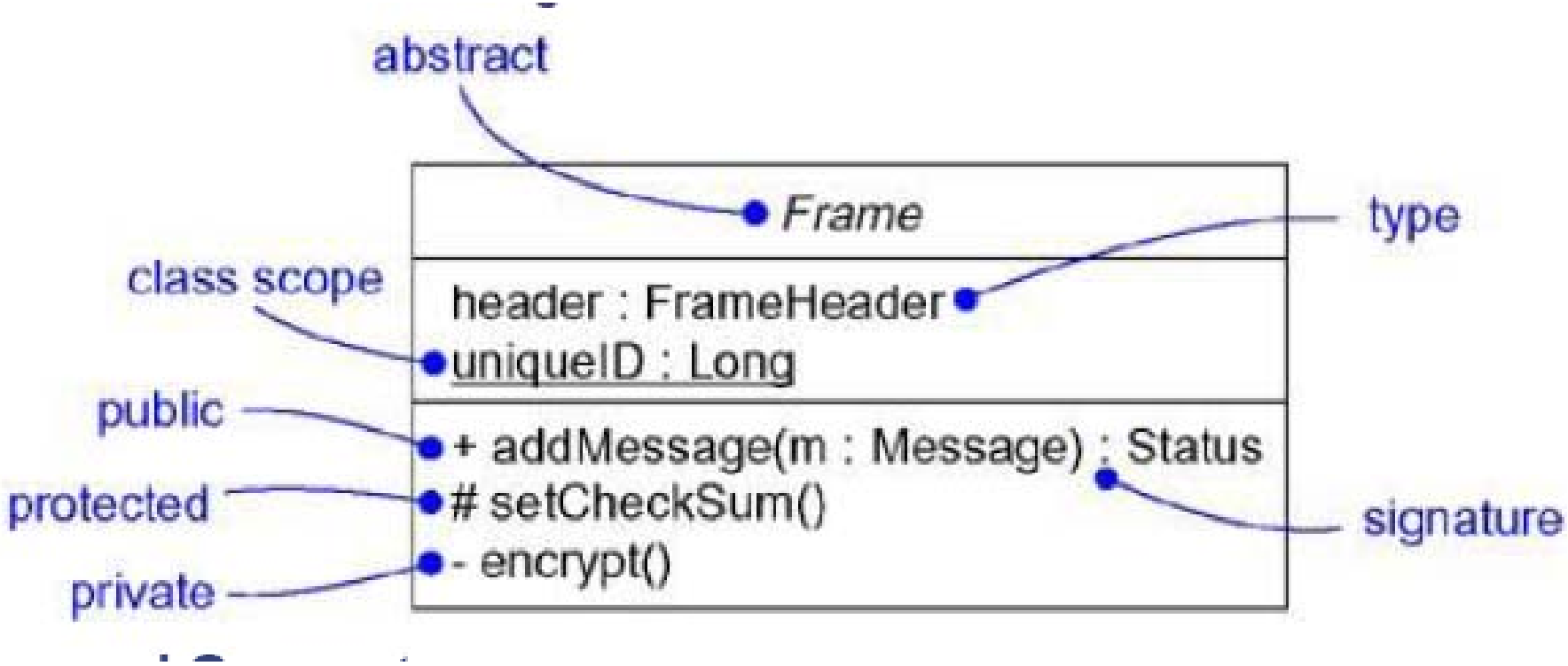
ADVANCED STRUCTURAL MODELING

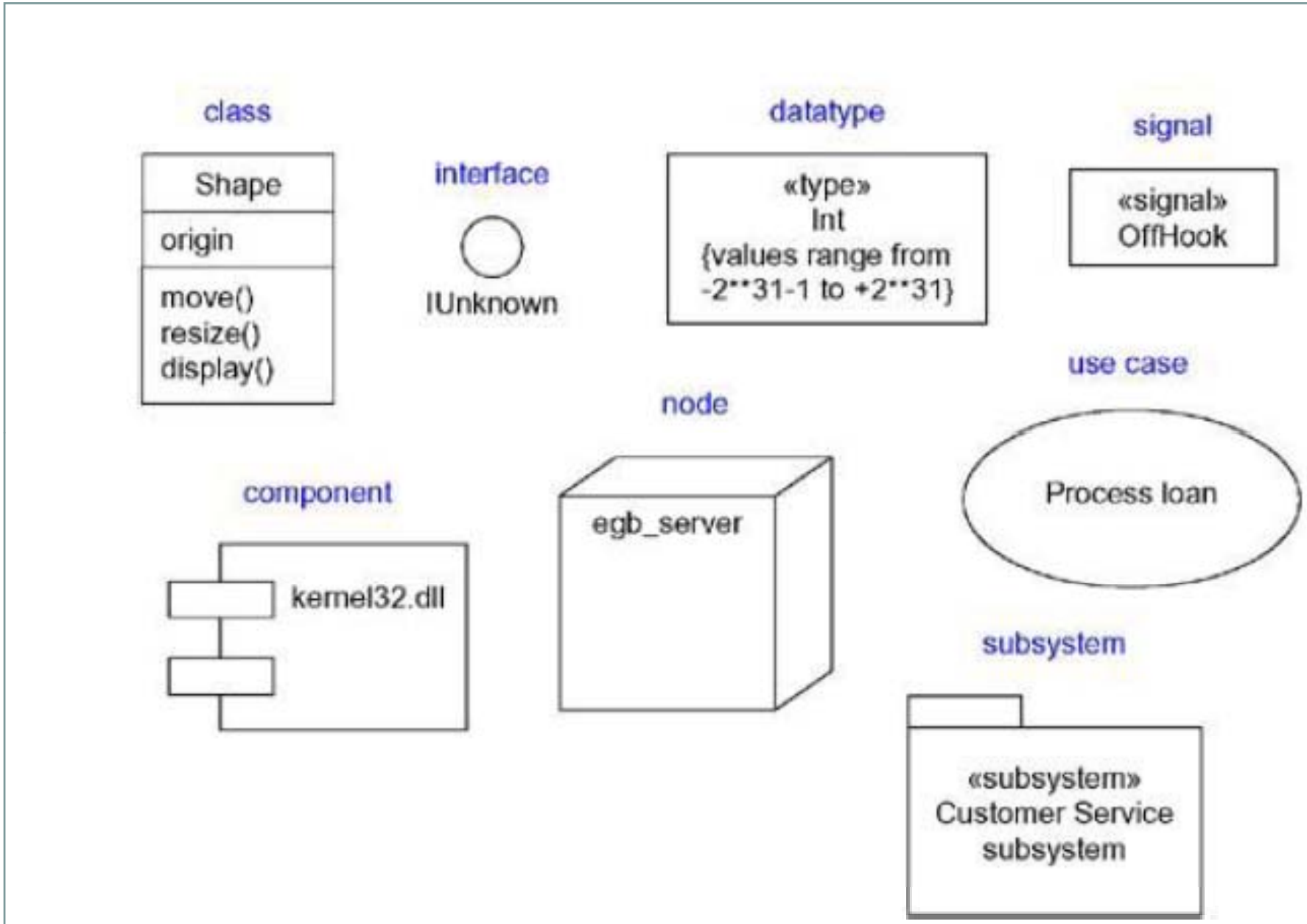
ADVANCED CLASSES

- **A classifier is a mechanism that describes structural and behavioral features.**

- **Classifiers include classes, interfaces, data types, signals, components, nodes, use cases, and subsystems.**



➢Interface	A collection of operations that are used to specify a service of a class or a component
➢Datatype	A type whose values have no identity, including primitive built-in types (such as numbers and strings), as well as enumeration types (such as Boolean)
➢ Signal	The specification of an asynchronous stimulus communicated between instances
➢ Component	A physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces.
➢ Node	A physical element that exists at run time and that represents a computational resource, generally having at least some memory and often processing capability
➢ Use case	A description of a set of a sequence of actions, including variants, that a system performs that yields an observable result of value to a particular actor.
➢Subsystem	A grouping of elements of which some constitute a specification of the behavior offered by the other contained elements

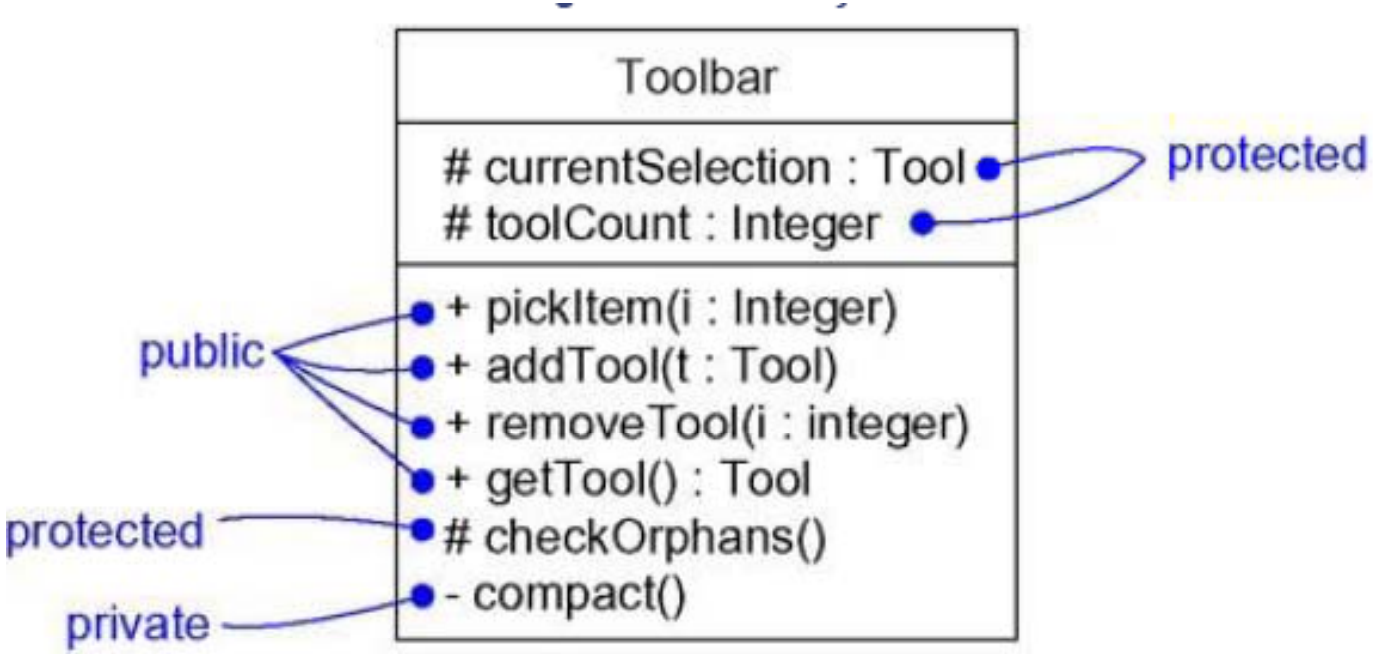

class
Shape
origin
move()
resize()
display()
interface
IUnknown
datatype
«type»
Int
{values range from
-2**31-1 to +2**31}
signal
«signal»
OffHook
use case
Process loan
node
egb_server
component
kernel32.dll
subsystem
«subsystem»
Customer Service
subsystem

Visibility

Public : Any outside classifier with visibility to the given classifier can use the feature; specified by prepending the symbol +

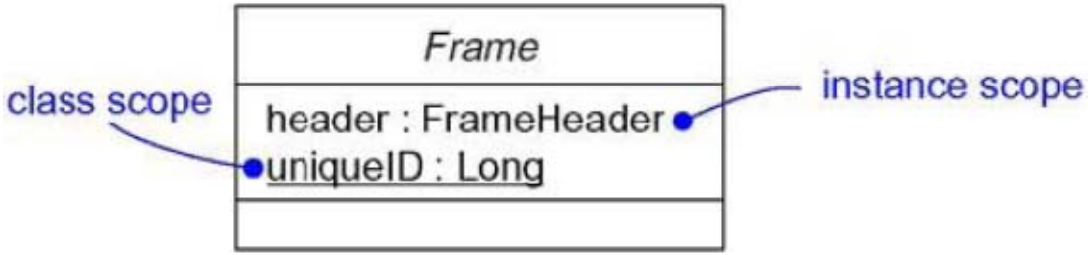
Protected: Any descendant of the classifier can use the feature; specified by prepending the symbol #

Private: Only the classifier itself can use the feature; specified by prepending the symbol -



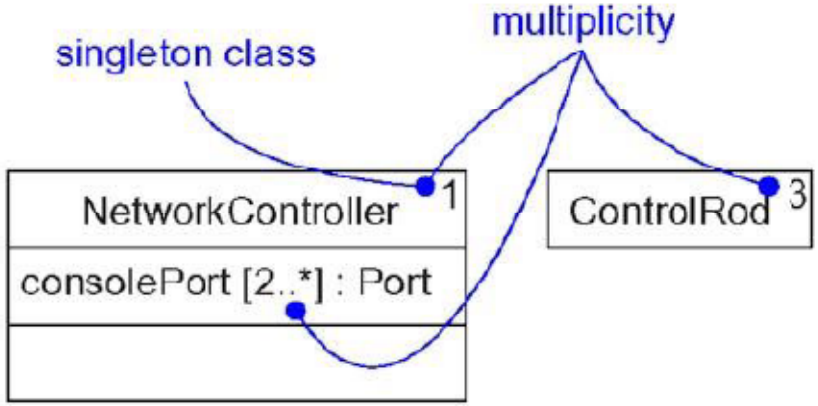
Scope

Instance - Each instance of the classifier holds its own value for the feature.
Classifier - There is just one value of the feature for all instances of the classifier.



Multiplicity

Multiplicity is a specification of the range of allowable cardinalities an entity may assume.



Attributes

[visibility] name [multiplicity] [: type] [= initial-value] [{property-string}]

origin	Name only
+ origin	Visibility and name
origin :	Point Name and type
head : *Item	Name and complex type
name [0..1] : String	Name, multiplicity, and type
origin : Point = (0,0)	Name, type, and initial value
id : Integer {frozen}	Name and property

There are three defined properties that you can use with attributes.

1. Changeable - There are no restrictions on modifying the attribute's value.
2. addOnly - For attributes with a multiplicity greater than one, additional values may be added, but once created, a value may not be removed or altered.
3. frozen - The attribute's value may not be changed after the object is initialized.

Operations

An operation specifies a service that can be requested from any object of the class to affect behavior; a method is an implementation of an operation.

[visibility] name [(parameter-list)] [: return-type] [{property-string}]

display	Name only
+ display	Visibility and name
set(n : Name, s : String)	Name and parameters
getID() : Integer	Name and return type
restart() {guarded}	Name and property

In an operation's signature, you may provide zero or more parameters, each of which follows the

syntax

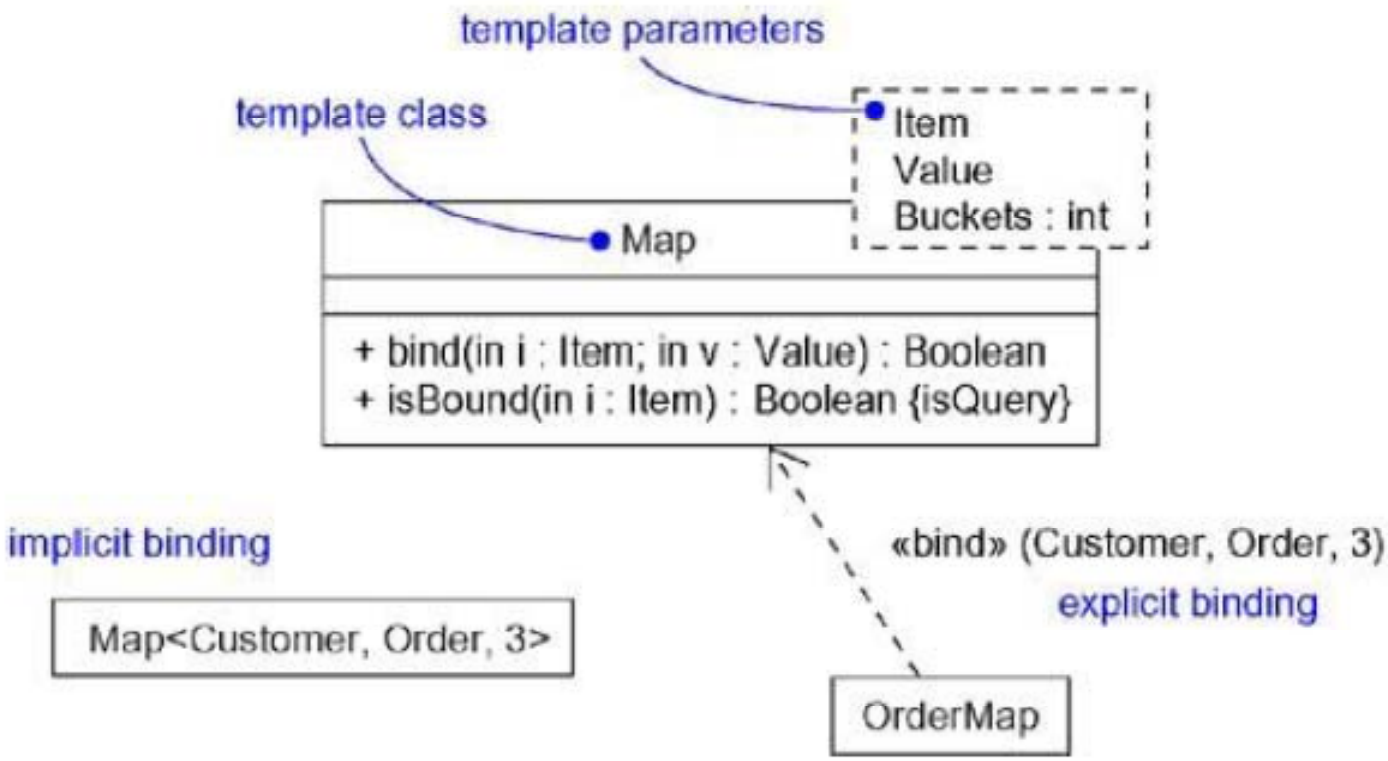
[direction] name : type [= default-value]

Direction may be any of the following values:

In - An input parameter; may not be modified
Out - An output parameter; may be modified to communicate information to the caller.
Inout - An input parameter; may be modified

Template Classes

A template includes slots for classes, objects, and values, and these slots serve as the template's parameters.



Standard Elements

1. metaclass - Specifies a classifier whose objects are all classes
2. powertype - Specifies a classifier whose objects are the children of a given parent
3. Stereotype - Specifies that the classifier is a stereotype that may be applied to other elements
4. utility - Specifies a class whose attributes and operations are all class scoped

Common Modeling Techniques

To model the semantics of a class

- Specify the responsibilities of the class.
- Specify the semantics of the class.
- Specify the body of each method using structured text or a programming language.
- Specify the pre- and post-conditions of each operation, plus the invariants of the class as a whole, using structured text.
- Specify a state machine for the class.
- Specify a collaboration that represents the class.

Reference

The Unified Modeling Language User Guide - *Grady Booch, James Rumbaugh, Ivar Jacobson* Addison-Wesley (International Student Edition)