Lecture 6

- Covers
  - Algorithms (problem solving) using functions (methods)
  - Object-oriented analysis and design

- Algorithm development using functions

Algorithm development

- We cover algorithm development in 3 parts
  - Problem-solving procedure
  - Three control structures
  - Using functions (methods, operations)

What is a function?

- Algorithm for crossing the road

DO
Look left
Look right
Look left
WHILE road is busy
Walk across

Function Look left

```
FUNCTION Look left
    Turn head left
    Observe
    IF cars are coming THEN
        road is busy
    ENDIF
ENDFUNCTION
```

What is a function?

- A segment of code extracted separately
- Given a name
- May be used (called) more than once
- May be used from different parts of a program

Functions in Java

- In Java all pieces of code must be inside a class
- To write a function therefore it must be defined inside a class
- It is therefore written as a method
- We have to be careful to select the appropriate class in which to place these methods
- Methods are often called “member functions” because they are functions that belong to a class

n^n table

- Problem
  - Print out a table comprising the integers n and n^n, for n = 1, 2, ..., 10

1	1
2	4
3	27
4	256
.	.
.	.
.	.

Solution

Problem

- Problem: How to calculate n^n? (i.e. the power function or pow)
- Solutions:
  - Use a library function
  - Or, write your own function

Using a pre-defined library function

- Java method Math.pow( ) takes two real numbers as arguments and returns a real number

```
public class PowerTable
{
    // Computes the values of n^n for n = 1,
    // 2, ..., 10 using the predefined function pow
    public static void main(String[ ] args)
    {
        int n = 0;
        do
        {
            n = n+1;
            System.out.println(n + " " + Math.pow(n,n));
         }
         while (n < 10);
    }
}
```

Using a user-defined function

```
// Returns x to the power of y
// Pre-condition: y >= 0
public static double myPow(double  x, int  y)
{
    double z;
    int counter;
    if (y == 0)
    {
        z = 1;
    }
    else
    {
        counter = 1;
        z = x;
        while (counter < y)
        {
            z = z * x;
            counter = counter + 1;
        }
    }
    return z;
}
```

Class exercise: max( ) function

- Problem
  - Write pseudocode for a function max( ) that takes two integers as arguments and returns the bigger of the two

Solution

Program using max( ) function

- Problem
  - Write an algorithm that uses function max( ) to compute the maximum of four numbers

Solution

Why use functions?

- Functions greatly assist algorithm development
  - By breaking tasks into smaller sub-tasks
- Functions re-use code
  - Library code (APIs in Java)
  - User code
  - Independent of algorithm in which they are used
  - Reduces redundant code
- Functions enhance maintainability

Why use functions?

- Functions make algorithms easier to read and understand
- Functions can be implemented by separate teams
- Libraries of useful functions can be developed

- Object-oriented analysis and design

Solving problems the OO way

- Analyse the problem (OO Analysis)
- Design the solution (OO Design)
- Code the design (OO Programming)
- Test the program

Object-oriented analysis*

- When we try to solve a problem in the object-oriented paradigm we first analyse the problem
- The major steps are:
  1. Identify the classes and objects
  2. Identify any relationships between them
  3. Identify the responsibilities of the classes and objects
  4. Identify the attributes
  5. Identify the operations

* Dillon and Tan, Object-Oriented Conceptual Modeling

Identifying classes and objects

- Examine natural language descriptions
- In general look for:
  1. Improper nouns → classes
     e.g. toaster, student
  2. Proper nouns → instance objects
     e.g. Fred Smith

Identifying classes and objects

- Key abstractions to look for:
  1. Roles or functions of humans
  2. Occurrences
  3. Locations
  4. Things or products
  5. Organisational units

Identifying classes

- Loose coupling
  - Dependencies between classes should be kept to a minimum
- Strong cohesion
  - A class should do or be one thing only

Identifying classes and objects

- Points to note
  - Classes and objects must have an independent existence
  - Only specify classes and objects that are related to the problem and need to be stored for later use
  - Avoid computer-related or implementation-related aspects

Determining relationships between classes

- Relationships
  1. Inheritance relationships
  2. Instance-of relationships
  3. Aggregation relationships
  4. Associations

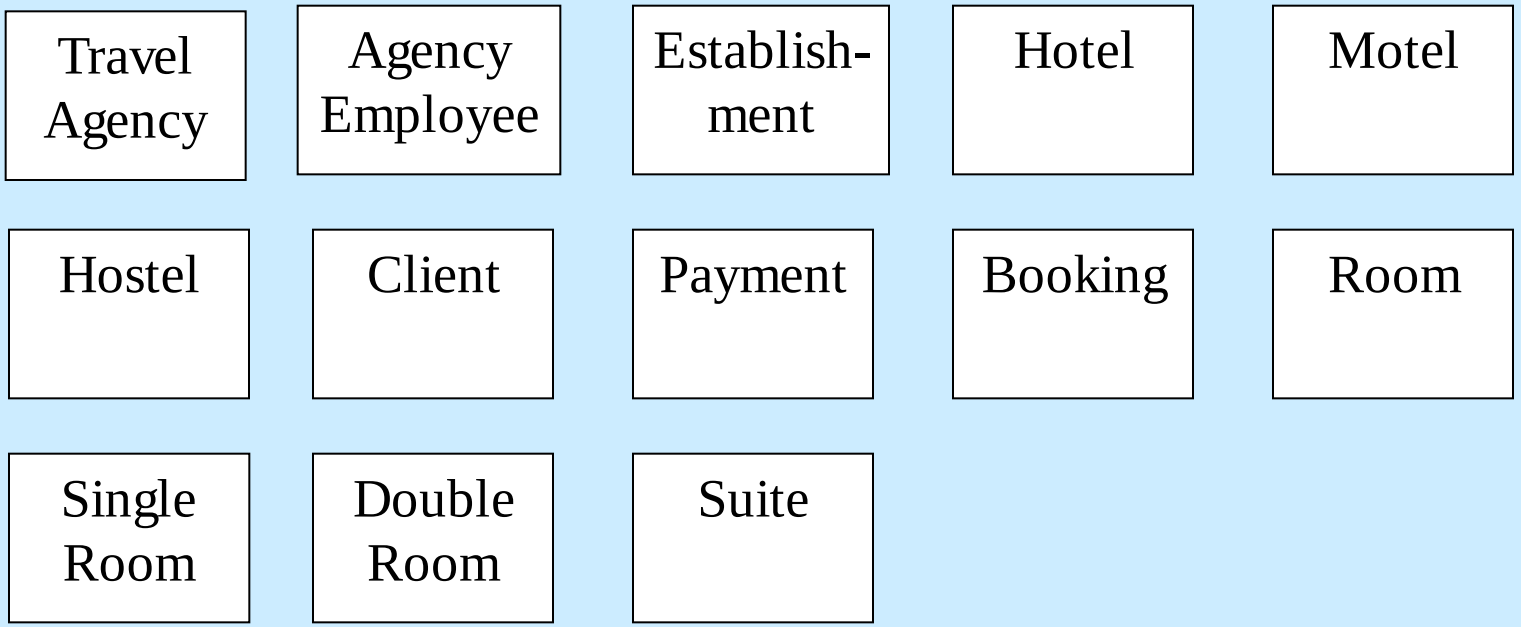
Example

- Travel agents book accommodation for their clients at various hotels, motels, and hostels. For each client, there must be an address and a phone number to contact in case of any changes. The client can pay for their room by cheque, cash or credit card.
- Each establishment has a rating from 1 to 5 stars and within an establishment each room may be graded standard or deluxe. People may book single rooms, double rooms, or suites. Each different type and standard of room has a different rate.
- Rooms themselves can have varying numbers and types of beds, and different amenities. Some clients may want to book a room in a quiet area of the hotel or a room on a particular floor. The system needs to cater for these requests.

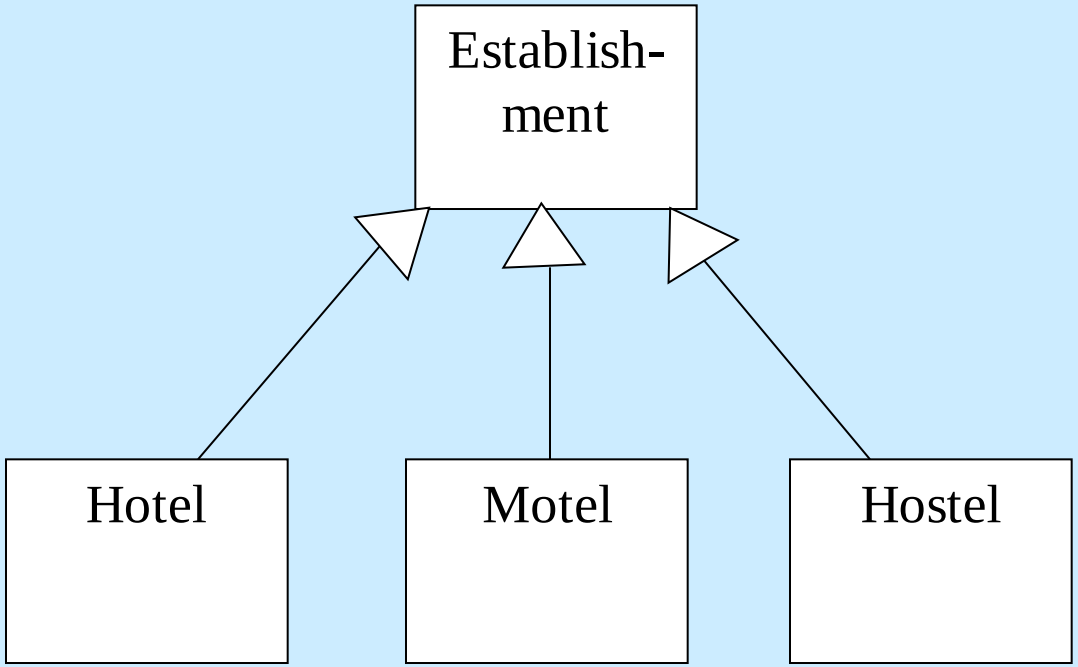
Example

- The roles or functions of humans
  - CLIENT
  - AGENCY EMPLOYEE
- Occurrences
  - BOOKING
  - PAYMENT
- Locations
  - HOTEL
  - MOTEL
  - HOSTEL
  - ESTABLISHMENT
- Things or products
  - ACCOMMODATION
  - ROOM
  - SINGLE ROOM
  - DOUBLE ROOM
  - SUITE
- Organisational units
  - TRAVEL AGENCY

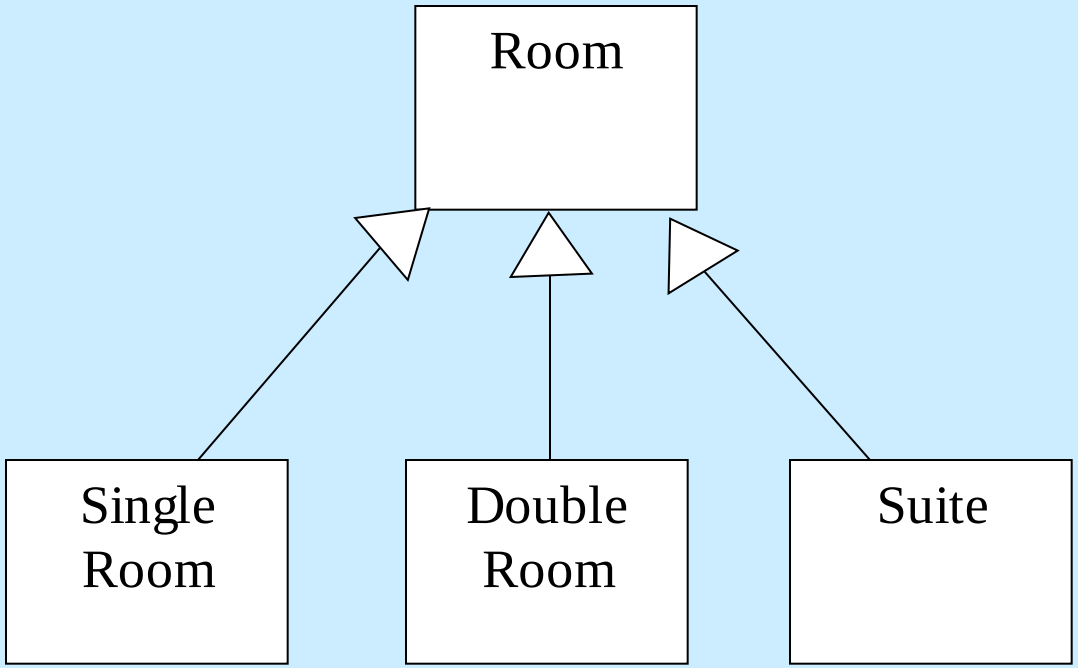
Example


Travel Agency
Agency Employee
Establish-ment
Hotel
Motel
Hostel
Client
Payment
Booking
Room
Single Room
Double Room
Suite

Example

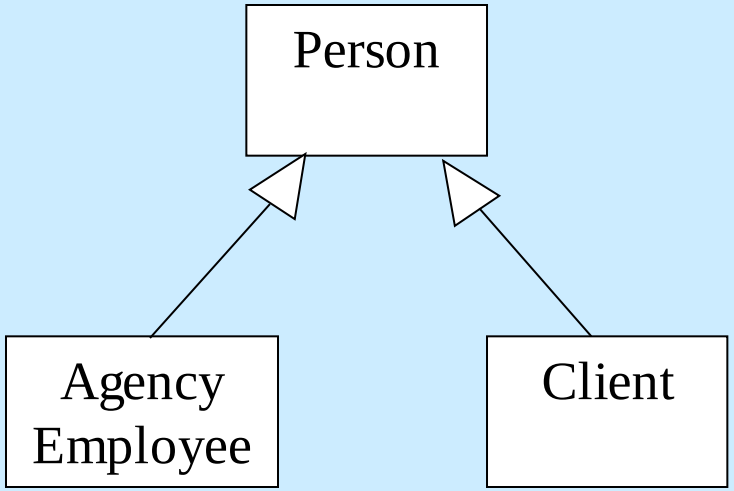


Example

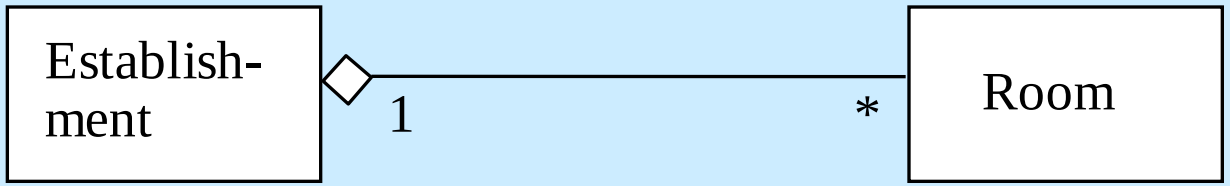


Inheritance relationships

Example

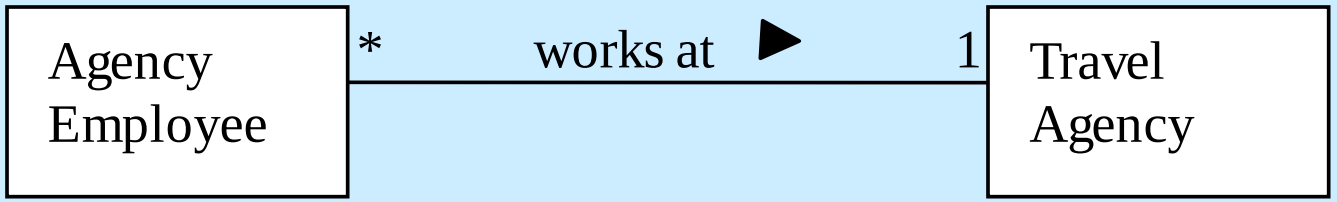


Example



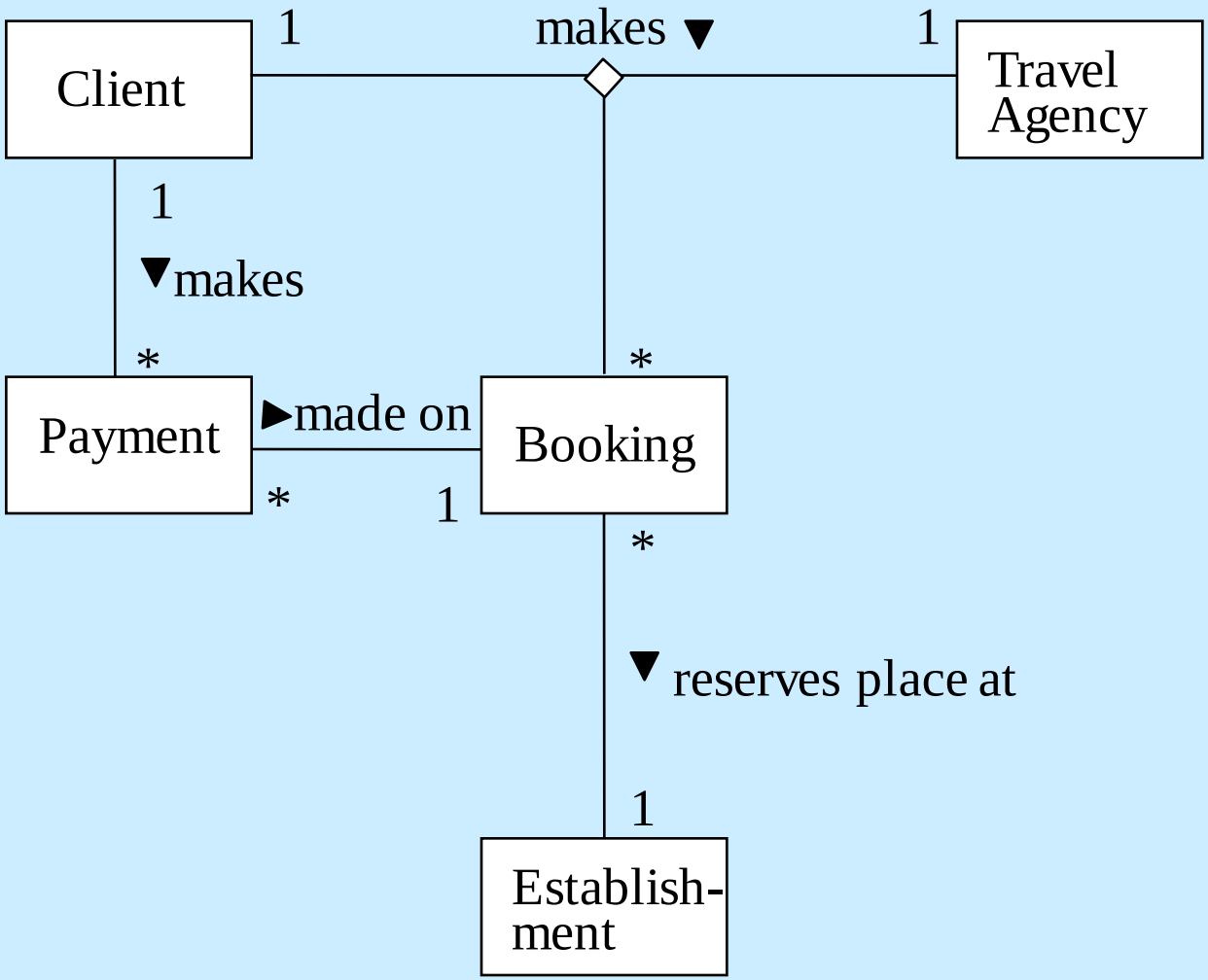
Aggregation relationships (part of, contains, has a)
E.g. An establishment contains rooms

Example



Other relationships between classes are called associations

Example


Client
1
makes
1
Travel Agency
1
makes
*
*
Payment
made on
Booking
*
1
*
reserves place at
1
Establish-ment

Identifying the attributes

- To identify attributes:
- For each object or class, list the features that describe it or its state
  - Adjectives
  - Possessive terms related to the object
  - Varying states of an object that are relevant
- Do not record attributes that are not likely to be used

Example

- CLIENT
  - Name
  - Address
  - Phone Number
- AGENCY EMPLOYEE
  - Staff Number
  - Commission Rate
  - Commission
- BOOKING
  - Establishment
  - Room Type
  - Dates
  - Cost
  - Requests
- PAYMENT
  - Type <Cheque, Cash, Credit Card>
  - Amount

Example

- ESTABLISHMENT
  - Name
  - Address
  - Number of Rooms
  - Rating <1 Star .. 5 Stars>
- ROOM
  - Beds
  - Amenities
  - Grading <Standard, Deluxe>
  - Rate

Identifying operations

- Consider the responsibilities of the class
- What operations does it need to carry out its responsibilities?
- Each operation that is performed may be
  - A single operation that does not use any other operation to carry out its task
  - An operation that invokes other operations in the same object, or sends messages to invoke operations in other objects or classes

Identifying operations

- Identify necessary service methods (accessor / mutator operations)?
- Consider which actions are necessary to solve the problem
- Consider operations that deal with exceptional or error situations?

Identifying operations*

- What does the class know how to do?
- What is the class expected to do for others?
- What are the responsibilities of the class?

** Reed, Developing Applications with Java and UML*

Further considerations

- If attributes or operations are common to a number of subclasses, consider putting them in the superclass
- Be careful when dealing with relationships: make sure you allocate attributes and operations to the appropriate place

Example

- CLIENT
  - Make New Booking
  - Change Details
  - Look Up Details
- BOOKING
  - Calculate Booking Cost
  - Create New Booking
  - Cancel Booking
  - Make Payment
  - Change Details
  - Look Up Details
- ESTABLISHMENT
  - Check Availability
  - Set Room Status
  - Change Details
  - Look Up Details
- ROOM
  - Change Details
  - Look Up Details
  - Check Availability

Next lecture

- Java variables, assignment and expressions
- Basic input and output