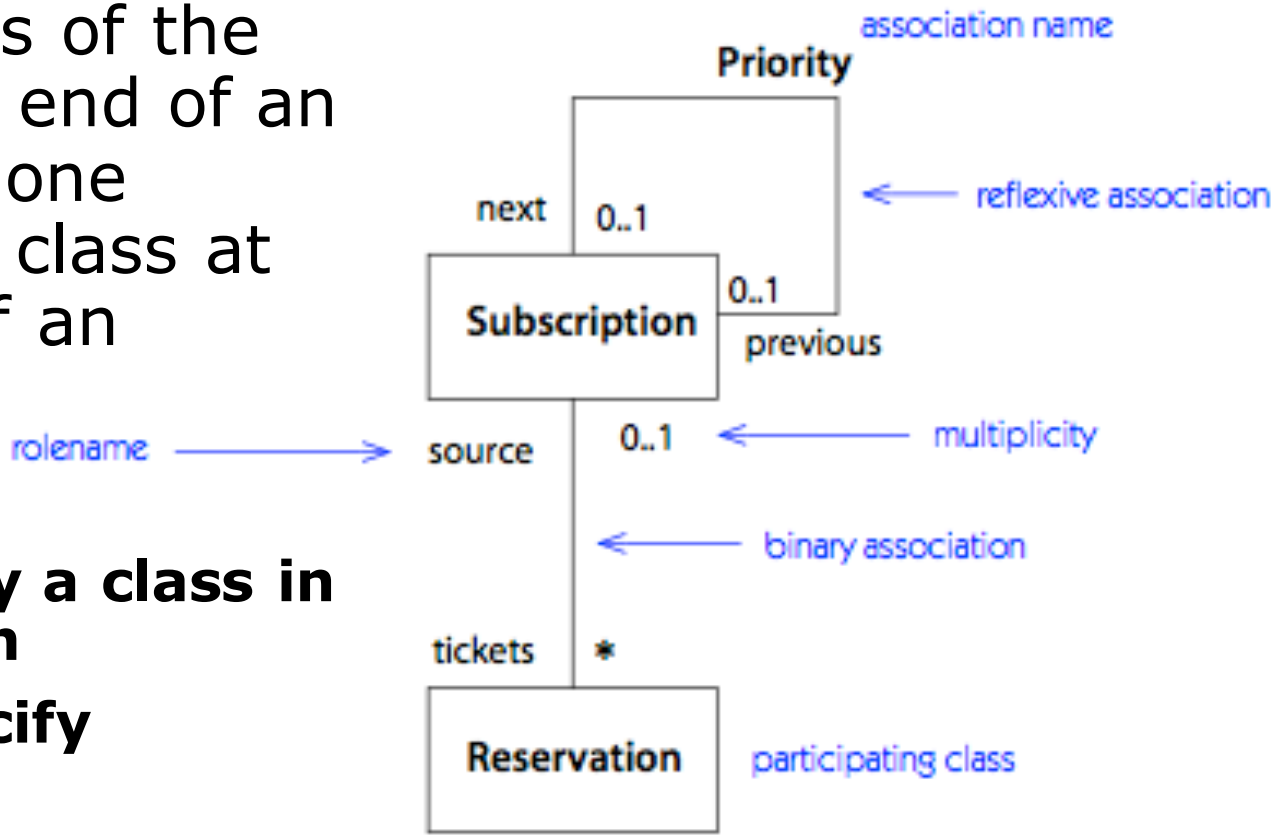


Associations denote collaborations between classes by means of message exchange.

The multiplicity of an association end denotes how many objects the instance of a class can legitimately reference.

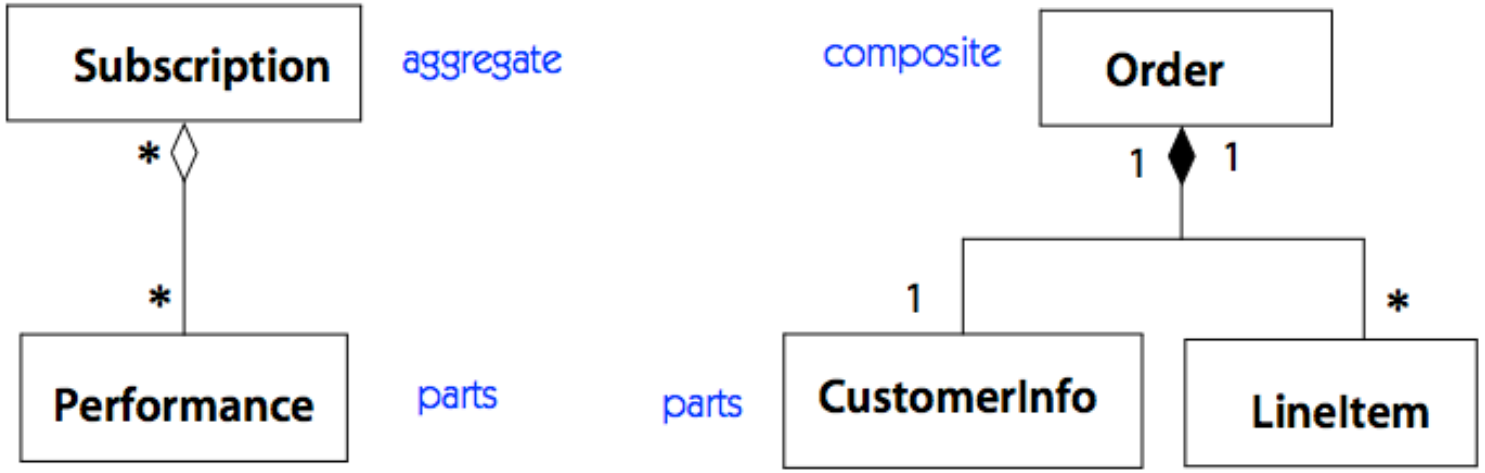
Association properties

- Name
- Multiplicity: number of object instances of the class at the far end of an association for one instance of the class at the near end of an association
- Role names
  - **role played by a class in an association**
  - **useful to specify methods**

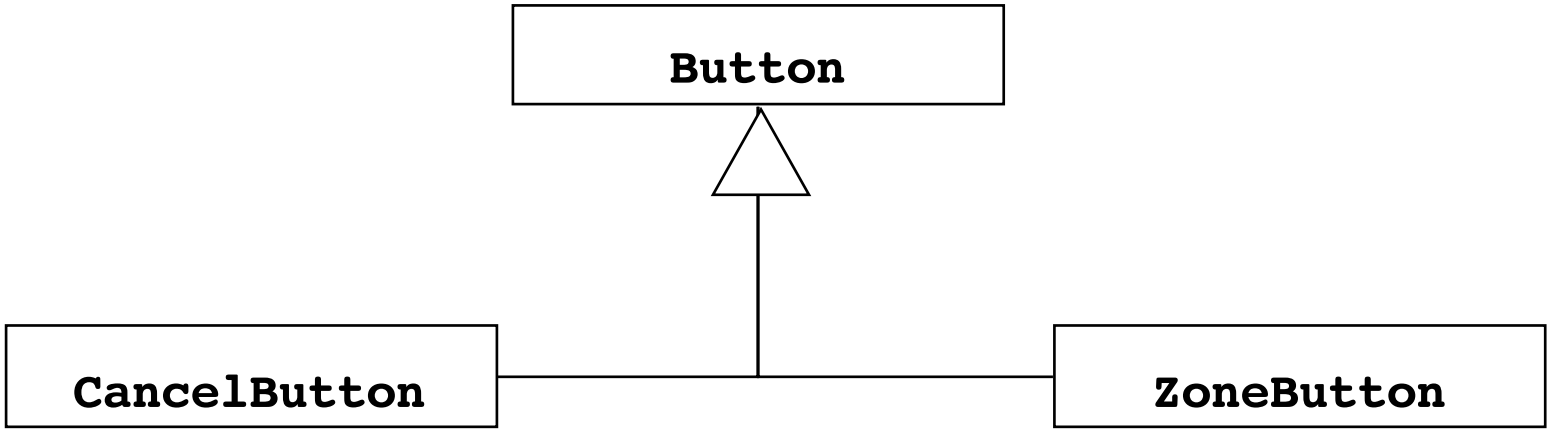


Aggregation

- An *aggregation* is a special case of association denoting that one class may consist of, or include, instances of another class.
- A solid diamond denotes *composition*: the *lifetime of the component instances* is controlled by the aggregate.



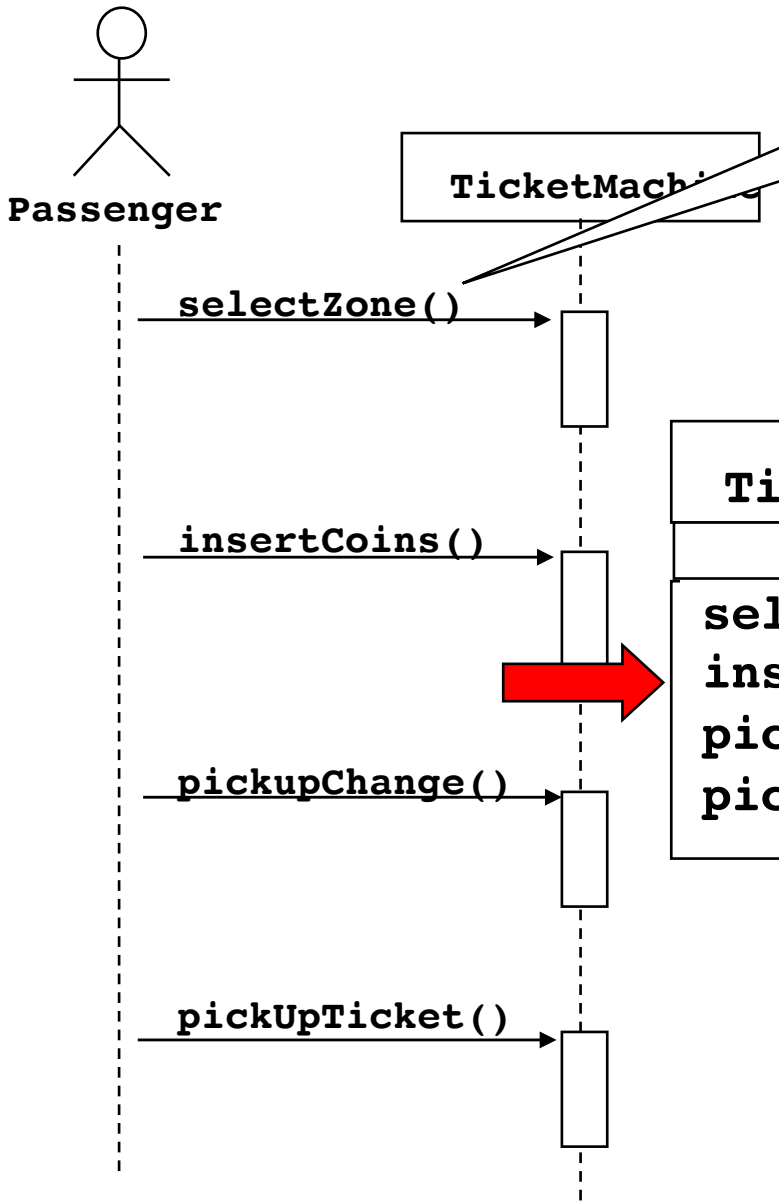
Inheritance



- *Inheritance* is another special case of an association denoting a "kind-of" hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The **children classes** inherit the attributes and operations of the **parent class.**

SEQUENCE DIAGRAM

Sequence Diagrams



Focus on control flow

- Used during analysis
  - **To refine use case descriptions**
  - **to find additional objects ("participating objects")**
- Used during system design
  - **ine su ystem interfaces**

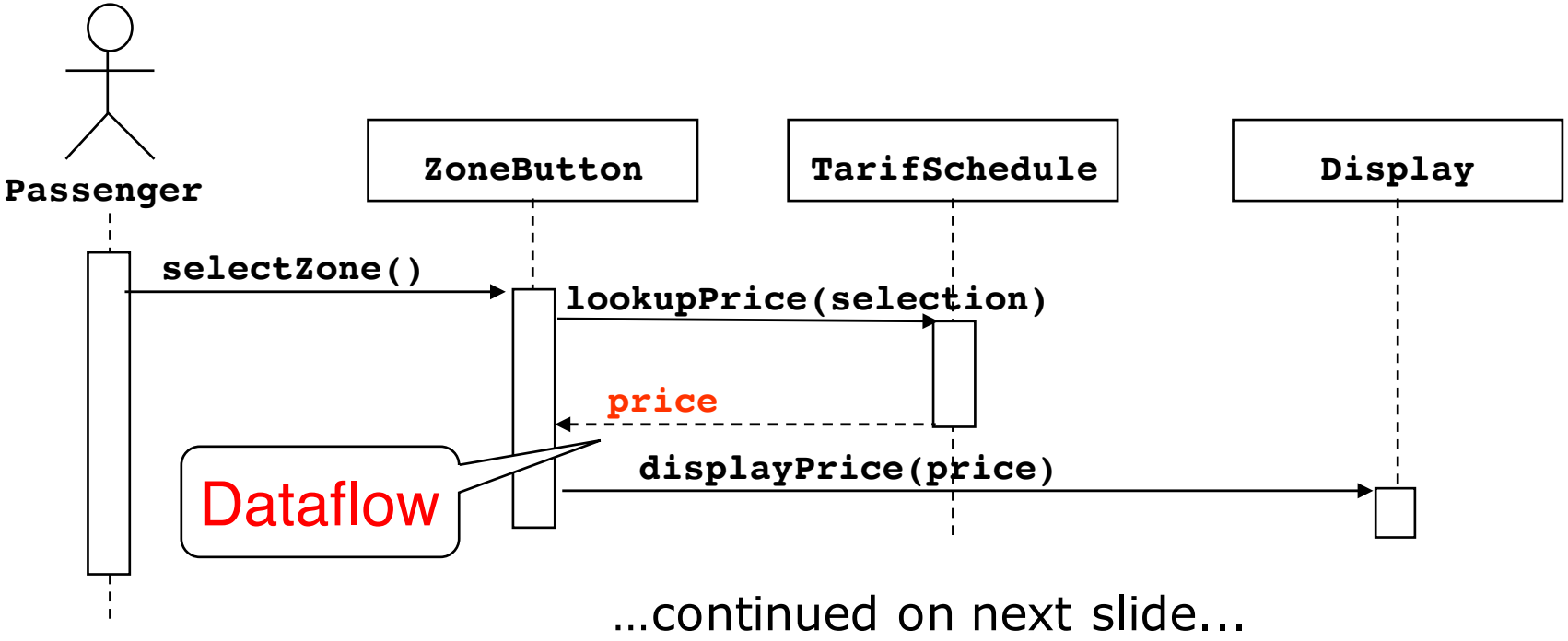
TicketMachine

selectZone()
insertCoins()
pickupChange()
pickUpTicket()

Messages -> Operations on participating Object

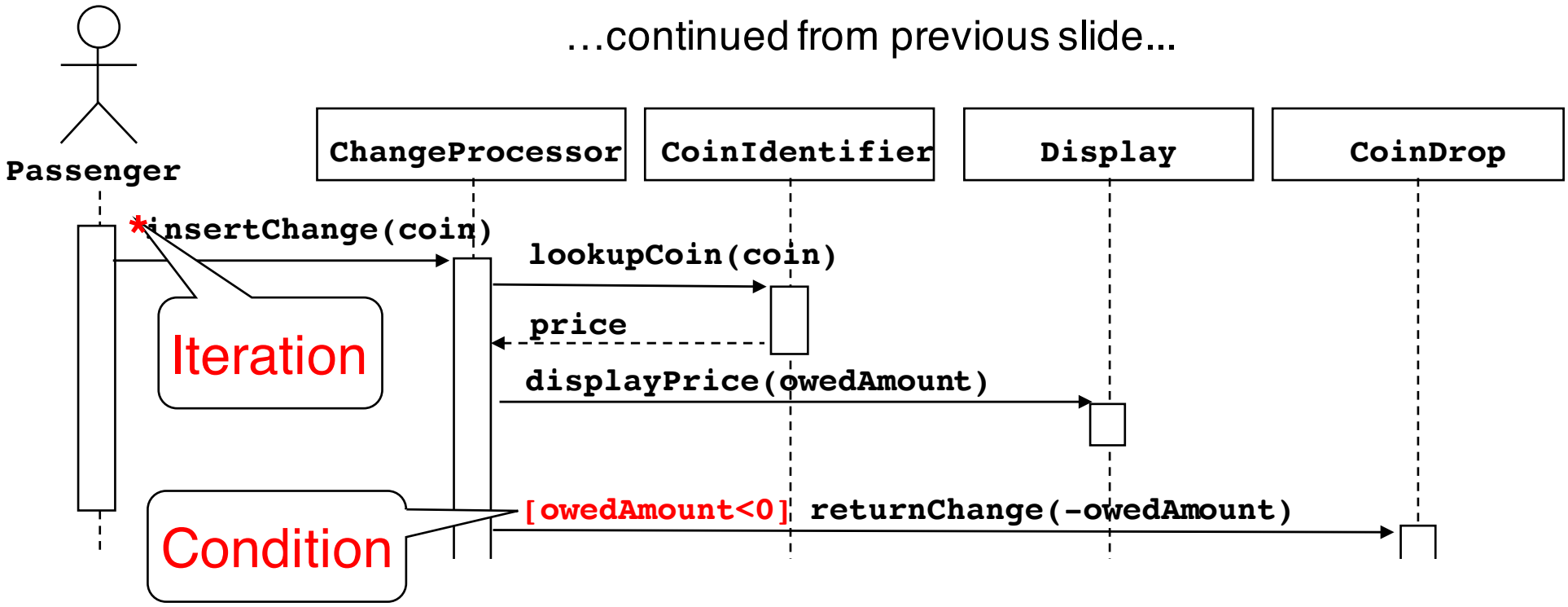
- ***s*** are repr
- s. ***Actors*** b
- are repres
- ***s*** are represented by arrows
- ***Activations*** are represented by narrow rectangles.

Sequence Diagrams can also model the Flow of Data



- The source of an arrow indicates the activation which sent the message
- Horizontal dashed arrows indicate data flow, for example return results from a message

Sequence Diagrams: Iteration & Condition



- Iteration is denoted by a * preceding the message name
- Condition is denoted by boolean expression in [ ] before the message name

Sequence Diagram Properties

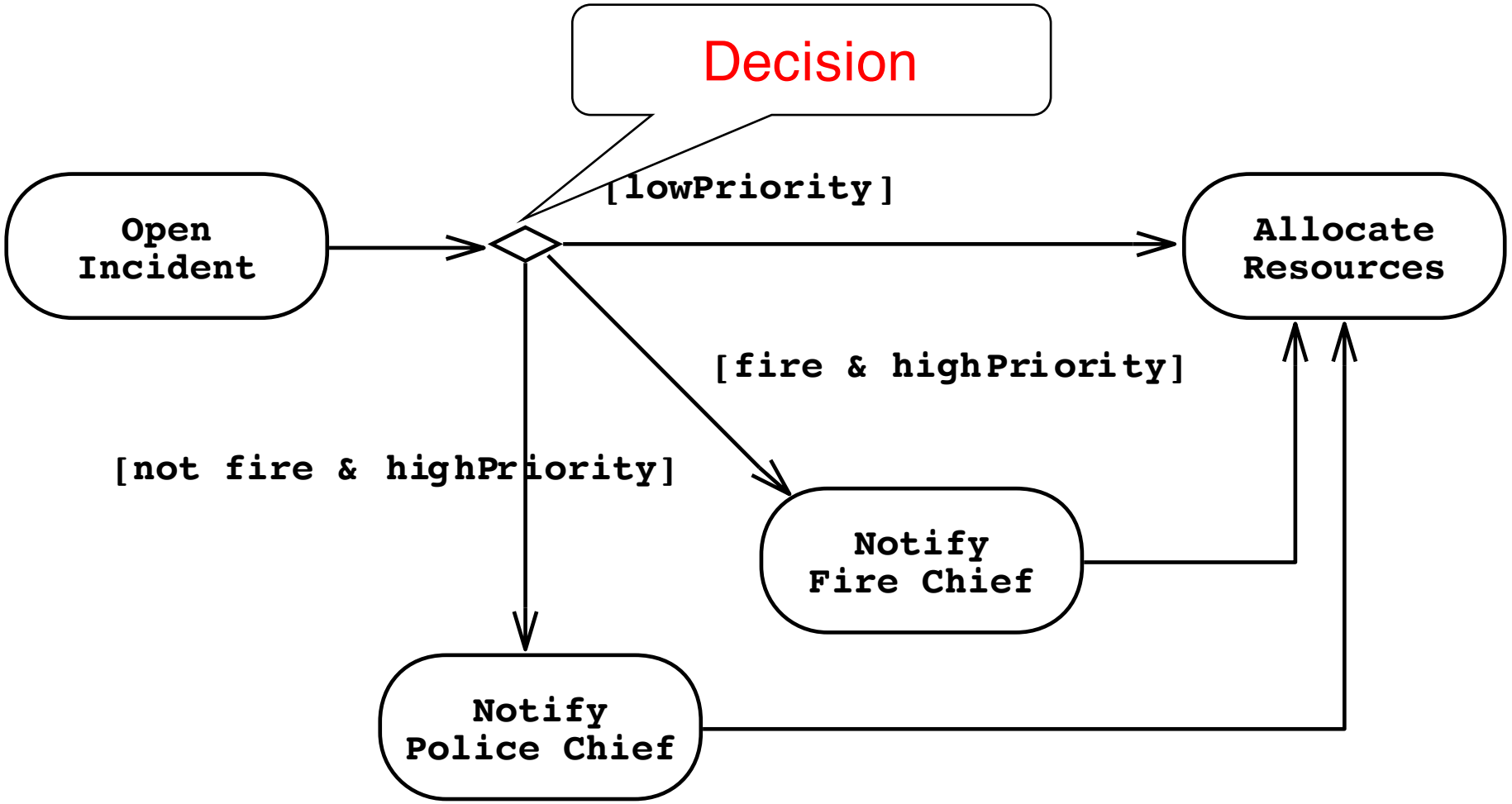
- UML sequence diagram represent *behaviour in terms of interactions*
- Useful to identify or find missing objects
- Time consuming to build, but worth the investment
- Complement class diagrams (which represent structure).

ACTIVITY DIAGRAM

Activity Diagrams

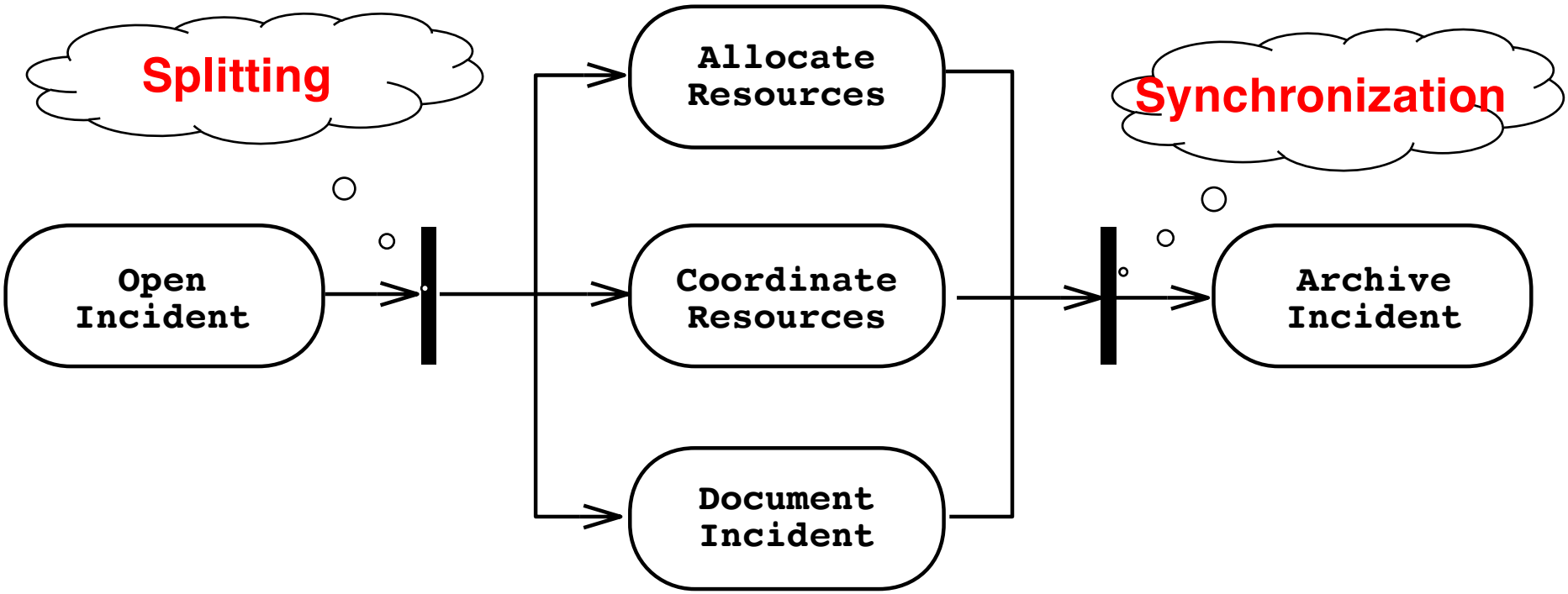
- An activity diagram is a special case of a state chart diagram
- The states are activities (“functions”)
- An activity diagram is useful to depict the workflow in a system

Activity Diagrams allow to model Decisions



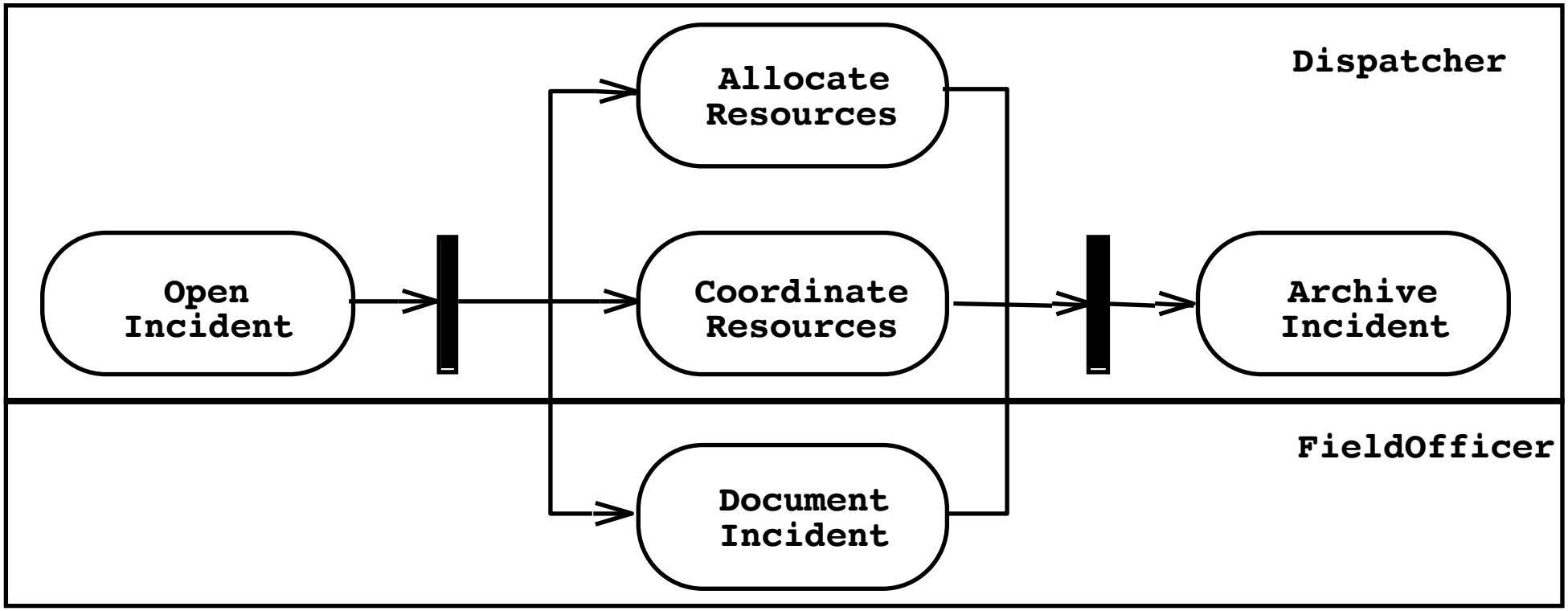
Activity Diagrams can model Concurrency

- Synchronization of multiple activities
- Splitting the flow of control into multiple threads



Activity Diagrams: Grouping of Activities

- Activities may be grouped into swimlanes to denote the object or subsystem that implements the activities.



UML Summary

- UML provides a wide variety of notations for representing many aspects of software development
  - **Powerful, but complex**
- UML is not a programming language
  - **Can be misused to generate unreadable models**
  - **Can be misunderstood when using too many exotic features**
- We concentrated on a few notations:
  - **Functional model: Use case diagram**
  - **Object model: class diagram**
  - **Dynamic model: sequence diagrams and activity diagrams**

Additional References

- Martin Fowler
  - **UML Distilled: A Brief Guide to the Standard Object Modelling Language, 3rd ed., Addison-Wesley, 2003**
- Grady Booch, James Rumbaugh, Ivar Jacobson
  - **The Unified Modelling Language User Guide, Addison Wesley, 2nd edition, 2005**
- Open Source UML tools (https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools)
  - **Eclipse Papyrus (built atop the Eclipse Modeling Framework)**
    - **https://www.eclipse.org/papyrus/**
  - **PlantUML (textual UML using DSLs)**
    - **http://plantuml.com/**