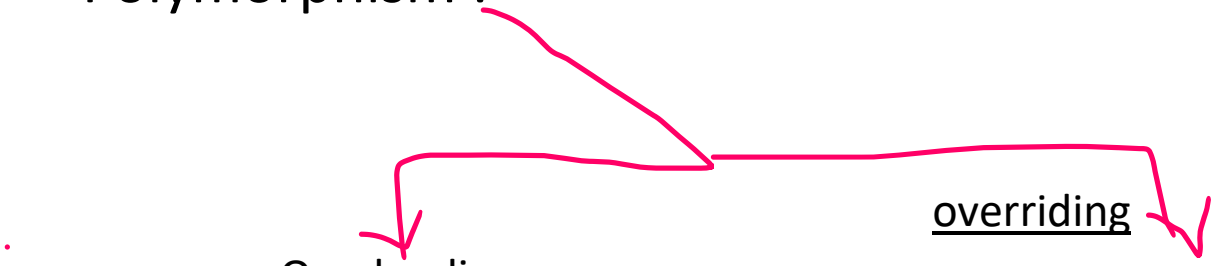
Abstraction : remove unnecessary things and group

Encapsulation : grouping things - class (methods and attribute)

Information hiding : In Encapsulation

Inheritance : generalization/ specialization (parent and child)

Polymorphism :



Overloading

Within the class

Can use same as the function name with deferent parameters

1. Default constructor
2. Overload constructor

overriding

Between the class (Inheritance only)

Virtual usage

Same function in different class

When we use virtual, that class called as "**Abstract class**".

All object should be **dynamic**.

if we assign the abstract class to a zero value, then it is known as a Pure Virtual class. : **pure virtual class**

Virtual Area() = 0;

Static

Shape s;

//function call

s.area();

Dynamic

Shape *s = new Shape();

s->area();

1. Composite

create a pointer using part class in whole class.
implement constructers (default constructer , overload parameter should be that pointer).
destructor.
in main, we create object only as **whole class pointer**. we do not create object as part class pointer.

2. Aggregation

same as composite **but do not use constructers** to join each table.
we use **add function** to join classes together in whole class (void addItem(Store *s[]);)
in main,

1st ->create part class object as pointer
2nd ->create whole class object as pointer
3rd ->join each class using **add function**

3. Uni-directional

create a pointer using part class in whole class.
implement constructers (default constructer , overload parameter should be that pointer).
destructor.

1st ->create part class object as pointer
2nd ->create whole class object as pointer and pass part class pointer as argument using constructor .

4. bi-directional

choose one class for **constructer (order)**, and other class for **add function (customer)**

In main,

if we need join customer- >order

Customer *C1 = new Customer("Thushara", "Kegalle");

Order *O1 = new Order("001", C1);

if we need join order - >customer

Customer *c1 = new Customer(“Thushan”, “kandy”);

Order *o1 = new Order(“

c1->addOrder(o1);

4.Association

create part classes(Student, Course) same as uml.

Association class

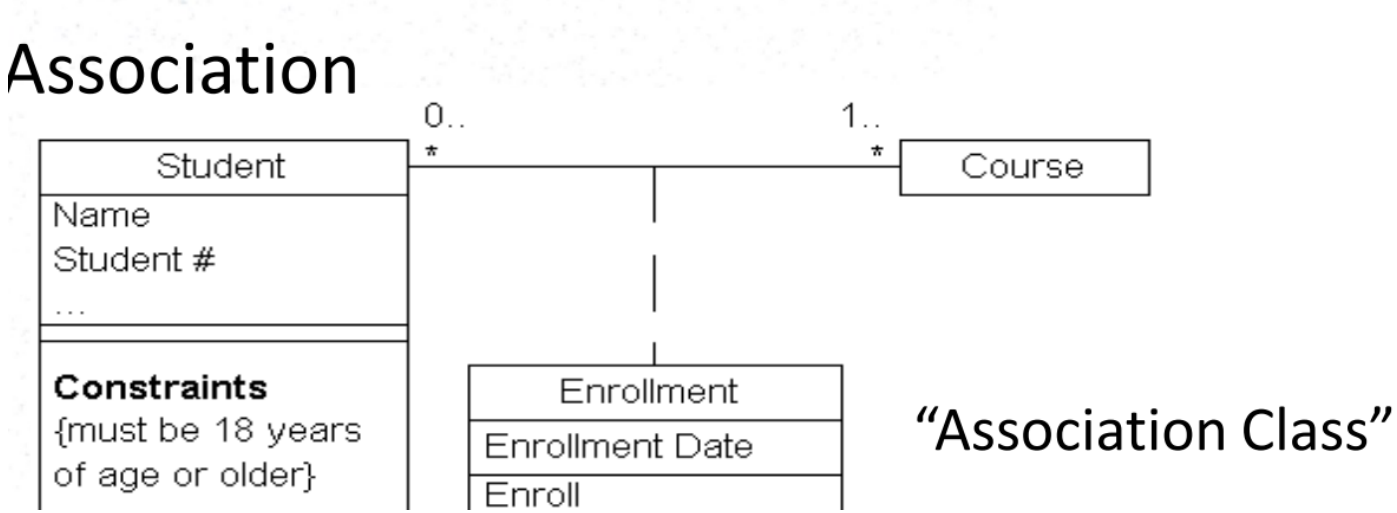
1st ->create part classes object as pointers

2nd -> join them using association class constructer.

in main,

should be implement class object as pointer separately.

use association class to join them



- Dependency

  same as uni-direction

  using add function to join them in whole class

Relationship implementation

Constructer

Composite

Uni-directional

Add function

Aggregation

bi-directional(one class uses **constructor** method,
other class uses **add function** method)

Association