OOP Concepts Handouts edited

Object Oriented Programming (COMSATS University Islamabad)

CS304-Handouts

Lecture No.01

01.1. Introduction

Course Objective:
Objective of this course is to make students familiar with the concepts of object oriented programming. These concepts will be reinforced by their implementation in C++.

Course Contents:
The main topics that we will study in the 45 lectures of this course are given below,

- Object Orientation
- Objects and Classes
- Overloading
- Inheritance
- Polymorphism
- Generic Programming
- Exception Handling
- Introduction to Design Patterns

Recommended Text Book:

C++ How to Program (Deitel & Deitel **)**

Reference Books:

1. **Object-Oriented Software Engineering**
   By Jacobson, Christerson, Jonsson, Overgaard
   (For object oriented programming introductory concepts)
2. **The C++ Programming Language**
   By Bjarne Stroustrup
   (For better c++ understanding)

Object-Orientation (OO)

What is Object-Orientation?

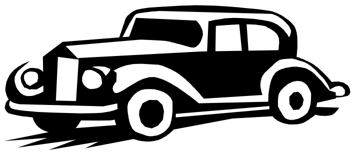
It is a technique in which we visualize our programming problems in the form of objects and their interactions as happens in real life.

Examples:
We have different objects around us in our real life that interact with each other to perform different operations for example,

A Person

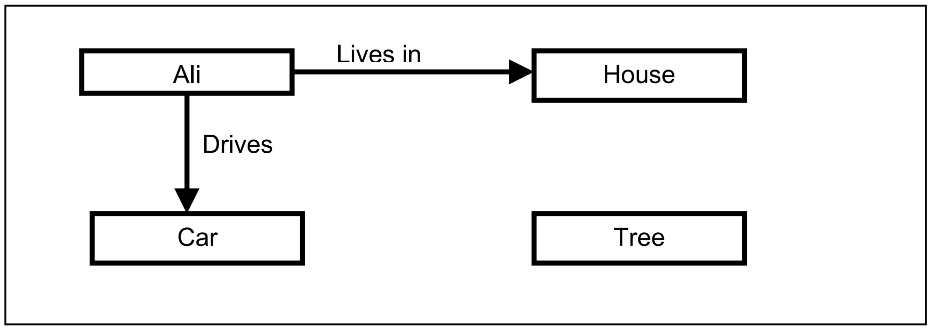
A House

A Tree


A Car

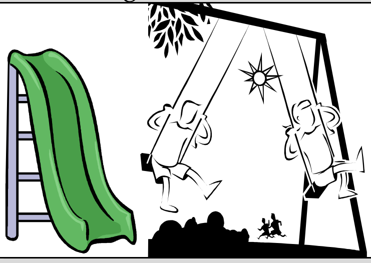
Different Objects

These objects interact with each other to perform different operations,



Take another example of a **School**; the objects in a school are **student**, **teacher**, **books, pen** ,**school bag**, **classroom, parents, playground and so on… ,**

Objects in a School

Teacher	Student	School Bag
		
Book	Pen	Playground
Parents	Classroom	Library

Suppose we want to develop a fee collection system for a school for this we will need to find out related objects and their interactions as happens in real life.
In this way we can say that **object orientation** makes it easier for us to solve our real world problems by thinking solution of the problem in terms of real world objects.

So we can say that in our daily life everything can be taken as an object that behaves in a certain way and has certain attributes.

In object orientation we move our concentration to objects in contrast to procedural paradigm in which we simply write our code in functions and call them in our main program.

01.2.What is a Model?

A model is an abstraction of something real or conceptual.
We need models to understand an aspect of reality.

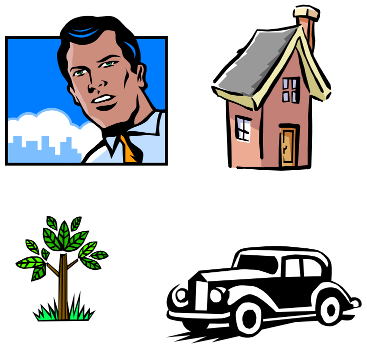
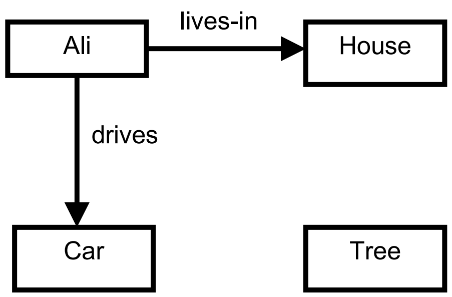
Model Examples

Highway maps
Architectural models
Mechanical models

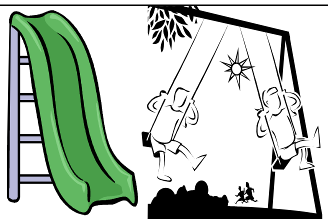
01.3.OO Models:

In the context of programming models are used to understand the problem before starting developing it.
We make Object Oriented models showing several interacting objects to understand a system given to us for implementation.

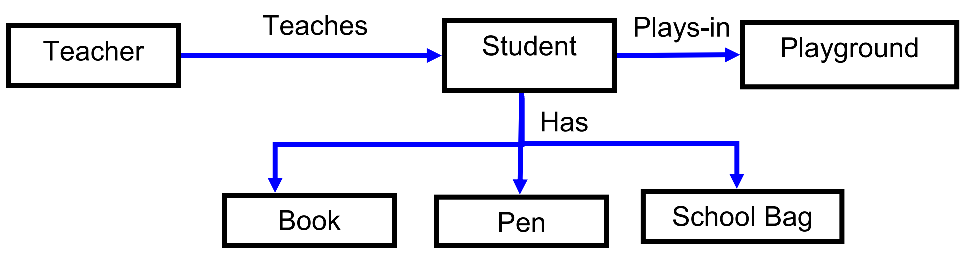
Example 1– Object Oriented Model

	
Objects Ali, Car, House, Tree	**Interactions** Ali lives in the house Ali drives the car

Example 2– Object Oriented Model (A School Model)

A School Model



Objects	**Interactions**
Teacher, Student, School Bag, Pen, Book Playground	Teacher teaches Student. Student has School Bag, Book and Pen

01.4.Object-Orientation - Advantages

As Object Oriented Models map directly to reality as we have seen in examples above therefore,

We can easily **develop** an object oriented model for a problem.
Everyone can easily **understand** an object oriented model.
We can easily implement an object oriented model for a problem using any object oriented language like c++ using its features[1] like classes, inheritance, virtual functions and so on…

01.5.What is an Object?

An object is,

1. Something tangible (Ali, School, House, Car).
2. Something conceptual (that can be apprehended intellectually for example time, date and so on…).

An object has,

1. State (attributes)
2. Well-defined behavior (operations)

[1] We will study these features in detail in this course

3. Unique identity

01.6.Tangible and Intangible Objects

Examples of Tangible Objects:

Ali is a tangible object, having some characteristics (attributes) and behavior as given below,

Ali	
Characteristics (attributes)	**Behaviour (operations)**
Name Age	Walks Eats

We will identify Ali using his name.

Car is also a tangible object having some characteristics (attributes) and behavior given below,

Car	
State (attributes)	**Behavior (operations)**
Color Model	Accelerate Start Car Change Gear

We can identify Car using its registration number

Examples of Intangible Objects (also called as conceptual objects):

Time is an intangible (conceptual) object

Time	
State (attributes)	**Behavior (operations)**
Hours Seconds Minutes	Set/Get Hours Set/Get Seconds Set/Get Minutes

We will assign our own generated unique ID in the model for Time object

Date is also an intangible (conceptual) object

State (attributes)	**Behavior (operations)**
Year Day Month	Set/Get Year Set/Get Day Set/Get Month

We will assign our own generated unique ID in the model for Date object.

01.7.Summary:

- Model is the abstraction of some real word scenario. It helps us to understand that scenario.
- Object oriented model of any scenario (problem) describes that scenario (problem) in the form of interacting objects.
- We use Object Orientation because it helps us in mapping real world problem in a programming language.
- Object Orientation is achieved using objects and their relationships.
- Properties of an object are described using its *data members* and behavior of an object is described using its *functions.*
- Objects may be tangible (physical) or intangible (also called conceptual or virtual).
- Generally when we have given a certain problem description, **nouns** in that problem description are *candidates* for becoming objects of our system.
- There may be more than one aspects of an object
- It is not necessary that every object has a specific role in implementation of a problem there may be some objects without any role, like school parking in our school.
- It is easier to develop programs using Object Oriented Programming because it is closer to real life.

Lecture No.02

Lecture Contents

1. Information Hiding
2. Encapsulation
3. Interface
4. Implementation
5. Separation of Interface & Implementation
6. Messages

02.1.Information Hiding:

Information hiding is one of the most important principles of OOP inspired from real life which says that all information should not be accessible to all persons. Private information should only be accessible to its owner.
By Information Hiding we mean *"Showing only those details to the outside world which are necessary for the outside world and hiding all other details from the outside world."*

Real Life Examples of Information Hiding

1. Ali's name and other personal information is stored in his brain we can't access this information directly. For getting this information we need to ask Ali about it and it will be up to Ali how much details he would like to share with us.

2. An email server may have account information of millions of people but it will share only our account information with us if we request it to send anyone else accounts information our request will be refused.

3. A phone SIM card may store several phone numbers but we can't read the numbers directly from the SIM card rather **phone-set** reads this information for us and if the owner of this phone has not allowed others to see the numbers saved in this phone we will not be able to see those phone numbers using phone.

In object oriented programming approach we have objects with their attributes and behaviors that are hidden from other classes, so we can say that object oriented programming *follows the principle of information hiding*.

In the perspective of **Object Oriented Programming** Information Hiding is,

"Hiding the object details (state and behavior) from the users"

Here by users we mean **"an object"** of another class that is calling functions of this class using the reference of this class object or it may be some other program in which we are using this class.

Information Hiding is achieved in Object Oriented Programming using the following principles,

- All information related to an object is stored within the object
- It is hidden from the outside world
- It can only be manipulated by the object itself

Advantages of Information Hiding

Following are two major advantages of information hiding,

It simplifies our Object Oriented Model:

As we saw earlier that our object oriented model only had objects and their interactions hiding implementation details so it makes it easier for everyone to understand our object oriented model.

It is a barrier against change propagation

As implementation of functions is limited to our class and we have only given the name of functions to user along with description of parameters so if we change implementation of function it doesn't affect the object oriented model.

We can achieve information hiding using **Encapsulation** and **Abstraction**, so we see these two concepts in detail now,

02.2. Encapsulation

Encapsulation means *"we have enclosed all the characteristics of an object in the object itself"*
Encapsulation and information hiding are much related concepts (information hiding is achieved using Encapsulation)
We have seen in previous lecture that object characteristics include data members and behavior of the object in the form of functions.

So we can say that Data and Behavior are tightly coupled inside an object and both the information structure and implementation details of its operations are hidden from the outer world.

Examples of Encapsulation

Consider the same example of object Ali of previous lecture we described it as follows,

Ali
Characteristics (attributes) • Name • Age
Behavior (operations) • Walks • Eats

You can see that Ali stores his personal information in itself and its behavior is also implemented in it.
Now it is up to object Ali whether he wants to share that information with outside world or not. Same thing stands for its behavior if some other object in real life wants to use his behavior of walking it can not use it without the permission of Ali.

So we say that attributes and behavior of Ali are encapsulated in it.

Any other object don't know about these things unless Ali share this information with that object through an interface,

Same concept also applies to phone which has some data and behavior of showing that data to user we can only access the information stored in the phone if phone interface allow us to do so.

Advantages of Encapsulation
The following are the main advantages of Encapsulation,

a. Simplicity and clarity
As all data and functions are stored in the objects so there is no data or function around in program that is not part of any object and is this way it becomes very easy to understand the purpose of each data member and function in an object.

b. Low complexity
As data members and functions are hidden in objects and each object has a specific behavior so there is less complexity in code there will be no such situations that a functions is using some other function and that functions is using some other function.

c. Better understanding
Everyone will be able to understand whole scenario by simple looking into object diagrams without any issue as each object has specific role and specific relation with other objects.

02.3.Interface

Interface is a set of functions of an object that he wants to expose to other objects.

As we discussed previously that data and behavior of each object is hidden in that object it self so we have to use the concept of interface of the object to expose its behavior to outer word objects.

- Different objects may need different functions of an object so interface of an object may be different for different objects.
- Interfaces are necessary for object communication. Each object provides interface/s (operations) to other objects through these interfaces other objects communicate with this object.

Example – Interface of a Car

- Steer Wheels
- Accelerate
- Change Gear
- Apply Brakes
- Turn Lights On/Off

Example – Interface of a Phone

- Input Number
- Place Call
- Disconnect Call
- Add number to address book
- Remove number
- Update number

02.4. Implementation

It is actual implementation of the behavior of the object in any Object Oriented language.

It has two parts,

- Internal data structures to hold an object state that will be hidden from us it will store values for an object data members.
- Functionality in the form of member functions to provide required behavior.

Examples of Implementation

a. **Gear Box in car system**
Consider object Gear Box in car system it has a certain structure and functionality. When this object will be implemented it will have two things,
- Physical structure of the gear box
- Functionality implemented in this structure to change gear.

Both these things are part of implementation.

So it has,

- **Data Structure** in the form of Mechanical structure of gear box
- **Functionality** mechanism to change gear

b. Address Book in a Phone

Similarly take the example of contact details saved in the SIM of a phone,

In that case we can say physical structure of SIM card as **Data Structure**
And Read/write operations provided by the phone as **Functionality**.

02.5.Separation of Interface & Implementation

As discussed earlier we only show interface of an object to outside world and hide actual implementation from outside world. The benefit of using this approach is that our object interface to outside word becomes independent from inside implementation of that interface.

This is achieved through the concepts of encapsulation and information hiding.

Real Life example of separation of interface and implementations

- ➢ Driver has a standard interface to drive a car and using that interface he drive can drive any car regardless of its model or type whatever engine type it has or whatever type of fuel it is using.

02.6.Messages

Objects communicate through messages they send messages (stimuli) by invoking appropriate operations on the target object. The number and kind of messages that can be sent to an object depends upon its interface

Examples – Messages

A Person sends message (stimulus) “stop” to a Car by applying brakes

A Person sends message “place call” to a Phone by pressing appropriate button

02.7.Summary

- Information hiding is achieved through encapsulation.
- Encapsulation and Information Hiding are related to each other.
- Interface of an object provides us the list of available functions.
- An object may have more than one interface.
- Interface and implementation are separated from each other to achieve Information Hiding.
- Objects communicate with each other using messages.

Useful Links:

http://www.alice.org/

A Graphical Programming Environment to teach Computer Programming.

Lecture No.03

Lecture Contents:

- Abstraction
- Classes
- Inheritance
- Major benefits of inheritance (Reuse)

03.1. Abstraction

Real life objects have a lot of attributes and many kind of behaviors but most of the time we are interested in only that part of the objects that is related to the problem we are currently going to solve, for example in implementing a school system we don't need to take care of the personnel life of a student or a teacher as it will not effect our system in any way so we will see these objects in the perspective of school system and will ignore their other characteristics, this concept is called "*Abstraction*". Abstraction is a way to cope with complexity and it is used to simplify things.

Principle of abstraction:

"Capture only those details about an object that are relevant to current perspective"

Abstraction Example:

Suppose we want to implement abstraction for the following statement,

"Ali is a PhD student and teaches BS students"

Here object Ali has two ***perspectives*** *one is his* ***student perspective*** *and second is his* ***teacher perspective.***

We can sum up Ali's attributes as follows,

Name
Age
Student Roll No
Year of Study
CGPA
Employee ID
Designation
Salary

As you can see out of all these listed attributes some belong to Ali's student perspective(Roll No, CGPA, Year of study) and some belong to Ali's teacher perspective(Employee ID, Designation, Salary).

Similarly we can sum up Ali's behavior as follows,

Study
DevelopExam

GiveExam
TakeExam
PlaySports
Eat
DeliverLecture
Walk

As was the case with attributes of object Ali, its behavior can also be divided in Ali's student perspective as well as Ali's teacher perspective.

Student's Perspective

Attributes:

- **Name**	- Employee ID
- **Student Roll No**	- Designation
- **Year of Study**	- Salary
- **CGPA**	- Age

Behaviour:

- **Study**	- DevelopExam
- **GiveExam**	- TakeExam
- **PlaySports**	- Eat
- DeliverLecture	- Walk

Teacher's Perspective

Attributes:

- **Name**	- **Employee ID**
- Student Roll No	- **Designation**
- Year of Study	- **Salary**
- CGPA	- Age

Behaviour:

- Study	- **DevelopExam**
- GiveExam	- **TakeExam**
- PlaySports	- Eat
- **DeliverLecture**	- Walk

A cat can be viewed with different perspectives

Ordinary Perspective A pet animal with Four Legs A Tail	Surgeon's Perspective A being with A Skeleton Heart

Two Ears Sharp Teeth	Kidney Stomach

A car can be viewed with different perspectives

Driver's View	Engineer's View

Abstraction – Advantages

Abstraction has following major advantages,

1. It helps us understanding and solving a problem using object oriented approach as it hides extra irrelevant details of objects.

2. Focusing on single perspective of an object provides us freedom to change implementation for other aspects of for an object later.

Similar to Encapsulation Abstraction is also used for achieving information hiding as we show only relevant details to related objects, and hide other details.

03.2.Classes

In OOP we create a general sketch for each kind of objects and then we create different instances using this sketch we call this sketch or prototype or map as "class".
All objects of same kind exhibit identical characteristics (information structure and behavior) however they have data of their own.

Class –Example 1

Consider the objects given below,
- Ali studies mathematics
- Anam studies physics
- Sohail studies chemistry

Each one is a Student so we say these objects are *instances* of the Student class.

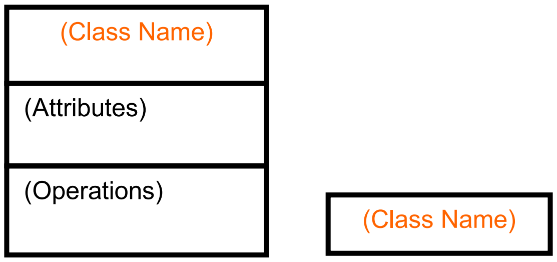
Class –Example 2

Consider the objects given below,

- Ahsan teaches mathematics
- Aamir teaches computer science
- Atif teaches physics

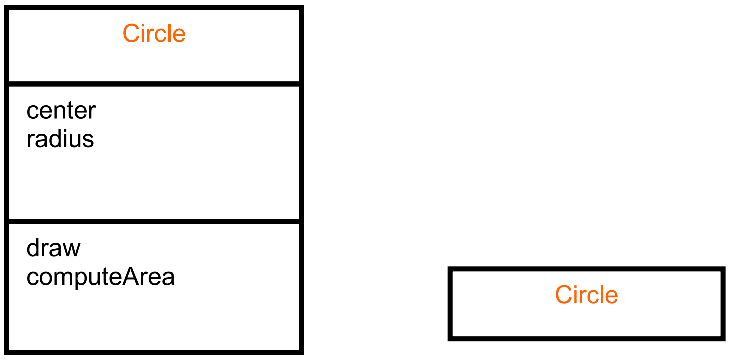
Each one is a teacher so we say these objects are *instances* of the Teacher class

Class Representation:
we can represent a class using a rectangle as follows,



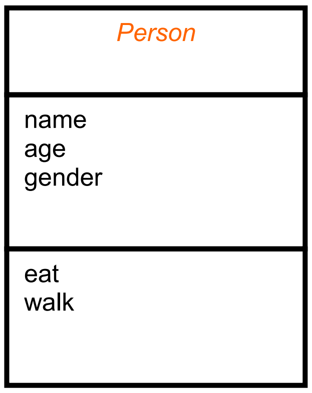
Normal Form **Suppressed Form**

Class Example: Circle

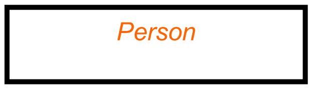


Normal Form **Suppressed Form**

Class Example: Person



Normal Form



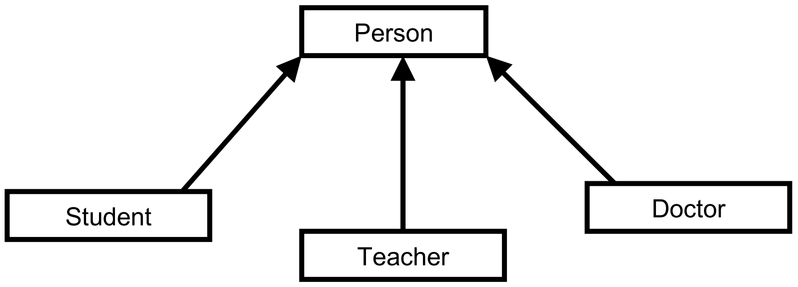
Suppressed Form

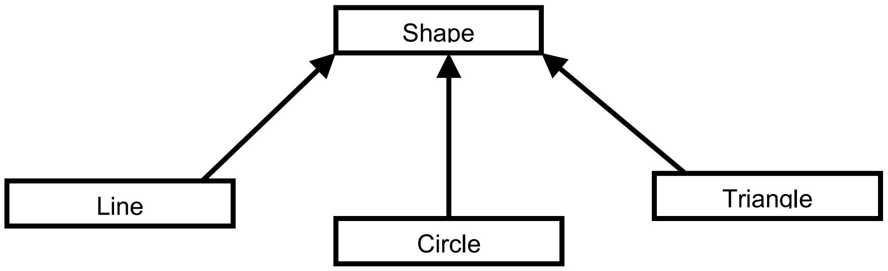
03.3.Inheritance

A child inherits characteristics of its parents, besides inherited characteristics, a child may have its own unique characteristics

Inheritance in Classes

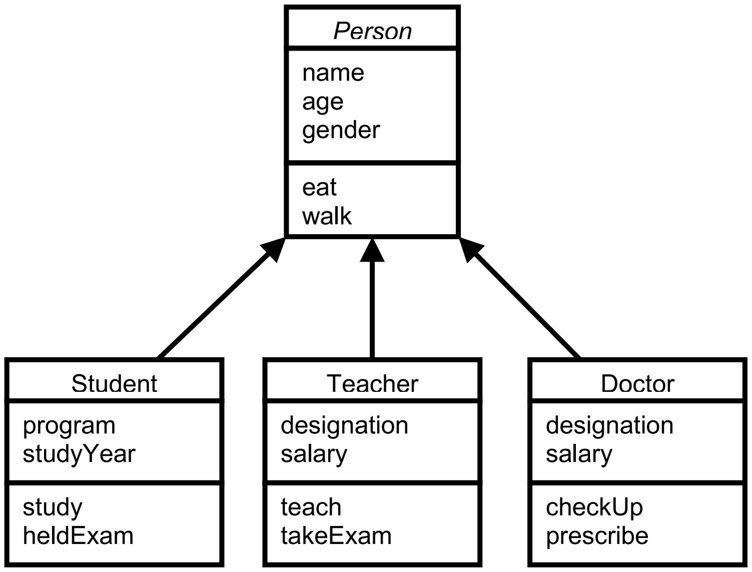
If a class B inherits from class A then it contains all the characteristics (information structure and behaviour) of class A
The parent class is called *base* class and the child class is called *derived* class
Besides inherited characteristics, derived class may have its own unique characteristics



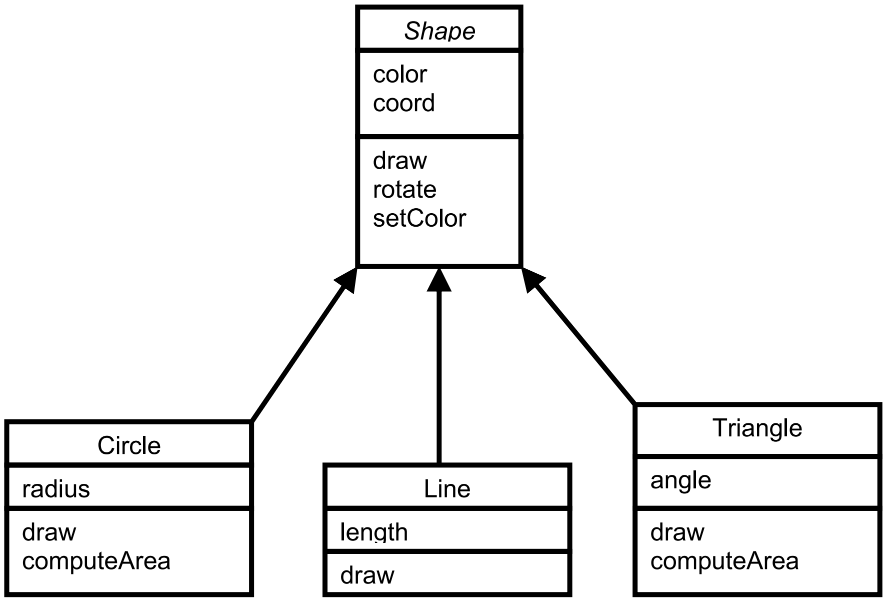


Inheritance – "IS A" or "IS A KIND OF" Relationship

Each derived class is a kind of its base class



Here,
Student IS A Person
Teacher IS A Person
Doctor IS A Person



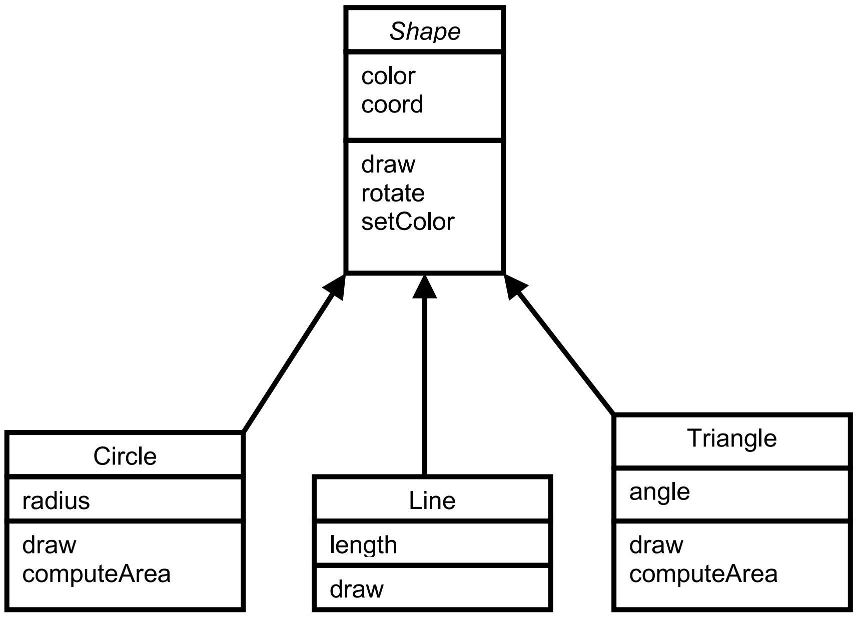
Here,
Circle IS A Shape
Line IS A Shape
Triangle IS A Shape

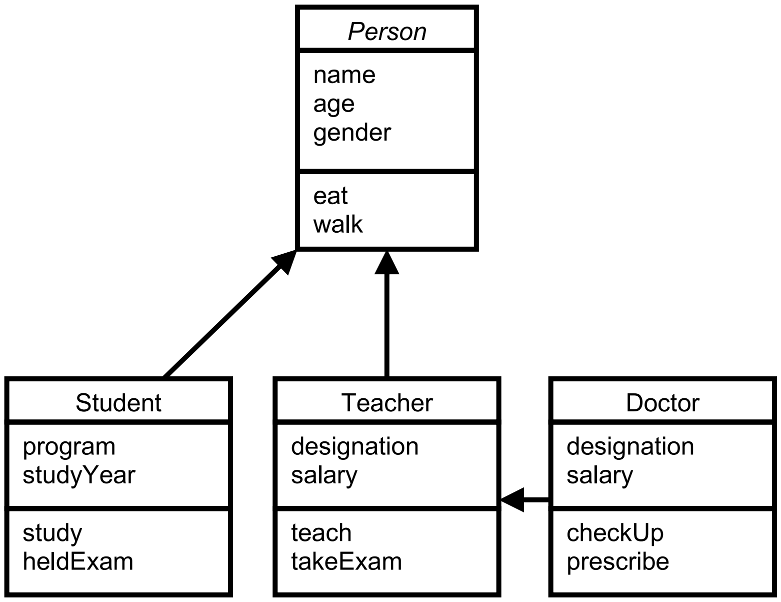
Inheritance – Advantages

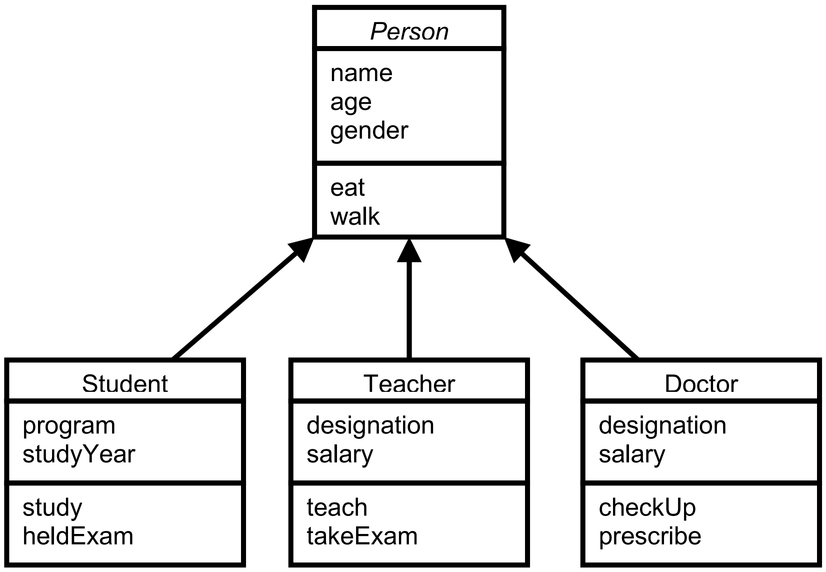
1. Reuse
2. Less redundancy
3. Increased maintainability

Reuse with Inheritance

Main purpose of inheritance is reuse, we can easily add new classes by inheriting from existing classes.
Select an existing class closer to the desired functionality, create a new class and inherit it from the selected class, add to and/or modify the inherited functionality




Person
name
age
gender
eat
walk
Student
program
studyYear
study
heldExam
Teacher
designation
salary
teach
takeExam
Doctor
designation
salary
checkUp
prescribe


Person
name
age
gender
eat
walk
Student
program
studyYear
study
heldExam
Teacher
designation
salary
teach
takeExam
Doctor
designation
salary
checkUp
prescribe

Lecture No.04

Lecture Contents

- Generalization
- Sub typing (extension)
- Specialization (restriction)
- Overriding
- Abstract classes
- Concrete classes

Recap – Inheritance

- Derived class inherits all the characteristics of the base class
- Besides inherited characteristics, derived class may have its own unique characteristics
- Major benefit of inheritance is reuse

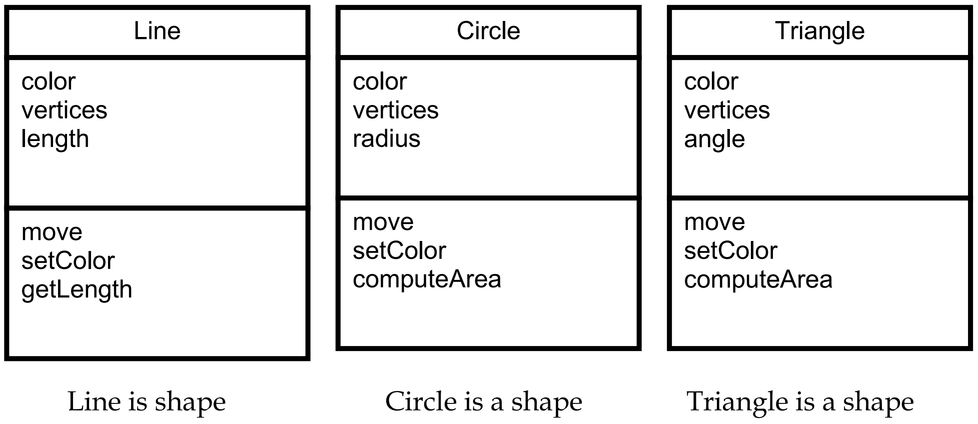
04.1.Concepts Related with Inheritance

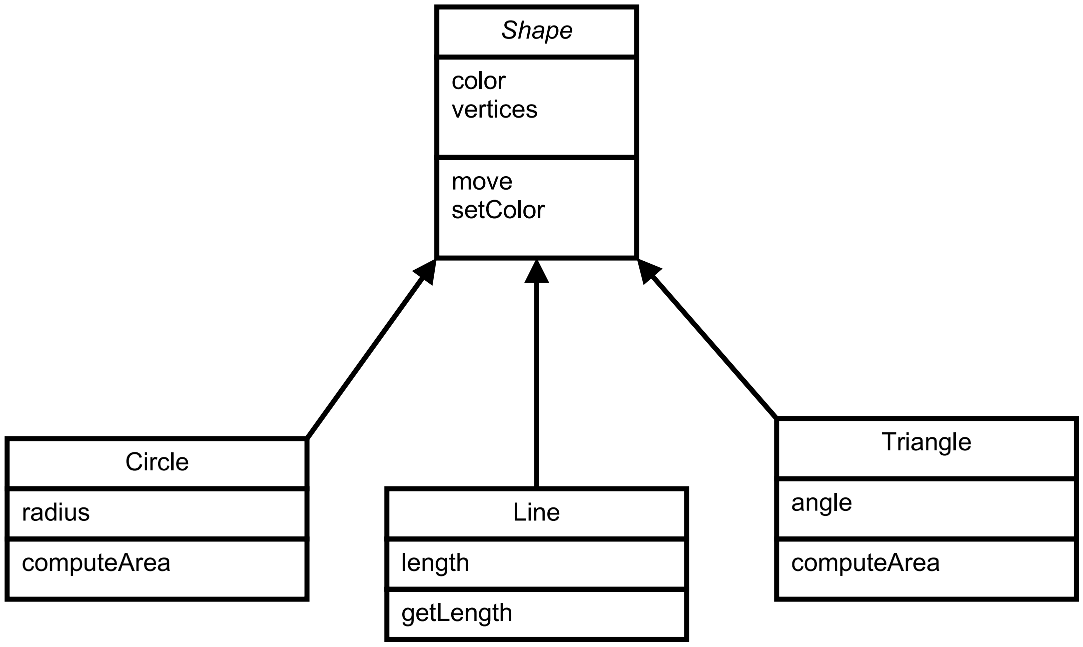
- Generalization
- Subtyping (extension)
- Specialization (restriction)

04.2.Generalization

In OO models, some classes may have common characteristics.
We extract these features into a new class and inherit original classes from this new class. There are many objects with common characteristics in object model. The common characteristics (attributes and behaviour) of all these objects are combined in a single general class. Base class encapsulates the idea of commonality of derived classes. Base class is general class representing common behaviour of all derived classes.
This concept is known as Generalization.
It reduces the redundancy and gives us reusability, using generalization our solution becomes less complex.
In generalization there should be "Is a Kind of Relationship" (also called "Is A relationship") between base and child classes.

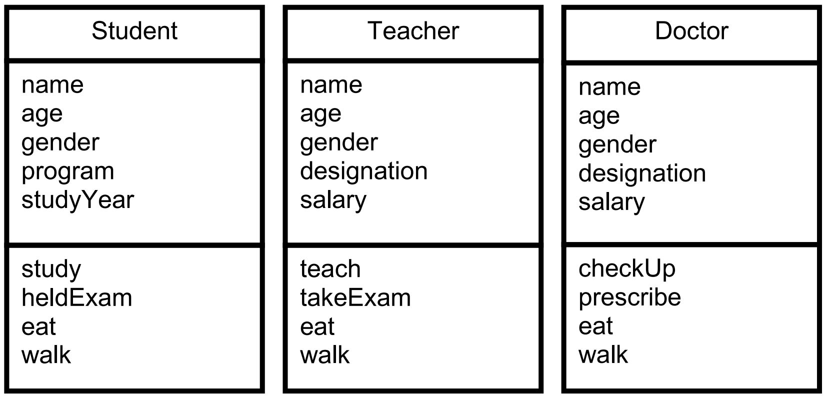
Example: Line, Circle and Triangle

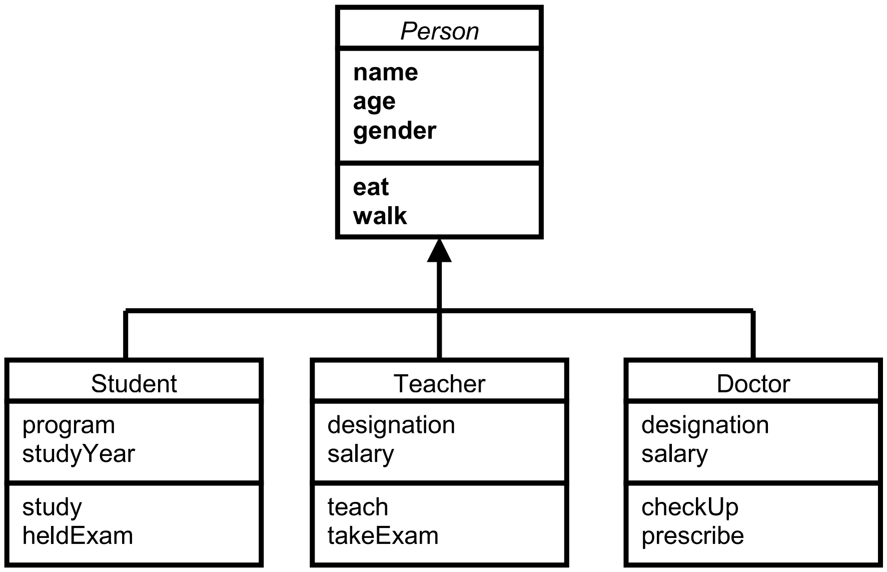




Common attributes
Color vertices
Common behaviour
Set Color, Move

Example: Student Doctor and Teacher





Common attributes,
Name, age, gender

Common behaviour
Eat, Walk

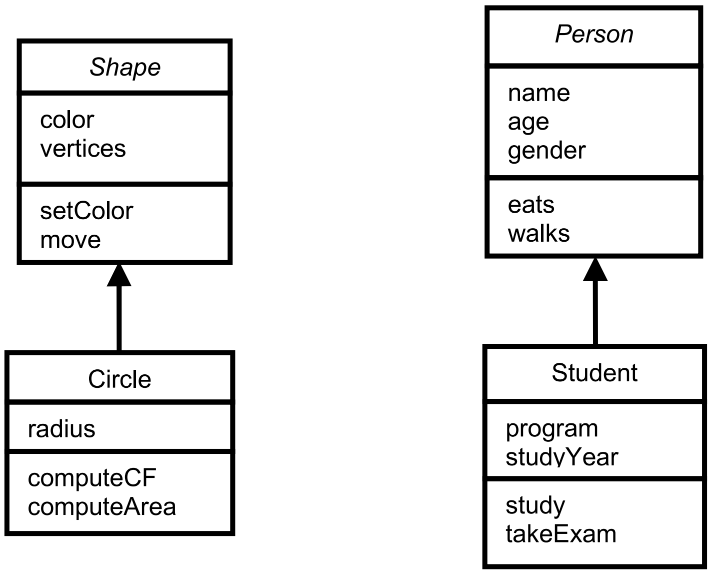
Sub-typing & Specialization

We want to add a new class to an existing model
We have developed an existing class hierarchy
Find an existing class that already implements some of the desired state and behaviour
Inherit the new class from this class and add unique behaviour to the new class

04.3.Sub-typing (Extension)

Sub-typing means that derived class is behaviourally compatible with the base class
Derived class has all the characteristics of base class plus some extra characteristics
Behaviourally compatible means that base class can be replaced by the derived class

Sub-typing (Extension) - Example



Circle is extending the behaviour of shape, it is extending attributes of shape by adding radius similarly it is extending behaviour of shape by adding compute Circumference and compute Area.

Student has two extra attributes program and studyYear
Similarly it has extended behaviour by adding study and takeExam.

Subtyping and generalization are related concepts, Subtyping (extension) and generalization is a way to look same thing in two ways.
Sub typing is looking at things from Top to bottom whereas in generalization we look at things from bottom to top.

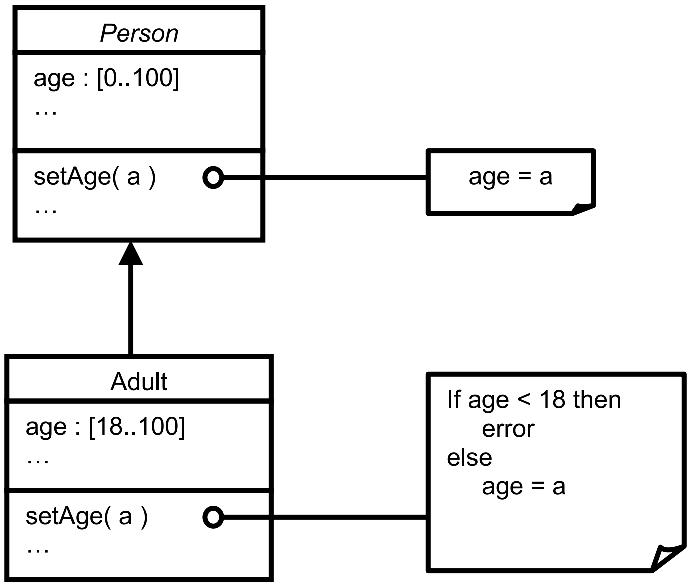
04.4. Specialization (Restriction)

We want to add a class to existing hierarchy of classes having many similarities to already existing classes but some part of its behaviour is different or restricted. In that case we will use the concept of specialization.

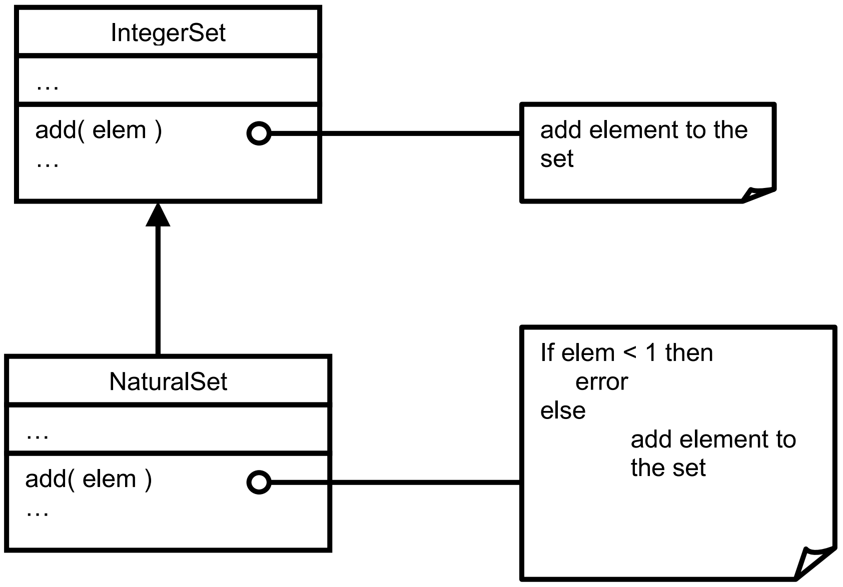
Specialization means that derived class is behaviourally incompatible with the base class
Behaviourally incompatibility means that base class can't always be replaced by the derived class
Derived class has some different of restricted characteristics than of base class.

Example – Specialization (Restriction)

Suppose we want to add one more class of Adult for some special requirement like for ID card generation such that it is a person but its age is greater than 18 and having all other behaviour of that of person class. One solution is that we write another class from beginning and write all code of person again in it with age limit, but better solution is that we derive adult class from person class and restrict age in that class as shown below in diagram,



Similarly Natural Numbers[2] are also Integers[3] with the restriction that natural numbers set can NOT contain zero or negative integers it consists of only positive integers so we can implement this relationship also as specialization,



[2] Natural numbers: positive integers only (numbers from 1 to …….onwards)

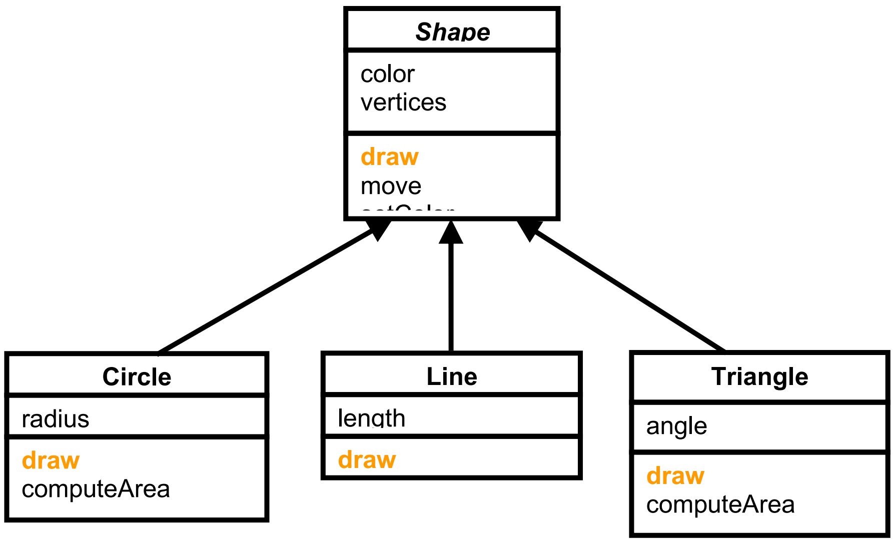
[3] Integers: all positive and negative numbers (…..-3 , -2 , -1 , 0 , 1 , 2 , 3………)

Add method behaviour is present in both base and derived classes but derived class behaviour is different in derived class. Derived class will not exhibit the behaviour of base class but it is **overriding** behaviour of base class with its own behaviour.

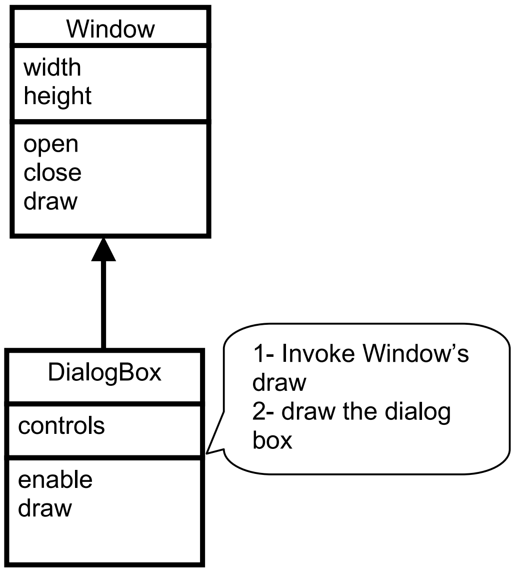
04.5.Overriding

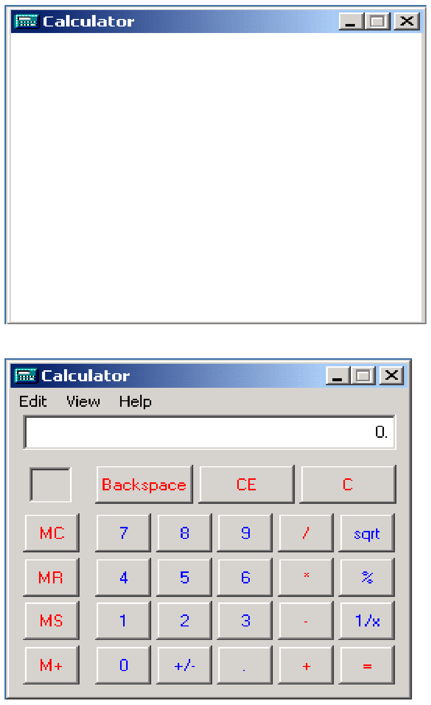
A class may need to override the default behaviour provided by its base class
Derived class overrides the behaviour of its base class.
Reasons for overriding
  Provide behaviour specific to a derived class (specialization)
  Extend the default behaviour (extension)
  Restrict the default behaviour (restriction)
  Improve performance
It is used for the implementation of inheritance.

Example – Specific Behaviour (Specialization)

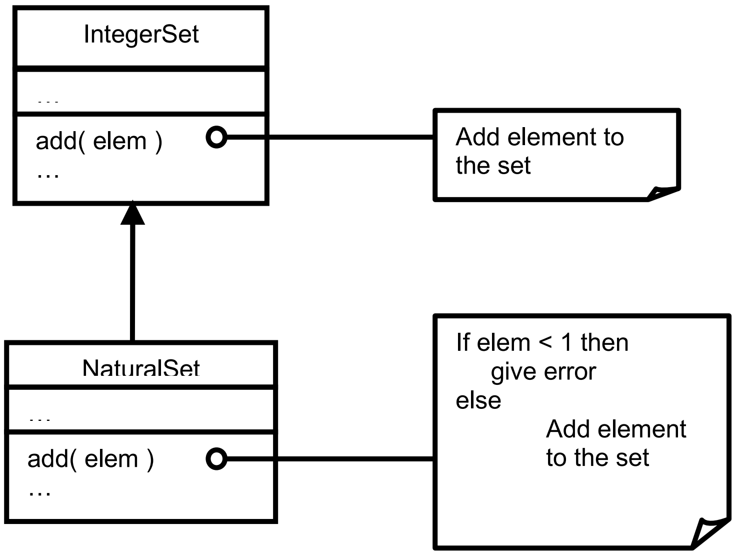


Example – Extention

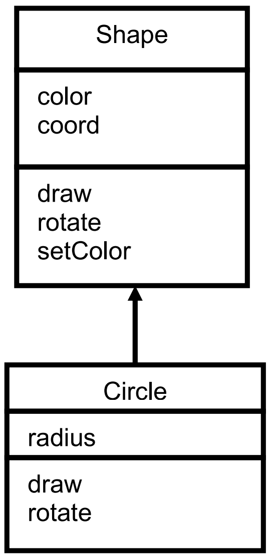




Example – Restriction



Example – Improve Performance
Class Circle overrides *rotate* operation of class Shape with a Null operation.

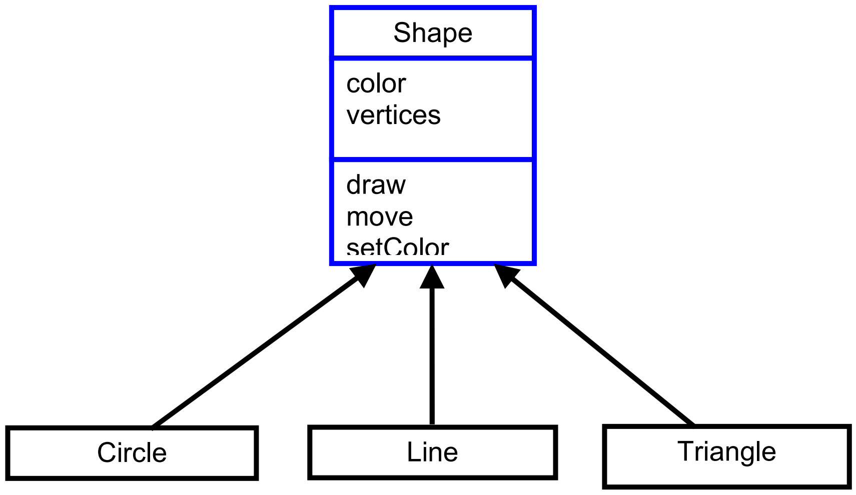


04.6. Abstract Classes

In our examples we made classes for shape and person. These are abstract concepts and the classes we make against abstract concepts are called abstract classes. They are present at or near the top in the class hierarchy to present most generalized behaviour.

An abstract class implements an abstract concept
Main purpose is to be inherited by other classes
Can't be instantiated
Promotes reuse

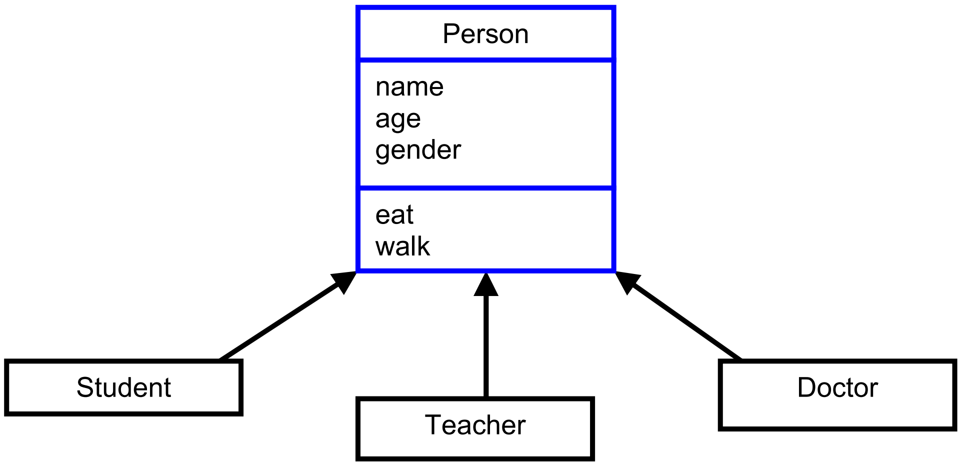
Abstract Classes - Example I



Here, Shape is an abstract class

Abstract Class	Shape			
Concrete Classes	Circle	Line	Triangle	….

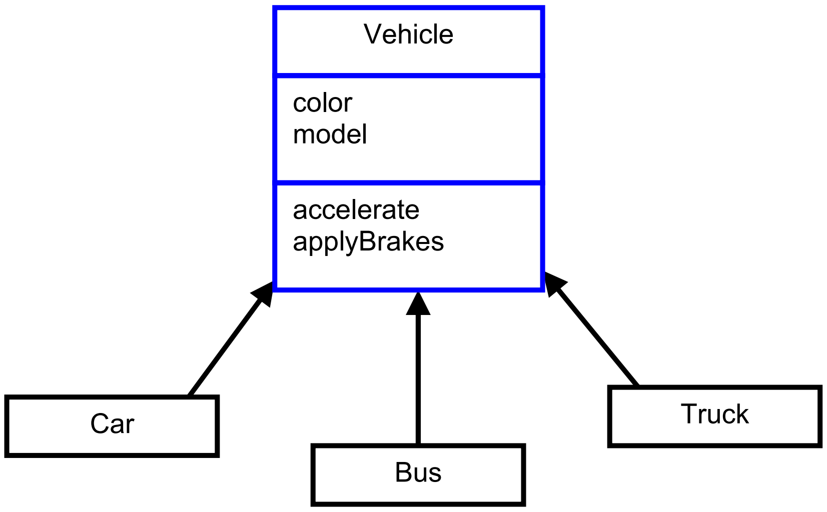
Abstract Classes - Example II



Here, Person is an abstract class

Abstract Class	Person					
Concrete Classes	Student	Teacher	Doctor	Engineer	Director	….

Abstract Classes - Example III



Here, Vehicle is an abstract class

Abstract Class	**Vehicle**			
Concrete Classes	Car	Bus	Truck	….

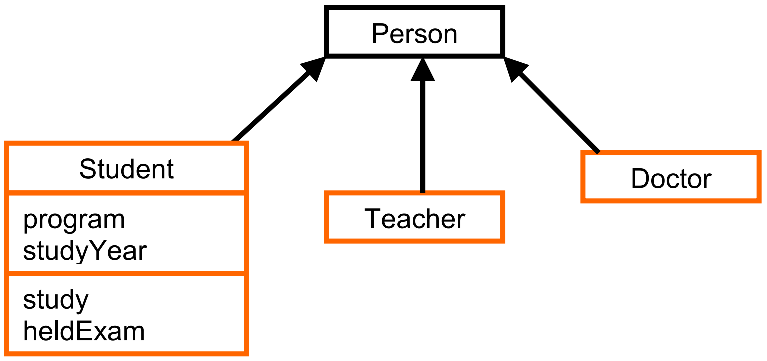
Abstract Classes can not exist standalone in an object model
While making object model we start by finding out objects in our object model and then we find out objects having common attributes and make them in the form of general classes at the top of class hierarchies.

04.7.Concrete Classes

The entities that actually we see in our real world are called concrete objects and classes made against these objects are called concrete classes.

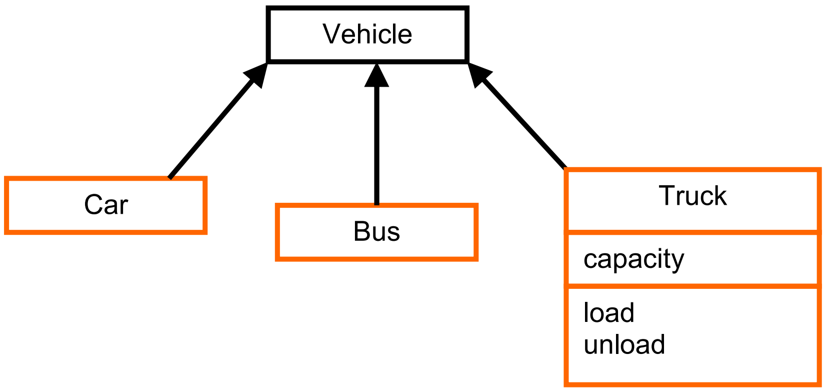
A concrete class implements a concrete concept
These are used to instantiate objects in our programs
Provides implementation details specific to the domain context

Concrete Classes - Example I



Here Student, Teacher and Doctor are concrete classes

Concrete Classes - Example II



Here Car, Bus and Truck are concrete classes

- A concrete class may exist in an object model independently
- Concrete classes mostly lie below the top of class hierarchy in a good object model.

If there is an abstract class then hierarchy exists in the object model as there will definitely be some concrete classes as well derived from this abstract class otherwise there is no use of abstract class.

Glossary:

a. Natural numbers: numbers from 1 to …….onwards
b. Integers: all positive and negative numbers …..-3,-2,-1,0,1,2,3………
c. Whole numbers: numbers from 0 ,1 ,2, 3 ….onwards (natural no's including 0)

Some times whole numbers are also called numbers without fractional part.