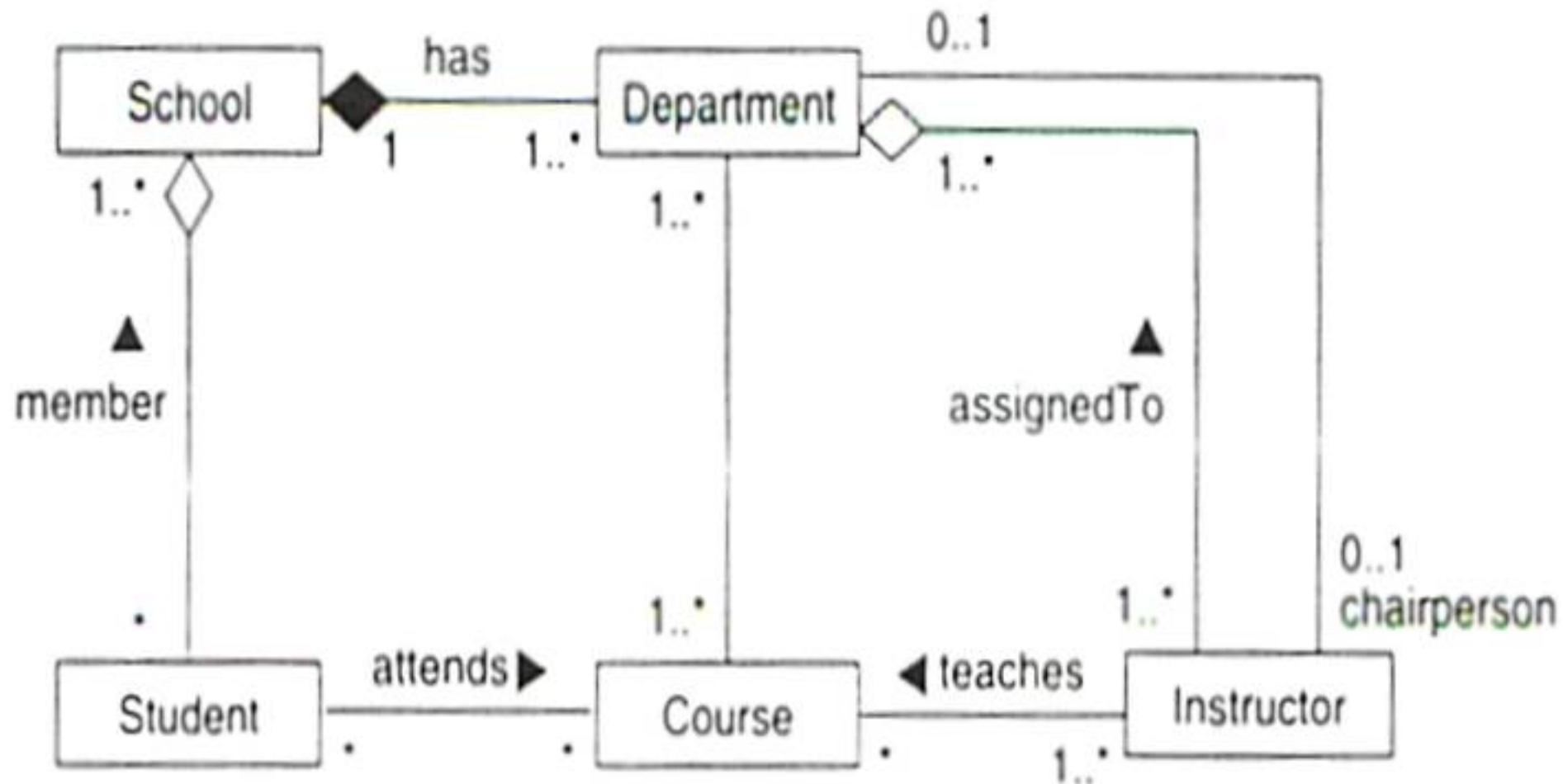




UML (Class Diagrams)

Aggregation and Composition



Structural Relationships

- ❑ The departments may have chairpersons who are not instructors, or some students may also be instructors
- ❑ That does not mean model is wrong
- ❑ Model depends how we want to use it

Example

A person owns multiple documents. Each document consists of paragraphs that, in turn consists of characters.

Show relationships between different classes with multiplicities.

Common Modeling Techniques

- Modeling association, aggregation and composition relationships.
 - It is not always easy to determine which is the best relationship to use. It all has to do with your perspective of the problem domain.
 - There is not always one best solution.
 - Personal experience is your most valuable modeling tool.

- In many cases the set of operations (algorithms) are logically the same no matter what type of data is being operated upon.
- Example: Quicksort sorting / Stack
- Create a generic function / class, to define the nature of the algorithm / object, independent of any data.

The general form of a generic class declaration:

```
template <class Ttype> class class-name {  
...  
}
```

Creating a specific instance of that class:

```
class-name <type> ob;
```

Member functions of a generic class are themselves automatically generic.

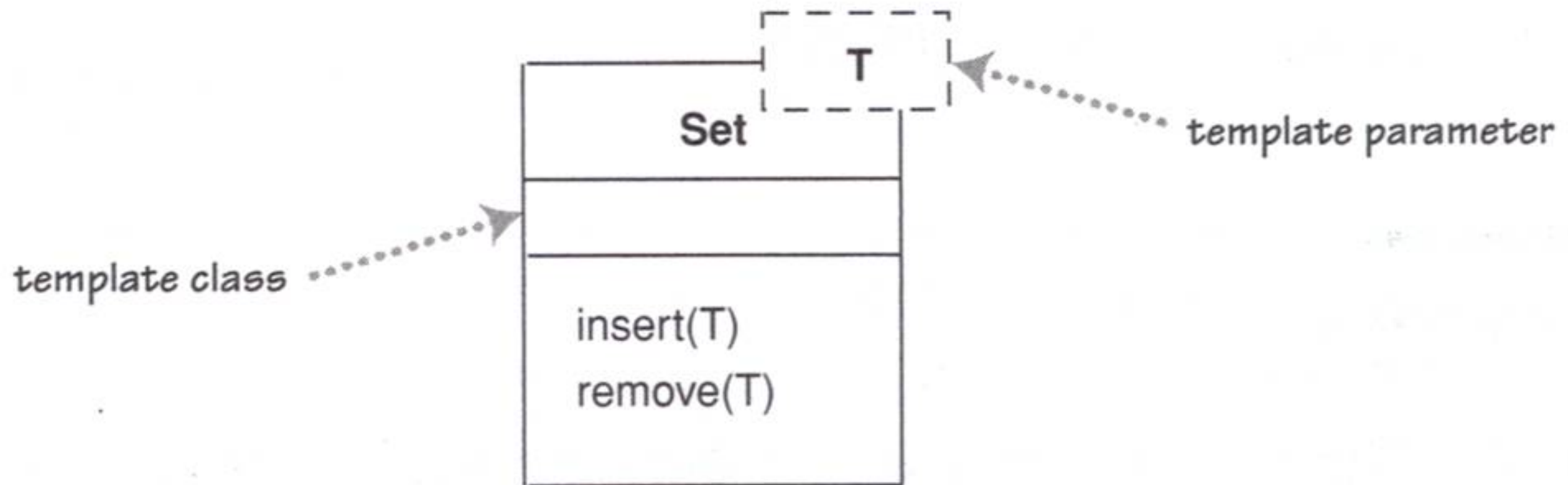
```
template <class StackType> class stack {  
    StackType stck[SIZE];  
    int tos;  
public:  
    stack() { tos = 0; }  
    void push(StackType ob);  
    StackType pop();  
};  
template <class StackType> void  
stack<StackType>::push(StackType ob) {  
    if(tos==SIZE) {  
        cout << "Stack is full.\n";  
        return;  
    }  
    stck[tos] = ob;  
    tos++;  
}
```

```
int main() {  
    stack<char> s1;  
    int i;  
    s1.push('a');  
    s1.push('b');  
    for(i=0; i<2; i++)  
        cout << "Pop s1: " << s1.pop() << "\n";  
    stack<double> ds1;  
    ds1.push(1.1);  
    ds1.push(3.3);  
    for(i=0; i<2; i++)  
        cout << "Pop ds1: " << ds1.pop() << "\n";  
    return 0;  
}
```


Template (Parameterized) Class

- Each of the template classes defines a family of classes
- Intended to facilitate software reusability.
- Used to automate the creation of class definitions.
- Similar to macros in assembly language.
- Essentially a class definition with the data types of certain attributes yet to be defined.
- Most commonly used to create container classes.
- Represented in UML as a dashed box in the upper right-hand corner of the class icon, which lists the template parameters.

Template Class



Template Class Derivation / Instantiation



UML 2.0

Figure 5.18 *Bound element (version 1)*

Implicit binding

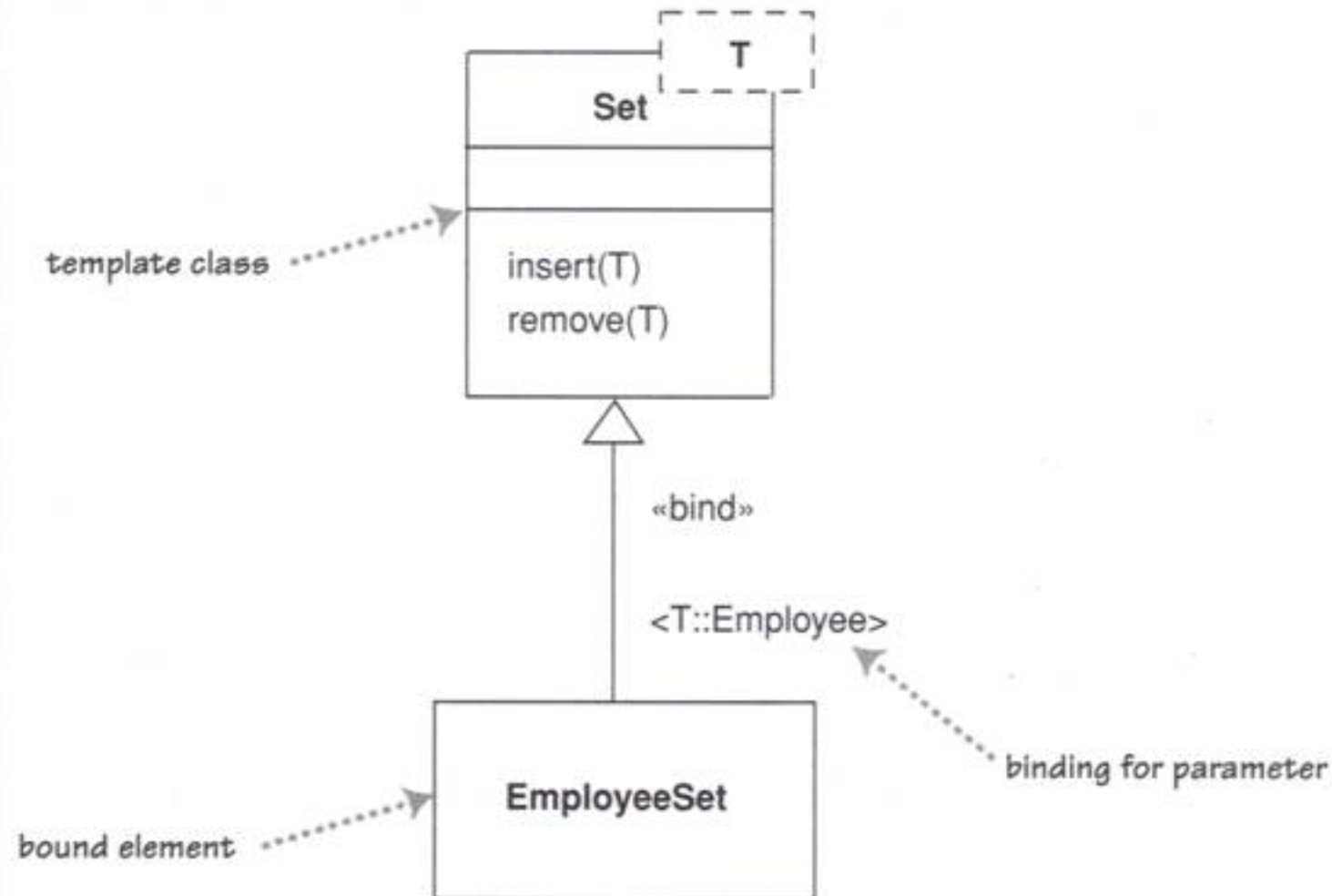
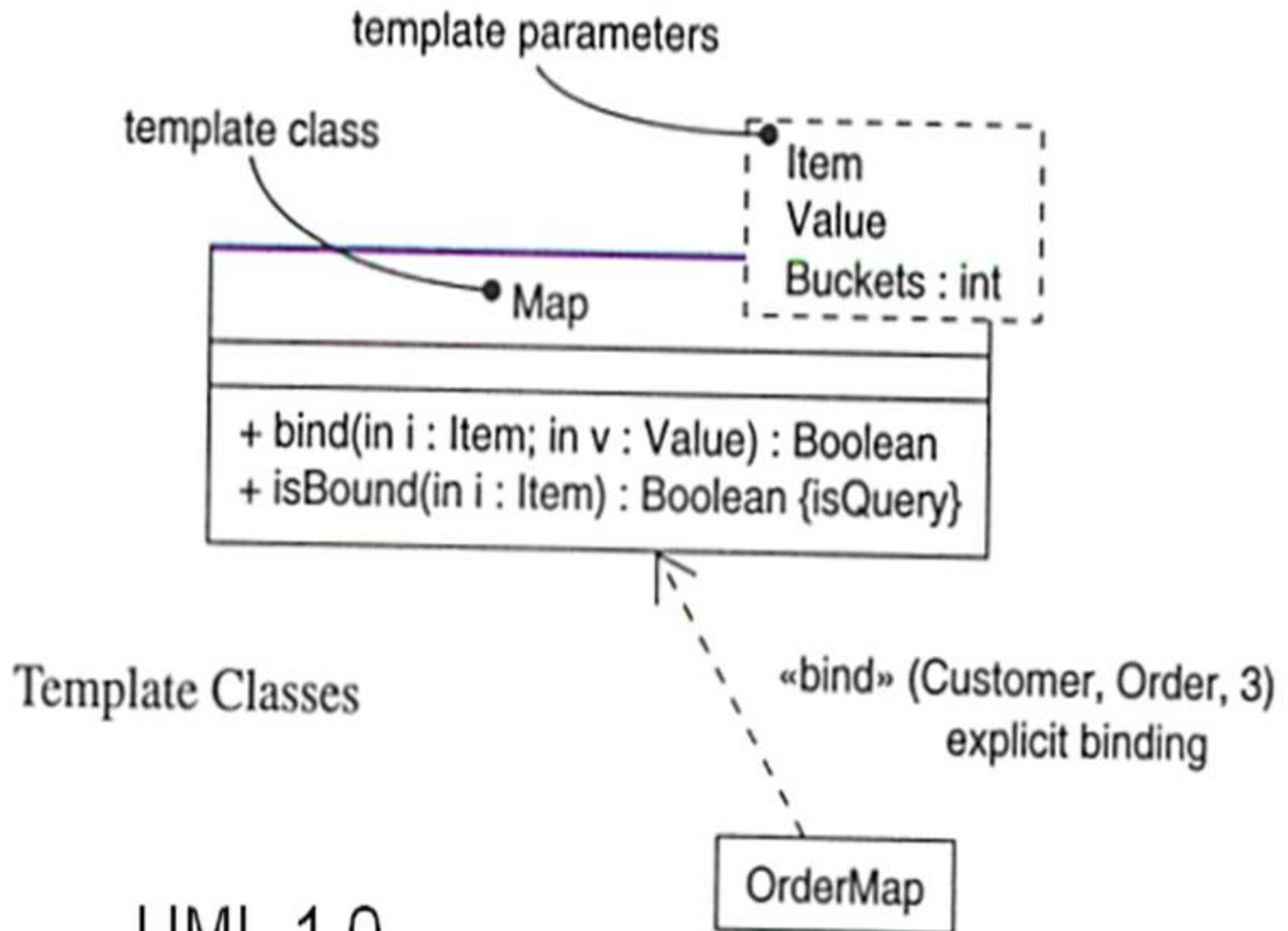


Figure 5.19 *Bound element (version 2)*

Explicit binding

Template Class



UML 1.0

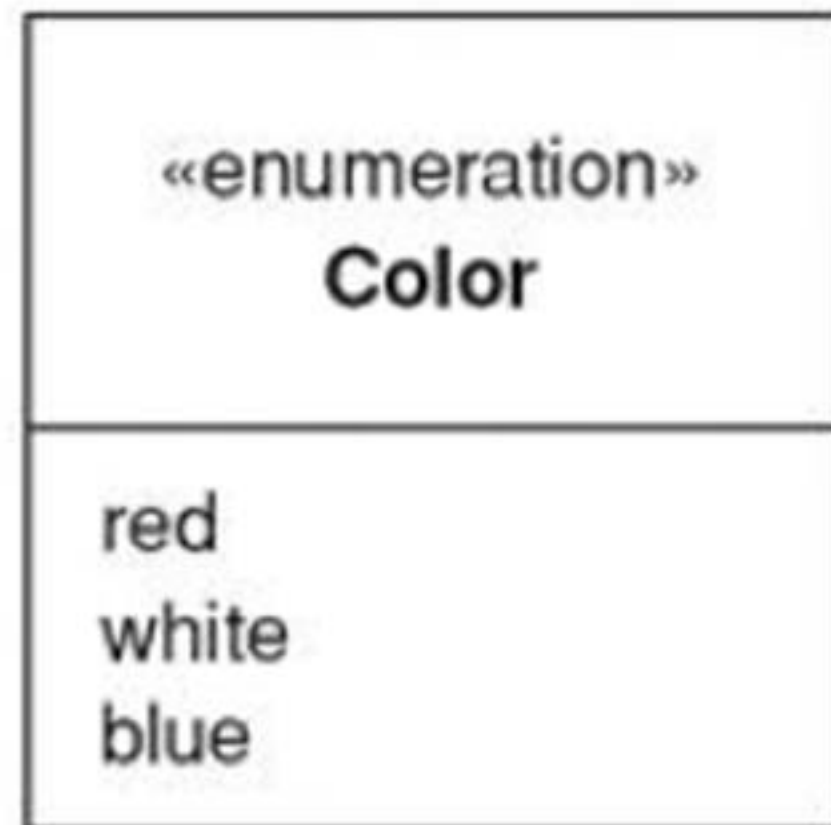
Active Class

- An active class has instances, each of which executes and controls its own thread of control.



Enumerations

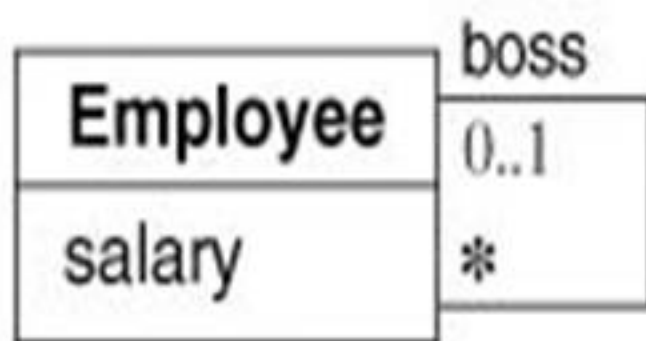
- Are used to show a fixed set of values that don't have any properties other than their symbolic value.



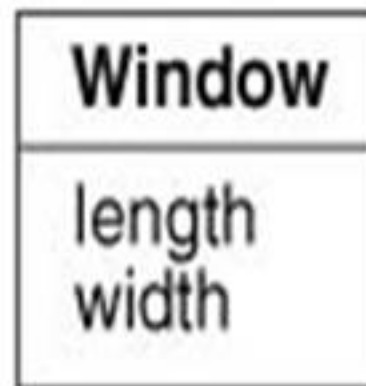
Constraints

A Boolean condition involving model elements such as objects, classes, attributes, links, associations etc.

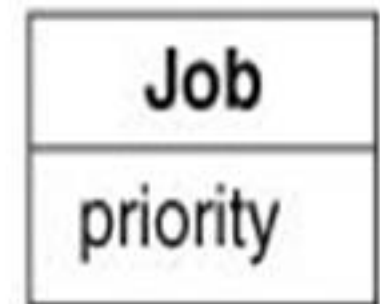
On Objects



{salary ≤ boss.salary}



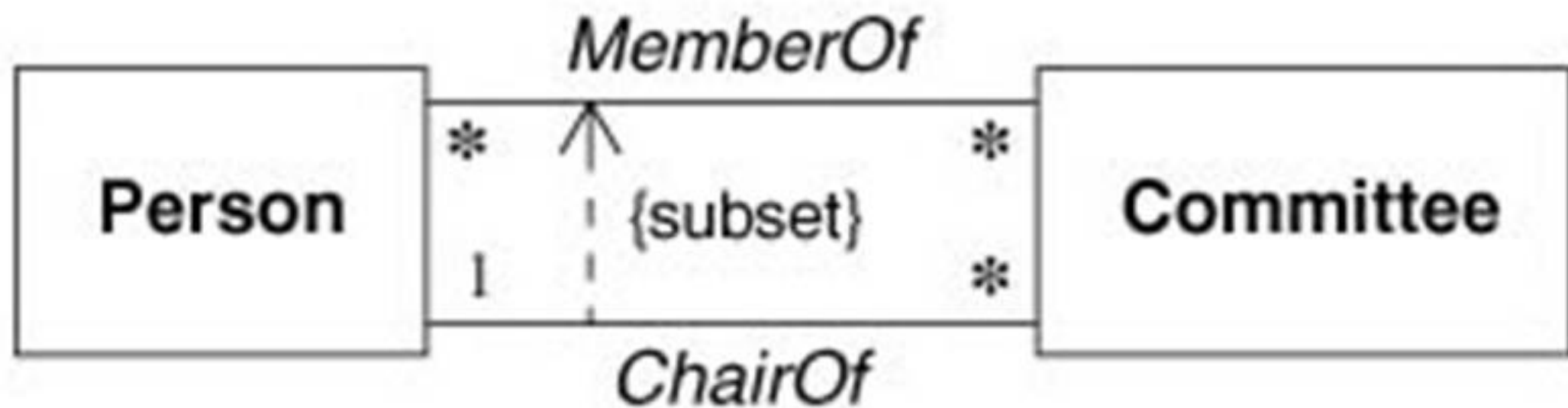
{0.8 ≤ length/width ≤ 1.5}



{priority never increases}


Constraints on Links / Associations

- Multiplicity
- Qualification
- Constraint {ordered}



Common Modeling Techniques

- Multiple class diagrams are required to model large systems.
- Each individual class diagram ...
 - Shows a single **aspect** of the system.
 - Contains only elements that are essential to understanding that aspect.
 - Provide details consistent with its **level** of abstraction.
 - Uses meaningful class and member names.
- Pointers to other classes are modeled as associations.

- 
- Deciding on the right set of abstractions for a given domain is the central problem in object-oriented design

The Meaning of Abstraction

- Abstraction is one of the fundamental ways that we as humans cope with complexity
- An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer
- An abstraction focuses on the outside view of an object and so serves to separate an object's essential behavior from its implementation