5. *Classes: Friends, Containers, & Relationships*

Friends of a Class

– Provides Access to class Internals
– Breach of Encapsulation
– Discourage use of Friends

Friend Functions & Friend Classes

```
class Paragraph {
...
friend int wordCount(const Paragraph& thepara);
   // function WordCount is a friend of the class
friend class Page; // class Page is a friend of the class
};
```

Container/Collection Classes

–The Object is actually a container

–Holds a collection of other objects

```
class SetOfBoxes {
  Box* ptr_to_boxes;
...
  void addBox(Box& abox);
  Box* getBox(const String& boxlabel);
  Box* removeBox(const String& boxlabel);
};
```

Generalized Template Classes may be used.

Relationships among Classes: Association & Aggregation

Members of a Class

Attributes:

- a data value held by an object
- has no identity
- pure data value

Association Attributes:

- Relationships between Classes

Relationships among Classes

- Association (semantic relationship)
- has (owner/part, Aggregation) relationship
- Inheritance (is-a) relationship

Association

Examples of Association

- A company employs several persons. The company also owns several computers. Each person may be assigned one computer for the person's use.

- A lab has several computers. A student may reserve & use a computer for a certain period of time.

Visibility of Classes

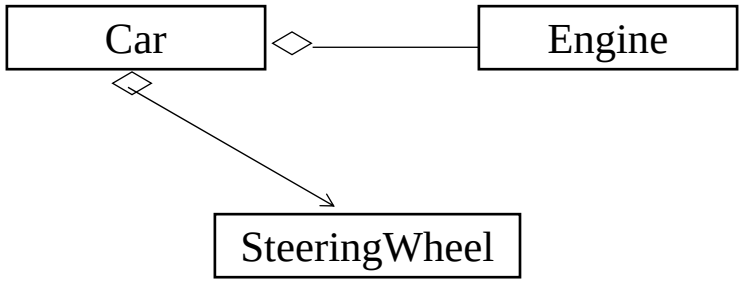
Ways that one object may be made visible to another

The supplier Object is

- global to the client
- a parameter to some operation of the client
- a part of the client object
- a locally declared object in the scope of the object diagram

Aggregation (has, whole/part)

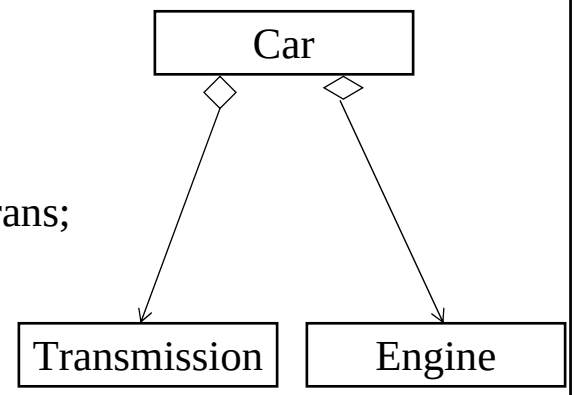
Expresses Containment (ownership)
(bidirectional Navigability to Engine)



Ways to Contain

- Aggregation by Value (Composite Aggregation)
- Aggregation by Reference

```
class Car {
...
   Engine* theEngine;
   // By Reference
   Transmission  theTrans;
   // By Value
};
```



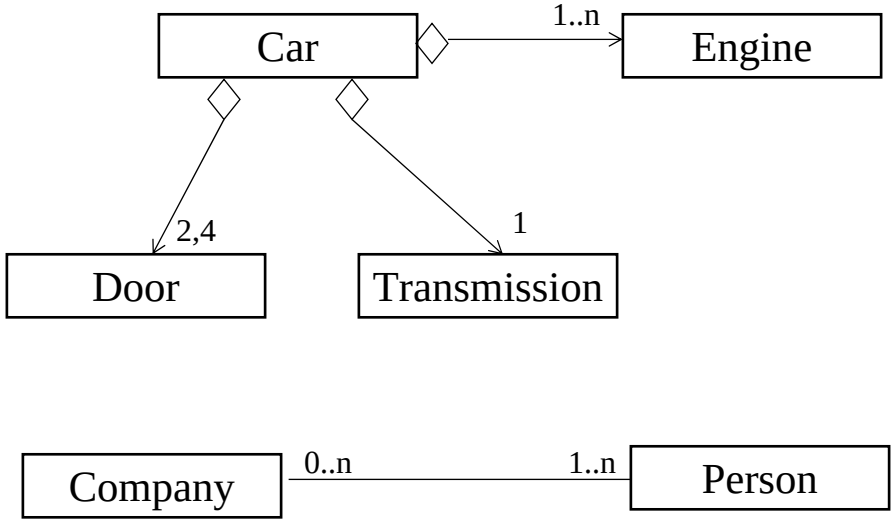
Containment : By Value Vs. By Reference

	By Value	By Reference
Creation	Created upon Creation of Container	Created, attached, detached, destroyed.
Destruction	Dies with Container	Dies with Container or earlier (unless detached)
Existence	Always exists	May or may not exist
Types	Container may contain only specific type	Container may contain specialized type.

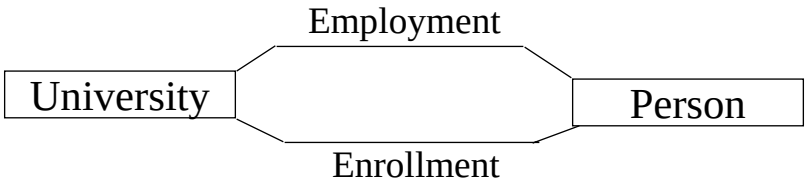
How would you express “Person Owns Car” ?

Cardinality of Relationship

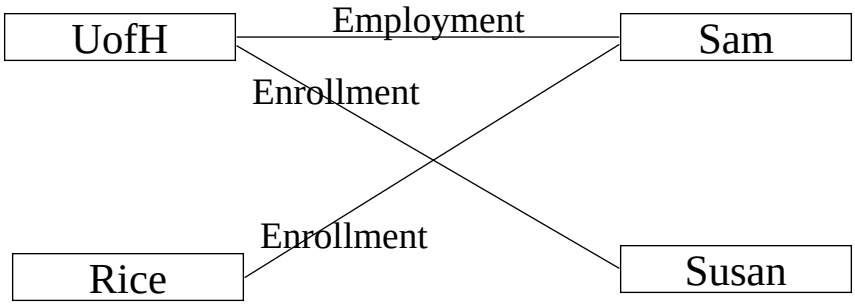
The number of Objects expressed in the Relationship.



Multiple Relationships between Classes



Example Object Diagram



Aggregation by Value : Writing the Constructor

```
class Elevator {
...
    Elevator (int bottomfloor, int topfloor);
    // Needs service floor numbers  to create an elevator
};

class Building {
    Elevators elev1, elev2;
...
    Building (int numoffloors);
};

Building::Building (int numoffloors) : elev1(0, numoffloors - 1), elev2 (1,
    numoffloors)
{// body of Building constructor }
```

Lab Work: Details provided on-line.