USN | | | | | | | | | | |

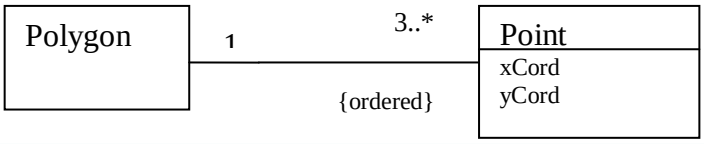
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – Sept. 2017

Sub:	OBJECT ORIENTED MODELING AND DESIGN					Sub Code:	10CS71	Branch:	ISE
Date:	20/09/2017	Duration:	90 min's	Max Marks:	50	Sem / Sec:	VII / A & B		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Describe the important characteristics of object orientation.	[06]	CO1	L1
(b)	Using the class diagram below, prepare an object diagram for the two triangles with a common side under the following condition: a) A point belongs to exactly one polygon b) A point belongs to one or more polygon.	[2+2]	CO2	L4



2 (a)	Explain aggregation and composition with suitable example.	[3+3]	CO2	L4
(b)	Draw class diagram for the following: a) Programmer uses computer language on projects. b) Worker is a butcher or baker or candlestick maker.	[2+2]	CO2	L1

USN | | | | | | | | | | |

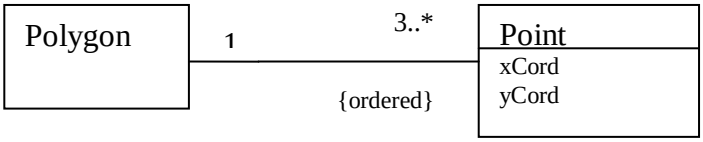
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – Sept. 2017

Sub:	OBJECT ORIENTED MODELING AND DESIGN					Sub Code:	10CS71	Branch:	ISE
Date:	20/09/2017	Duration:	90 min's	Max Marks:	50	Sem / Sec:	VII / A & B		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Describe the important characteristics of object orientation.	[06]	CO1	L1
(b)	Using the class diagram below, prepare an object diagram for the two triangles with a common side under the following condition: a) A point belongs to exactly one polygon b) A point belongs to one or more polygon.	[2+2]	CO2	L4



2 (a)	Explain aggregation and composition with suitable example.	[3+3]	CO2	L4
(b)	Draw class diagram for the following: a) Programmer uses computer language on projects. b) Worker is a butcher or baker or candlestick maker.	[2+2]	CO2	L1

3 (a)	Prepare a class diagram for the group of classes given below. Add at least three relationships (associations, generalization). Use association names where needed and show multiplicity. School, playground, principle, book, student, teacher, cafeteria, class room, rest room, computer.	[06]	CO2	L4
(b)	What is visibility? Explain with suitable example.	[04]	CO2	L1
4 (a)	Explain the concept of workaround and its approaches with suitable example.	[10]	CO1	L4
5 (a)	Explain state diagram and write state model for a telephonic line with activities.	[10]	CO2	L4
6 (a)	A simple digital watch has a display and two buttons to set it, the A button, and the B button. The watch has two modes of operation, display time, set time in the display time mode, the watch displays hours and minutes, separated by a flashing colon. The set time mode has two sub modes, set hours, set minutes. The button A selects modes. Each time it is pressed, the mode advances in the sequence: display, set hours, set minutes, display etc. Within the sub modes the button B advances the hours or minutes once each time it is pressed. Buttons must be released before they can generate another event. Prepare a state diagram of the watch.	[06]	CO2	L4
(b)	Explain qualified association with suitable example.	[04]	CO2	L4
7 (a)	Explain the following with the help of UML: a) Derived attributes b) Ordering c) Packages d) Enumeration e) Multiplicity (object and attributes).	[10]	CO2	L4

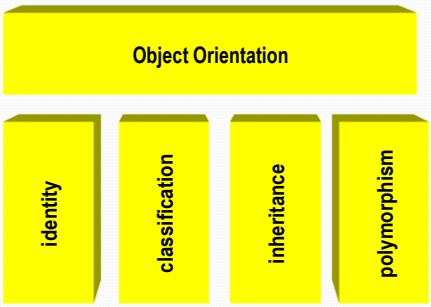
---------- (All the Best) ----------

3 (a)	Prepare a class diagram for the group of classes given below. Add at least three relationships (associations, generalization). Use association names where needed and show multiplicity. School, playground, principle, book, student, teacher, cafeteria, class room, rest room, computer.	[06]	CO2	L4
(b)	What is visibility? Explain with suitable example.	[04]	CO2	L1
4 (a)	Explain the concept of workaround and its approaches with suitable example.	[10]	CO1	L4
5 (a)	Explain state diagram and write state model for a telephonic line with activities.	[10]	CO2	L4
6 (a)	A simple digital watch has a display and two buttons to set it, the A button, and the B button. The watch has two modes of operation, display time, set time in the display time mode, the watch displays hours and minutes, separated by a flashing colon. The set time mode has two sub modes, set hours, set minutes. The button A selects modes. Each time it is pressed, the mode advances in the sequence: display, set hours, set minutes, display etc. Within the sub modes the button B advances the hours or minutes once each time it is pressed. Buttons must be released before they can generate another event. Prepare a state diagram of the watch.	[06]	CO2	L4
(b)	Explain qualified association with suitable example.	[04]	CO2	L4
7 (a)	Explain the following with the help of UML: a) Derived attributes b) Ordering c) Packages d) Enumeration e) Multiplicity (object and attributes).	[10]	CO2	L4

---------- (All the Best) ----------

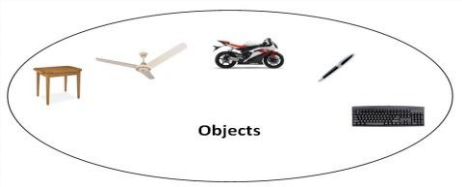
1. A) Describe the important characteristics of object orientation. (6M)

Basic Principles of Object Orientation



1.Identity

- Means that data is quantized into discrete(individual), distinguishable entities .
- Eg: Queen in chess game, monitor, bicycle, binary tree, cot, dinning table, etc...



2.Classification

- Means that objects with the same data structure (attributes) and behavior (operations) are grouped into classes.
- Eg: class bycycle

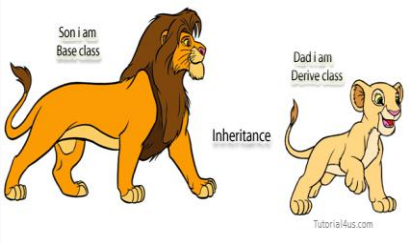
```
{   int frame_size, wheel_size, no_of_gears;

    void shift();
    void move();
    void repair();
}
```

Classes

- A class is a set of objects that share common structure and a common behavior
- A class is an abstraction in that it:
  - Emphasizes relevant characteristics
  - Suppresses other characteristics

3.Inheritance

- allows objects to be built from other objects
- Provides programming by extension as opposed to programming by reinvention
- Allows classes to share and reuse behaviors and attributes

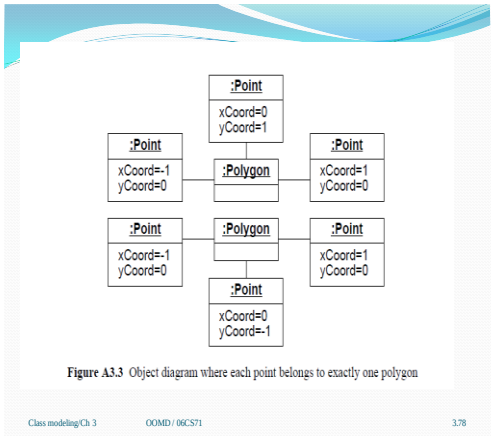


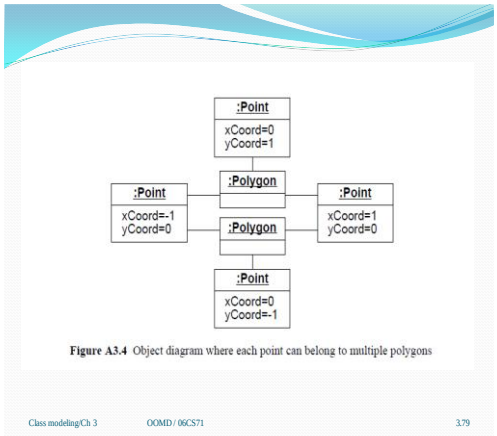
CSE

4.Polymorphism

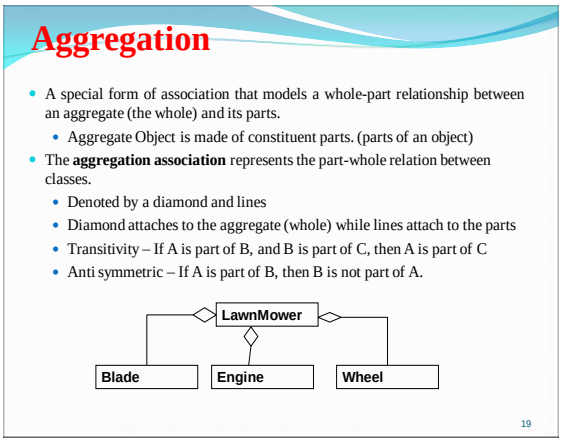
- it means objects that can take on or assume many different forms
- The same operation may behave differently on different classes
- Allows user to write generic, reusable code more easily

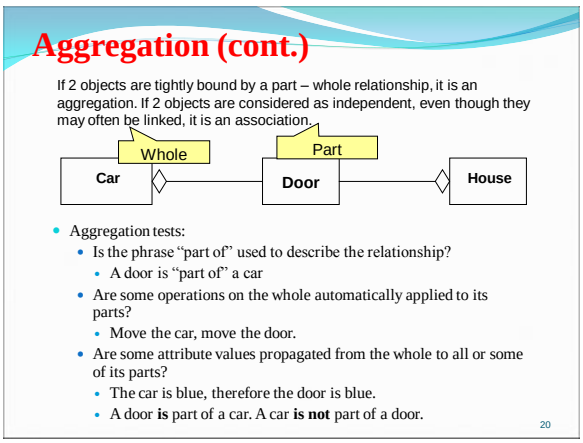
B) Using the class diagram below, prepare an object diagram for the two triangles with a common side under the following condition: a) A point belongs to exactly one polygon b) A point belongs to one or more polygon. (4M)

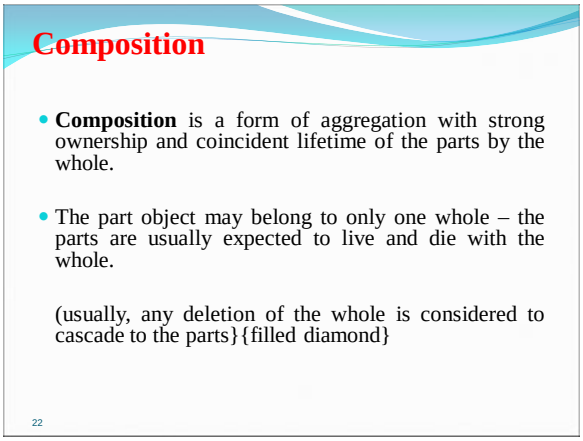




2. A) Explain aggregation and composition with suitable example. (6M).

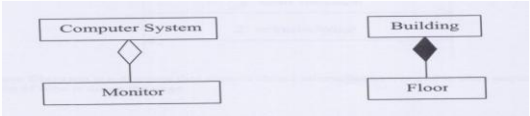






Aggregation and Composition

- **Aggregation** is a special form of association that specifies a whole-part relationship between the aggregate (the whole) and a component (the part); aggregation is the part-of relationship.
- **Composition** is a form of aggregation with strong ownership and coincident lifetime of the parts by the whole; the part object may belong to only one whole – the parts are usually expected to live and die with the whole.



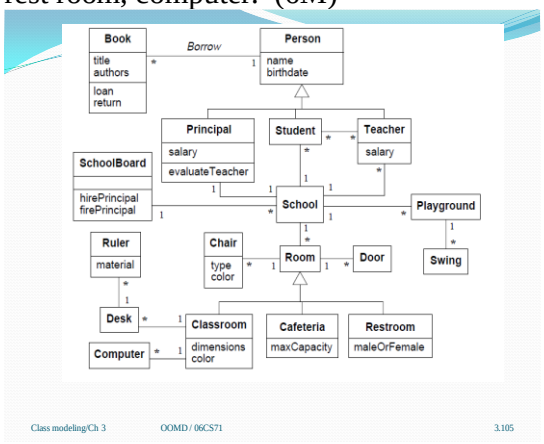
Composition vs. Aggregation

Aggregation	Composition
Part can be shared by several wholes category 0..4 document	Part is always a part of a single whole Window Frame
Parts can live independently (i.e., whole cardinality can be 0..*)	Parts exist only as part of the whole. When the wall is destroyed, they are destroyed
Whole is not solely responsible for the object	Whole is responsible and should create/destroy the objects

Intro | Classes | **Relations** | Generalization | Objects | Guidelines 40

B) Draw class diagram for the following: a) Programmer uses computer language on projects. b) Worker is a butcher or baker or candlestick maker. (4M)

3. A) Prepare a class diagram for the group of classes given below. Add at least three relationships (associations, generalization). Use association names where needed and show multiplicity. School, playground, principle, book, student, teacher, cafeteria, class room, rest room, computer. (6M)

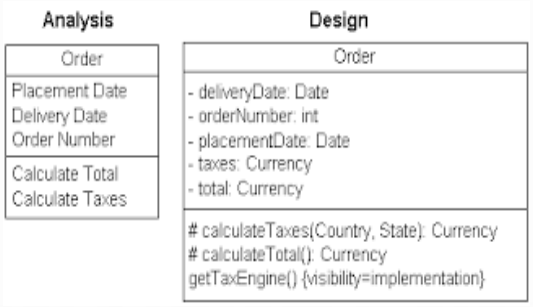


B) What is visibility? Explain with suitable example. (4M)

Visibility

- Visibility refers to the ability of a method to reference a feature from another class and has the possible values of *public*, *protected* & *private*.
- ***Public***—Visible anywhere that the class in which it appears is visible; denoted by **+**.
- ***Package***—Visible anywhere in the package containing the class in which it appears; denoted by **~**.
- ***Protected***—Visible in the class in which it appears and all its sub-classes; denoted by **#.**
- ***Private***—Visible only in the class in which it appears; denoted by **-**
- Restricting visibility is the same as restricting accessibility.

Visibility Example



4. Explain the concept of workaround and its approaches with suitable example. (10M)

Workarounds:

• A method for overcoming a problem or limitation in a program or system.

•To deal with implementation issues with multiple inheritance – workaround is used.

•3 Approaches:

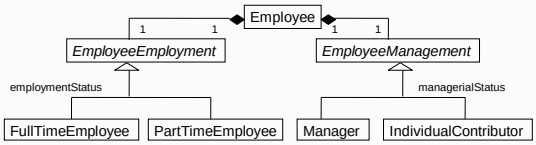
1. Delegation using composition of parts.

- Delegation: Implementation mechanism by which an object forwards an operation to another object for execution.
- Recast a super class as composition in which each constituent part replaces a generalization.
- Need not create the various combinations as explicit classes. All combinations of subclasses from the different generalizations are possible.

Eg: EmployeeEmployment becomes a superclass of FullTimeEmployee and PartTimeEmployee. EmployeeManagement becomes a superclass of Manager and individualContributor.

Employee has been modeled as composition of EmployeeEmployment and EmployeeManagement.

An operation sent to Employee object will be redirected to the EmployeeEmployment and EmployeeManagement part by the Employee class.

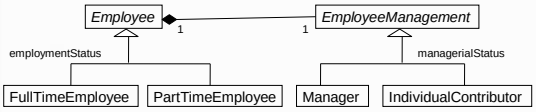


Workaround for multiple inheritance - delegation

2. Inherit the most important class and delegate the rest:

• Preserves identity and inheritance across the most important generalization.

•Degrade the remaining generalizations to composition and delegate their operations.



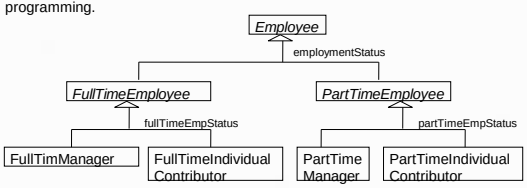
Workaround for multiple inheritance – inheritance and delegation

3. Nested Generalization:

• Factor on one generalization first, then the other. This multiplies out all possible combinations.

•Eg: Under FullTimeEmployee and PartTimeEmployee, add two subclasses for managers and individual contributors.

•This preserves inheritance but duplicates code and violates the spirit of OO programming.



Workaround for multiple inheritance – nested generalization

Issues to consider when selecting the best workaround:

1. **Superclasses of equal importance:** If a subclass has several superclasses, all of equal importance, it may be best to use Delegation using composition of parts (1st Approach)
2. **Dominant superclass:** If one superclass clearly dominates and the other are less important, preserve inheritance by using : Inherit the most important class and delegate the rest, or nested generalization (2nd or 3rd Approach)
3. **Few subclasses:** If the number of combinations is small, consider nested generalization (3rd Approach). If the number of combinations are large, avoid it.
4. **Sequencing generalization sets:** If nested generalization is used, then factor on the most important criterion first, the next most important second and so forth.
5. **Large quantities of code:** Avoid nested generalization (3rd Approach) as it will duplicate large quantity of code.
6. **Identity:** Strict identity has to be considered as important. Only nested generalization (3rd Approach) preserves it.

5. Explain state diagram and write state model for a telephonic line with activities. (10M)

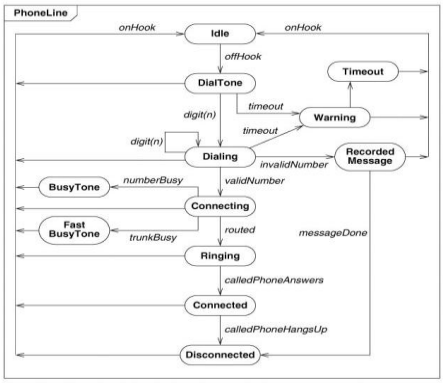


Figure 5.8 State diagram for a telephone line. A state diagram specifies the state sequences caused by event sequences.

6. A simple digital watch has a display and two buttons to set it, the A button, and the B button. The watch has two modes of operation, display time, set time in the display time mode, the watch displays hours and minutes, separated by a flashing colon. The set time mode has two sub modes, set hours, set minutes. The button A selects modes. Each time it is pressed, the mode advances in the sequence: display, set hours, set minutes, display etc. Within the sub modes the button B advances the hours or minutes once each time it is pressed. Buttons must be released before they can generate another event. Prepare a state diagram of the watch. (6M)

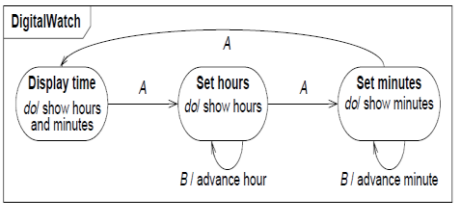


Figure A5.2 State diagram for a simple digital watch

In Figure A5.2 the event A refers to pressing the A button. In this diagram, releasing the button is unimportant and is not shown (although you must obviously release the button before you can press it again). Note that a new button event cannot be generated while any button is pressed. You can consider this a constraint on the input events themselves and need not show it in the state diagram (although it would not be wrong to do so).

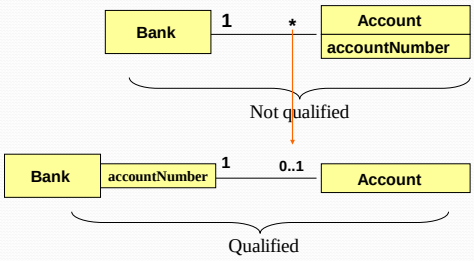
B) Explain qualified association with suitable example. (4M)

Qualified Association

- A *qualified association* is an association in which an attribute called the *qualifier* disambiguates the objects for a "many" association end.
- It is possible to define qualifiers for one-to many and many-to-many associations.
- A qualifier selects among the target objects, reducing the effective multiplicity, from "many" to "one."

Contd..

The notation for qualifier→ small box on the end of the association line near the source class. The source class and qualifier yields target class.



Qualified Association

- Both below models are acceptable but the qualified model adds information.
- But the qualified model adds multiplicity constraint, that the combination of a bank and an account number yields at most one account.
- Figure 3.22 illustrates the most common use of a qualifier- for associations with one to-many multiplicity. A bank services multiple accounts. An account belongs to a single bank. Within the context of a bank, the account number specifies a unique account. *Bank* and *Account* are classes and *accountNumber* is the qualifier. Qualification reduces the effective multiplicity of this association from one-to-many to one-to-one.

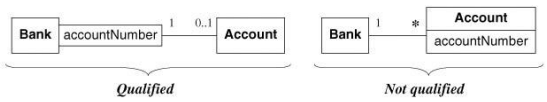


Figure 3.22 Qualified association. Qualification increases the precision of a model.

Association: ordering

- On a 'many" association end, sometimes, it is required that objects have an explicit order.
- For example, Figure 3.15 shows a workstation screen containing a number of overlapping windows. Each window on a screen occurs at most once. The windows have an explicit order, so only the topmost window is visible at any point on the screen.
- In this case the ordering is an important part of the association
- Example:

Figure 3.15 Ordering the objects for an association end. Ordering sometimes occurs for "many" multiplicity.

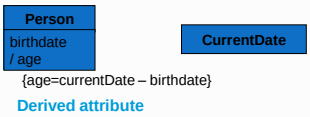
43

7. Explain the following with the help of UML: a) Derived attributes b) Ordering c) Packages d) Enumeration e) Multiplicity (object and attributes). (10M)

a) Derived Attributes

Derived data

- Classes, attributes and associations may be derived from others.
- The notation for a derived element is a slash (/) in front of the element name.
- The constraint that determines the derivation should also be shown.



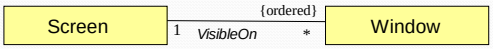
48

b) Ordering

Ordering

Ordering occurs for "many" multiplicity

Often the objects on a many association end have no explicit order and can regard them as a set. However, the objects have explicit order some times.

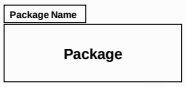


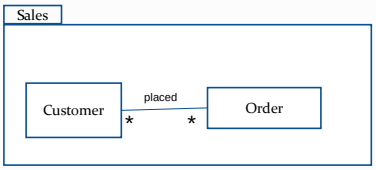
- Example: A workstation screen containing a number of overlapping windows. Each window on a screen occurs at most once.

c) Packages

Packages

- A package is a group of elements (classes, association, generalization, and lesser packages) with a common theme.
- A package partitions a model making it easier to understand and manage. Large applications may require several tiers of packages.
- Notation for package is a box with a tab.





d) Enumeration

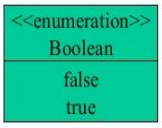
Enumeration

Data type – description of values.

Enumeration is a data type that has finite set of values.

- To define data type, called enumeration type, we need 2 things:
  - A name for the data type
  - A set of values for the data type
- enum {FALSE, TRUE};
- enum rank {TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE};
- enum colors {BLACK, BLUE, GREEN, CYAN, RED};
- The values are written in all caps because they are constants

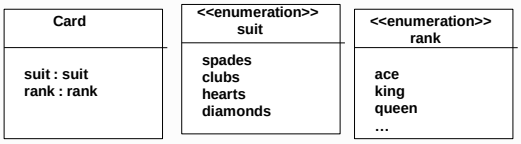
Enumeration



An ***enumeration*** is a user-defined data type that consists of a name and an ordered list of enumeration literals.

Draw class diagram with enumeration data type

```
Class card
{
    enum suit={spades,clubs,hearts,diamonds};
    enum rank={ace,king,queen};
}
```



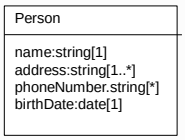
e) Multiplicity

Multiplicity

Multiplicity is a collection on the cardinality of a set, also applied to attributes (database application).

Multiplicity of an attribute specifies the number of possible values for each instantiation of an attribute. i.e., whether an attribute is mandatory ([1]) or an optional value ([0..1] or * i.e., null value for database attributes) .

Multiplicity also indicates whether an attribute is single valued or can be a collection.



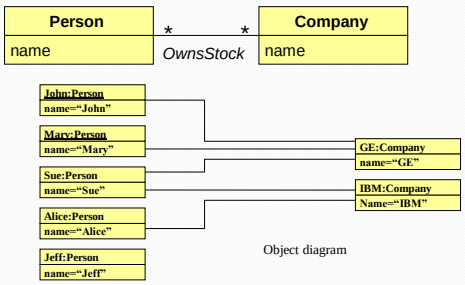
Multiplicity

- Specifies the number of instances of one class that may relate to a single instance of an associated class
- Multiplicity constrains the number of related objects
- UML diagrams explicitly list multiplicity at the end of association lines.

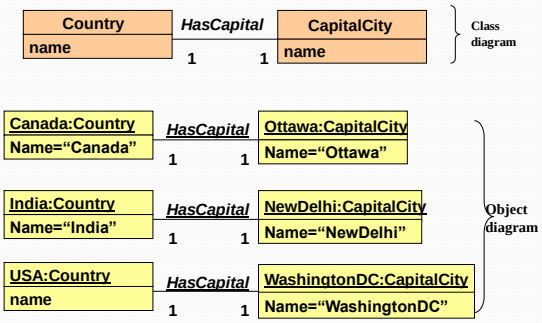
Multiplicity – UML notations

- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One or more - 1..*
- Exact Number - e.g. 3..4 or 6
- Or a complex relationship – e.g. 0..1, 3..4, 6..* would mean any number of objects other than 2 or 5

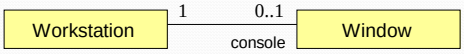
Multiplicity:Many-to-Many



Multiplicity: one-to-one



Multiplicity: Zero-or-one



- A workstation may have one of its windows designated as the console to receive general error messages.
- It is also possible that no console exists.
- Multiplicity v/s cardinality: multiplicity is a constraint on the size of a collection ; cardinality is the count of elements that are actually in a collection. therefore multiplicity is a constraint on a cardinality