

## Relationships

Decide whether the statement pertains rather to an association, an aggregation, a composition or a generalization-specialization relationship?

- A country has a capital.
- A file is a regular file or a directory.
- A file belongs to a directory.
- Files contain records.
- A polygon is composed of segments.
- A person uses a programming language.



## Relationships

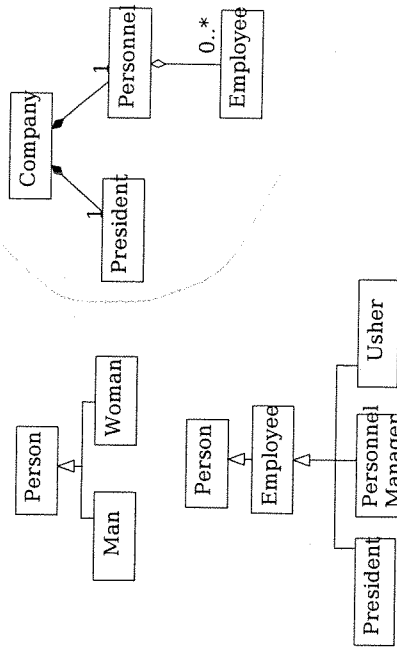
Decide whether the statement pertains rather to an association, an aggregation, a composition or a generalization-specialization relationship?

- A country has a capital.
- aggregation or association
- A file is a regular file or a directory.
- generalization-specialization
- A file belongs to a directory.
- aggregation (Unix), composition (Windows)
- Files contain records.
- composition
- A polygon is composed of segments.
- aggregation
- A person uses a programming language.
- association

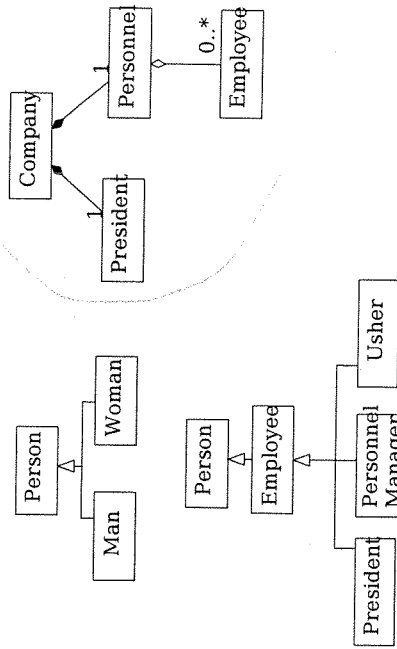


## Employees

Using a Class Diagram, show the generalization and aggregation relationships between: personnel, woman, employee, man, personnel manager, usher, president, person, company.



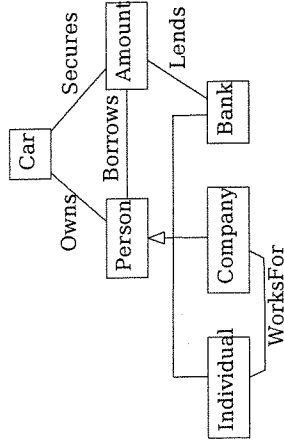
## Employees



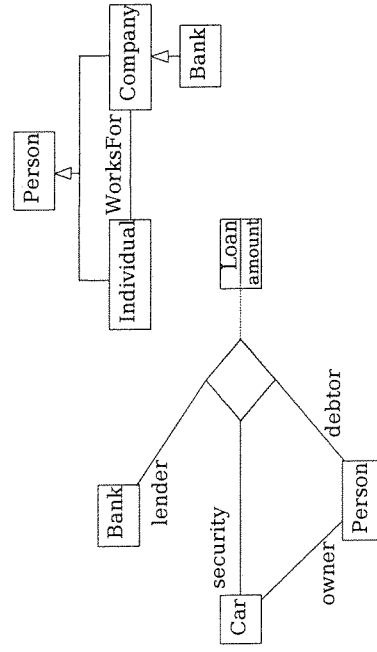
## Automobiles, statement

- Sketch the Class Model for the following classes and associations:
  - A person can work for one or several companies.
  - A car is owned by a person, a bank, or a company.
  - Banks give loans for buying cars.
  - A loan can be secured against a car.
- Add multiplicities to the associations, and state possible other constraints (invariants).

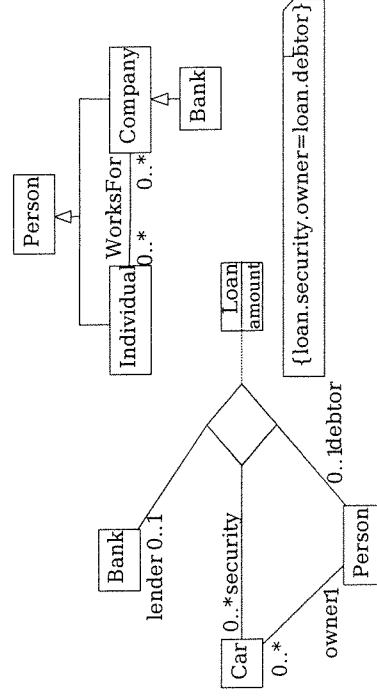
## Automobiles, for discussion



## Automobiles without multiplicities

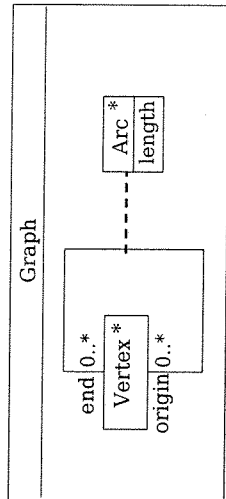


## Automobiles with multiplicities



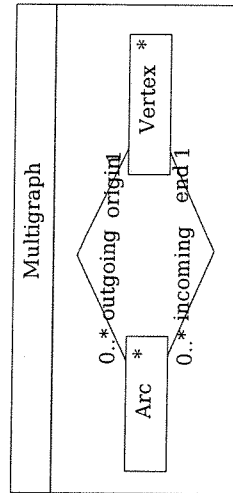
## Graph

Develop a Class Model, with multiplicities, for a graph. Recall that a graph is composed of a set of vertices (nodes) and a set of arcs (edges), each arc connecting two vertices. You might also want to show how to model arc lengths.



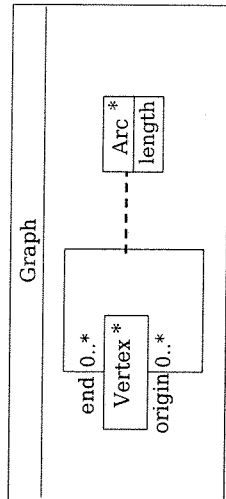
## Multigraph

Show how multigraphs can be modeled. Recall that in a multigraph, several arcs might connect the same two vertices.



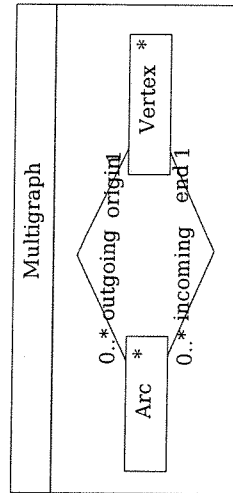
## Graph

Develop a Class Model, with multiplicities, for a graph. Recall that a graph is composed of a set of vertices (nodes) and a set of arcs (edges), each arc connecting two vertices. You might also want to show how to model arc lengths.



## Multigraph

Show how multigraphs can be modeled. Recall that in a multigraph, several arcs might connect the same two vertices.



## Terminal

Provide declarations for the entities needed to describe the exchange of information by messages between a system and a terminal "actor". With a terminal, characters can be input, one at a time, and the terminal can display characters, one at a time, in three modes: regular, inverted, and underlined.

## Terminal

Provide declarations for the entities needed to describe the exchange of information by messages between a system and a terminal "actor". With a terminal, characters can be input, one at a time, and the terminal can display characters, one at a time, in three modes: regular, inverted, and underlined.

Input (c: Character) -- input message (Terminal)  
 Display (c: Character, m: Mode) -- output message (Terminal)  
 type Character = character set of the terminal  
 type Mode = enum {normal, inverted, underlined}

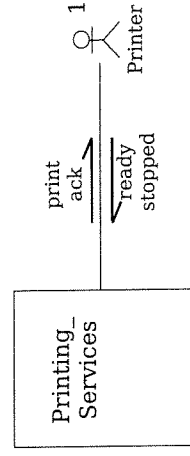
## Printer: Statement

The printer "actor" provides printing lines of characters, one line at a time, and informs the system when printing of a line is terminated. The printer also informs the system when an incident makes it impossible to continue printing. Some possible problems are: out of paper, paper jam, lack of ink, and some other cases that will be defined later on during the project, and which should be easy to add when time comes. When an incident arises, the system must acknowledge the error, deciding if the current printing should be canceled or if the printer should retry.

1. Establish an Environment Model
2. Provide message declarations for both the input and output messages. If needed, provide also data type declarations.

Test 1999–2000

## Printer: Environment Model



## Printer: Message Declarations

Output messages:

Print (line: String)  
Ack (response: Directive)

Input messages:

Ready  
Stopped (diagnostic: Problem)

type Directive = enum {cancel, retry}

type Problem = enum {out-of-paper, paper-jam, lack-of-ink}

-- The enumeration is easy to complete if needed.



## Gas Station

A gas station is to be set up for fully automated operation. Drivers swipe their credit card through a reader connected to the pump, the card is verified by communication with a credit card company computer and a credit limit is granted. The driver may then take the fuel for up to the granted amount of money. When fuel delivery is complete and the fuel-dispensing gun is returned to its holster, the driver's credit card account is debited with the cost of the fuel taken. The credit card is returned after debiting. If the card is invalid, it is returned by the pump before fuel is dispensed.

Environment Model part used for test doctoral school 2000-2001.



## Gas Station: Environment Model

The purpose of this exercise is to build the Environment Model. Draw on your own experience to complete the statement so it works as you are used to.

1. Provide the list of actors.
2. Provide the list of input messages, with their main parameters/attributes.
3. Provide the list of output messages, with their main parameters/attributes.
4. Show the Environment Model by a diagram.

Sommerville, exercise 12.7



## Gas Station: Environment Model

1. Provide the list of actors

Reader

Credit Company Computer

Holster

Fuel Gun

Notes:

Driver is only an "indirect" actor and can/could be omitted.

Card is passive, and therefore not an actor. It is used as a data medium.

Pump is a synonym for the System.

The Fuel(-dispensing) Gun is used by the driver to start/resume/stop fueling. It also stops fueling when it senses a full tank.



## Gas Station: Environment Model

2. Input messages, with their main parameters/attributes

CardInserted (id: AccountID)

InvalidCard ()

ValidCard (limit: Money)

FuelGunReturned ()

StartFueling () -- The user arms the gun handle.

StopFueling () -- The user releases the gun handle, or the tank is full

3. Output messages, with their main parameters/attributes

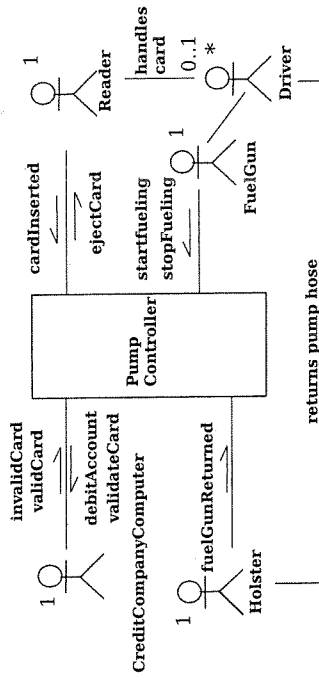
EjectCard ()

DebitAccount (id: AccountID, amount: Money)

ValidateCard (id: AccountID)

## Gas Station: Environment Model

4. Show the Environment Model by a diagram

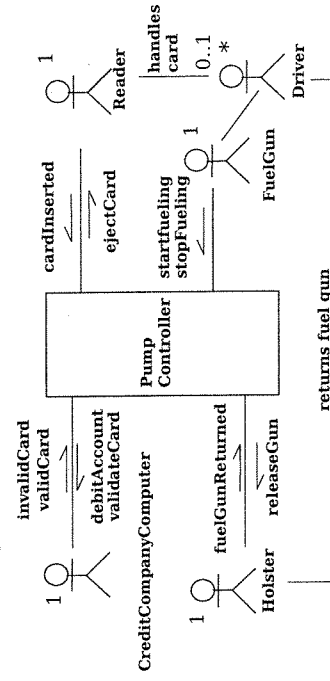


## Gas Station: Environment Model

5. In some gas stations, the fuel gun is locked in the holster and only released when fuel dispensing is granted. We will suppose that the fuel gun is locked upon return by the holster automatically, i.e. without intervention of the system.

Change the Environment Model to take into account this additional requirement.

6. You might also want to add a small display to the system that informs the user about the system state, and actions to be taken, e.g. "card use denied", "fuel now!", etc.



## Gas Station: Environment Model

5. Releasing/locking the fuel gun.

