Object-Oriented Development Process

How to produce Class Diagram!!!!

Stages in building a class diagram

Two approaches:

- Use case realisation – identify classes needed to perform functionality identified in use cases. Proceed use case by use case.
- A domain model – model classes for the whole problem domain

Stages in building a class diagram

- identify the objects and derive classes from them;
- identify attributes of classes;
- identify relationships between the classes;
- separate responsibilities into operations and attributes;
- iterate and refine the model.

Object identification using nouns

1. Find a complete description of system requirements
2. Underline all the nouns and noun phrases (person, place or thing)
3. This gives a list of candidate objects
4. Reject objects that will not make suitable classes

Description of problem:

- R1 keep a complete list of all bikes and their details including bike number, type, size, make, model, daily charge rate, deposit; (this is already on the Wheels system);
- R2 keep a record of all customers and their past hire transactions;
- R3 work out automatically how much it will cost to hire a given bike for a given number of days;
- R4 record the details of a hire transaction including the start date, estimated duration, customer and bike, in such a way that it is easy to find the relevant transaction details when a bike is returned;
- R5 keep track of how many bikes a customer is hiring so that the customer gets one unified receipt not a separate one for each bike;
- R6 cope with a customer who hires more than one bike, each for different amounts of time;
- R7 work out automatically, on the return of a bike, how long it was hired for, how many days were originally paid for, how much extra is due;
- R8 record the total amount due and how much has been paid;
- R9 print a receipt for each customer;
- R10 keep track of the state of each bike, e.g. whether it is in stock, hired out or being repaired;
- R11 provide the means to record extra details about specialist bikes.

Underline the nouns and noun phrases

- R1 keep a complete list of all bikes and their details including bike number, type, size, make, model, daily charge rate, deposit; (this is already on the Wheels system);
- R2 keep a record of all customers and their past hire transactions;
- R3 work out automatically how much it will cost to hire a given bike for a given number of days;
- R4 record the details of a hire transaction including the start date, estimated duration, customer and bike, in such a way that it is easy to find the relevant transaction details when a bike is returned;
- R5 keep track of how many bikes a customer is hiring so that the customer gets one unified receipt not a separate one for each bike;
- R6 cope with a customer who hires more than one bike, each for different amounts of time;
- R7 work out automatically, on the return of a bike, how long it was hired for, how many days were originally paid for, how much extra is due;
- R8 record the total amount due and how much has been paid;
- R9 print a receipt for each customer;
- R10 keep track of the state of each bike, e.g. whether it is in stock, hired out or being repaired;
- R11 provide the means to record extra details about specialist bikes

List of *nouns*:

- list of bikes
- details of bikes: bike number, type, size, make, model, daily charge rate, deposit
- Wheels system
- record of customers
- past hire transactions
- bike
- number of days
- details of a hire transaction: start date, estimated duration
- customer
- receipt
- different amounts of time
- return of a bike
- total amount due
- state of each bike
- extra details about specialist bikes

Remove *attribute nouns*:

list of bikes
details of bikes: *bike number, type, size, make, model, daily charge rate, deposit*
Wheels system
record of customers
past hire transactions
bike
number of days
details of a hire transaction: *start date, estimated duration*
customer
receipt
different amounts of time
return of a bike
total amount due
state of each bike
extra details about specialist bikes

… information about a class, not a class itself

Remove *redundancy/duplicates*

list of bikes
Wheels system
record of customers
past hire transactions
bike
hire transaction
customer
receipt
return of a bike
specialist bike

list of bikes
Wheels system
record of customers
bike
hire transaction
customer
receipt
return of a bike
specialist bike

… different names for the same thing

Remove *vague* nouns:

list of bikes	*list of bikes*
Wheels system	Wheels system
record of customers	*record of customers*
bike	bike
hire transaction	hire transaction
customer	customer
receipt	*receipt*
return of a bike	specialist bike
specialist bike	

... words without precise meaning

Remove nouns too tied up with physical inputs and outputs:

list of bikes
Wheels system
record of customers
bike
hire transaction
customer
receipt
specialist bike

... data inputs or system products

Remove *association* nouns:

- Is hires an association or a class?
- If there is data associated, probably a class
- Hire: start date, number of days, so
- Keep Hire as a class

Remove nouns that represent the whole system:

Wheels system
bike
hire
customer
specialist bike

... we want to divide the system into separate objects

Remove nouns outside scope of system:

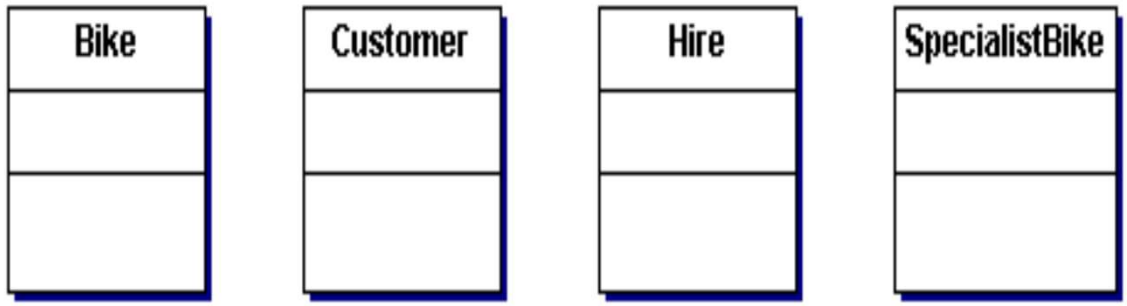
The system will not cover:

- payroll
- personnel
- general accounting

... not part of the intended system

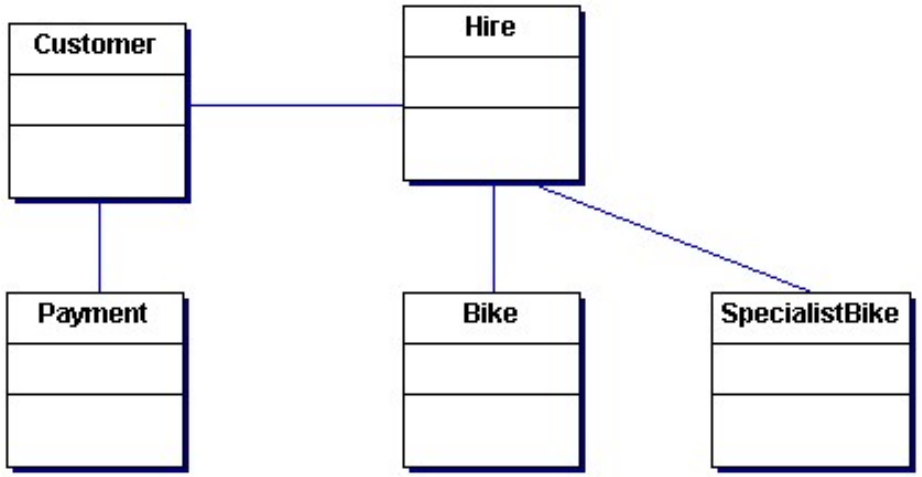
Identified objects, derive classes

bike
customer
hire
specialist bike



... these nouns are left as potential classes

Add missing classes



- Add Payment class
- Not all classes appear as nouns in the problem description
- Apply common sense

Identify attributes of classes

- Many nouns will appear in the text being analysed e.g. bike number, available, type etc are attributes of bike.

Avoid

- Attributes not relevant to current system – e.g. Customer passport number
- Derivable attributes – e.g. cost of hire (dialyHireRate*numberOfDays)
- Implementation attributes - pointers

Identify relationships between classes

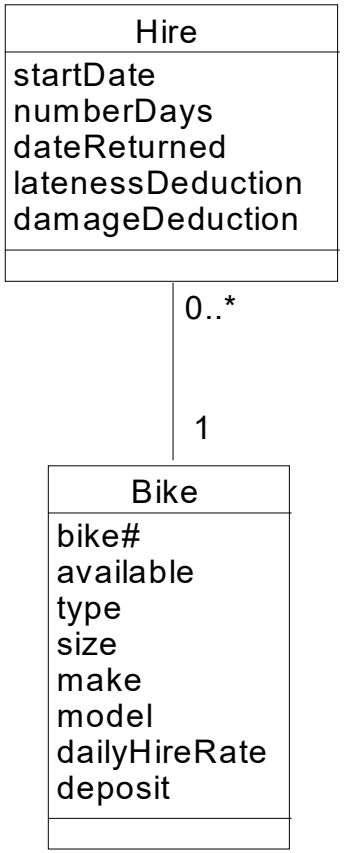
- During analysis we have not yet got an exact notion of how objects will need to communicate with each other
- The relationships that we include at this stage model real-life relationships that we think might be useful
- We will not have an exact idea of the navigable paths we need to build in until after looking at the interaction diagram.

Associations and Multiplicity

The Hire class holds data about the hiring of a bike. It needs to communicate with the Bike class to work out the cost of a hire (numberDays*dailyHireRate)

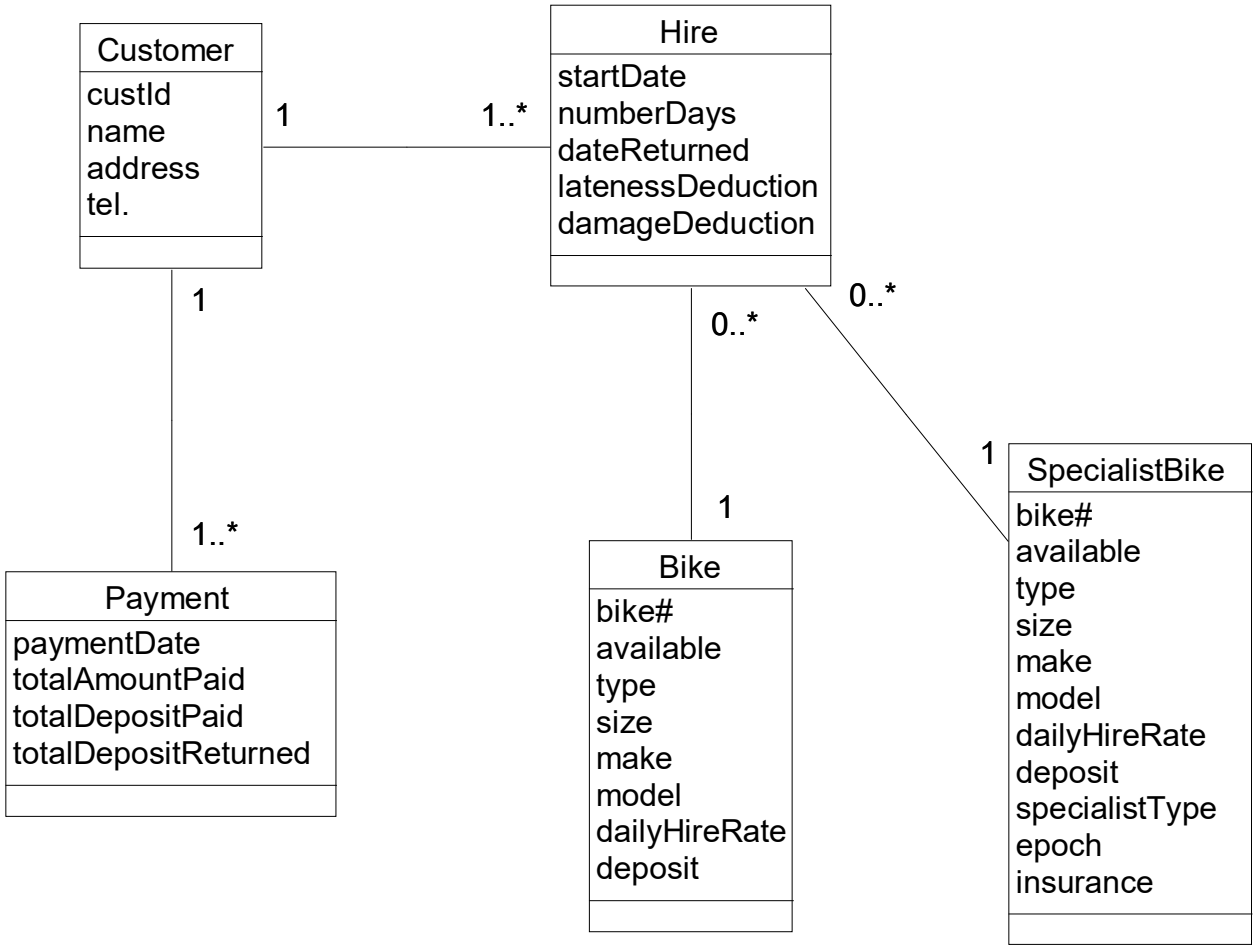
To perform this calculation there must be a relationship between Hire and Bike

Associations and Multiplicity

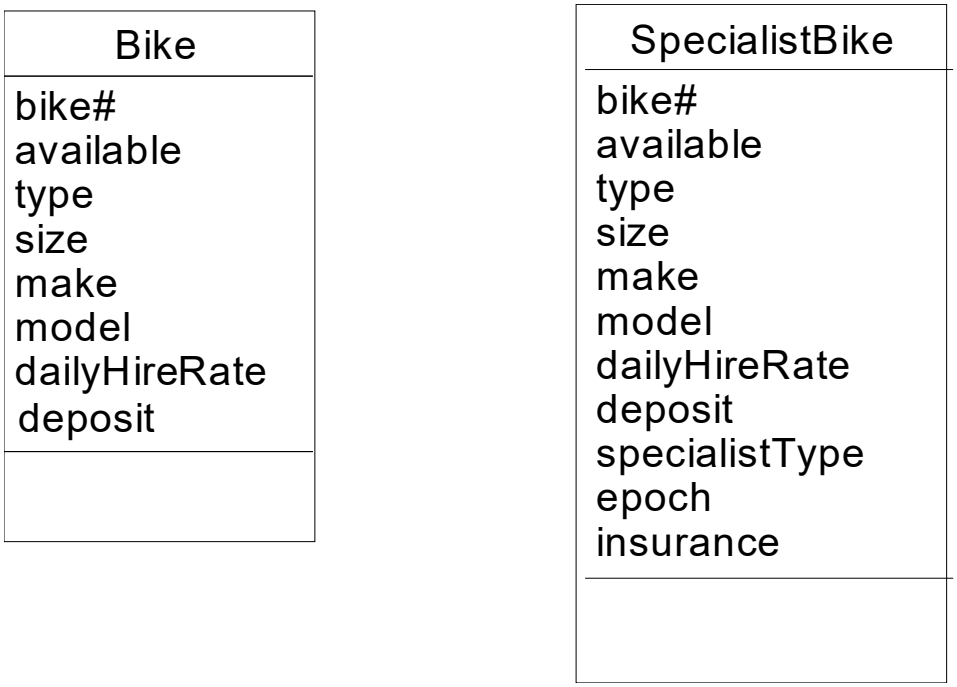


Relationship between Hire and Bike showing that a :Hire is for only one :Bike but a :Bike can be hired 0, 1 or many times

Wheels class diagram with initial associations

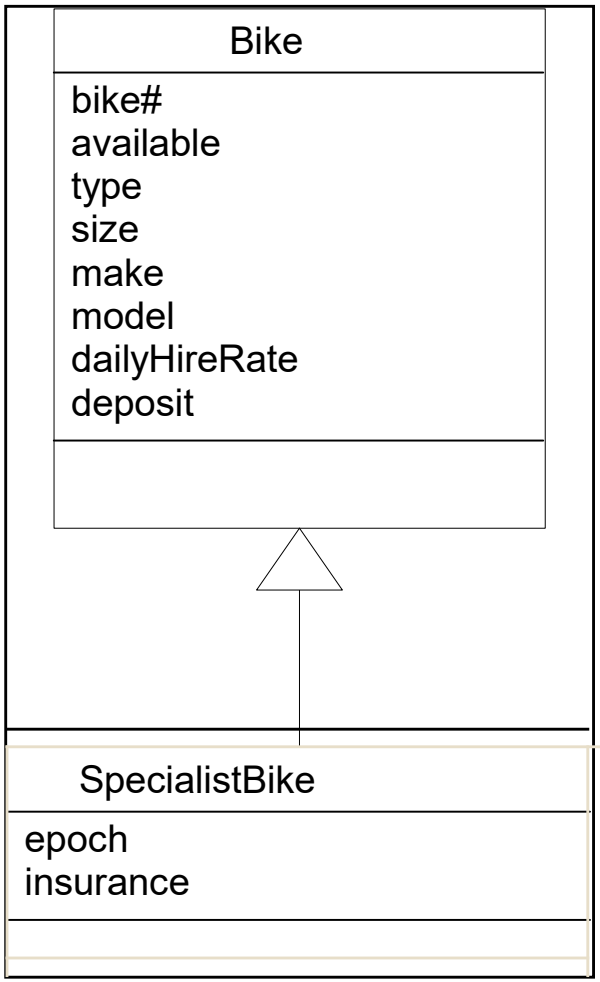


Generalization and inheritance

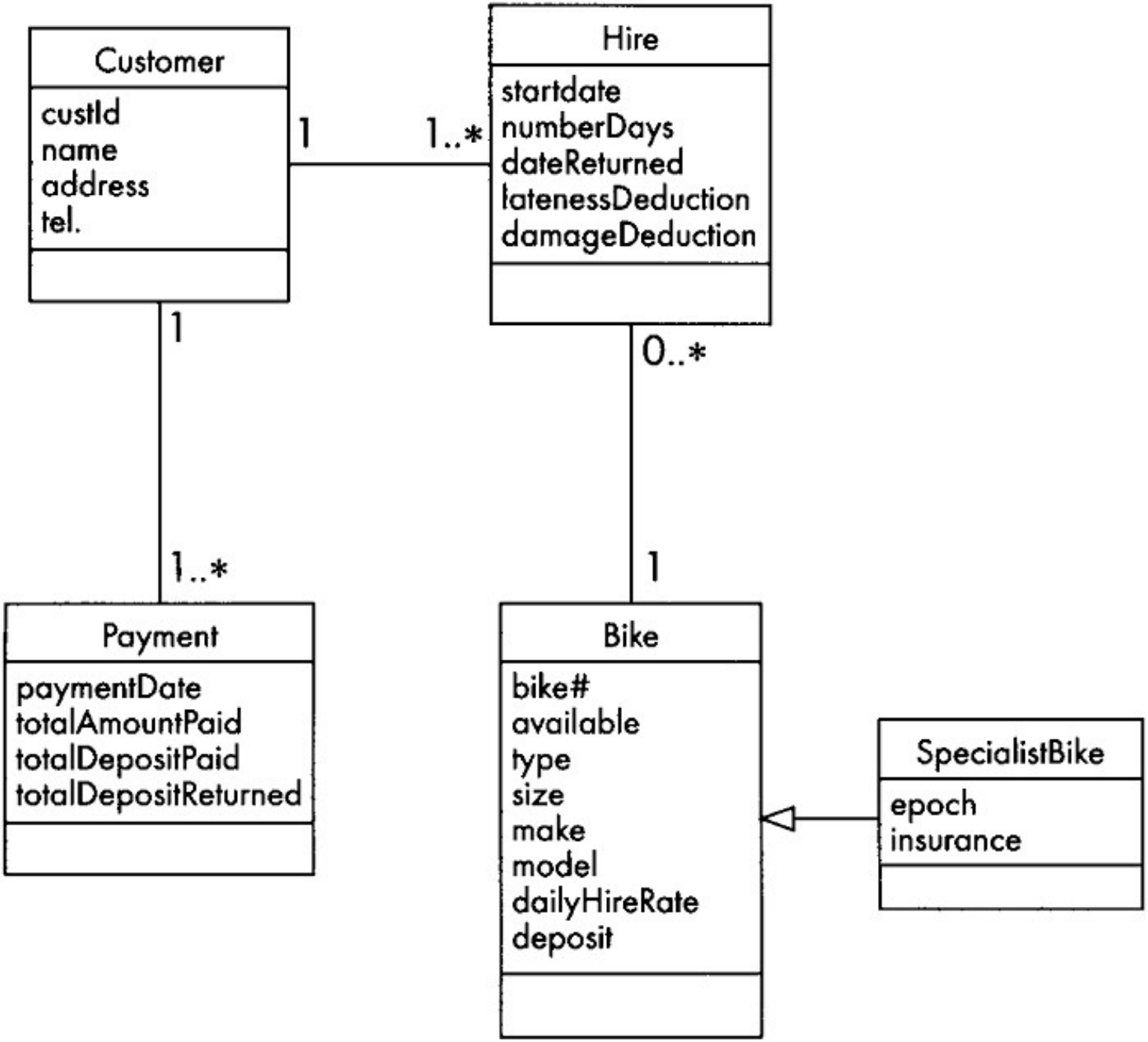


- Many shared attributes
- type same as specialistType ,

Inheritance



- Shared attributes inherited by SpecialistBike
- Distinguishing attributes (epoch and insurance) are unique to SpecialistBike


Customer
custId
name
address
tel.
Hire
startdate
numberDays
dateReturned
latenessDeduction
damageDeduction
1
1..*
1
0..*
1..*
1
Payment
paymentDate
totalAmountPaid
totalDepositPaid
totalDepositReturned
Bike
bike#
available
type
size
make
model
dailyHireRate
deposit
SpecialistBike
epoch
insurance