3.4 Features of Object-oriented Programming

Operator Overloading:

- It is a type of polymorphism in which an operator is overloaded to give user-defined meaning.
- To add two integers we will use + operator.
- To add or concatenate two strings we use + operator.

How to overload the Operator:

- an operator function is defined inside a class to overload an operator.

Syntax:

```
Class class_name
{
        ---------
Public:
        return_type operator sign(args)
        {
        ----------
        }
};
```

Syntax:

i. Class function:

```
return_type operator op(){
Body;
}
```

ii. Frined function:

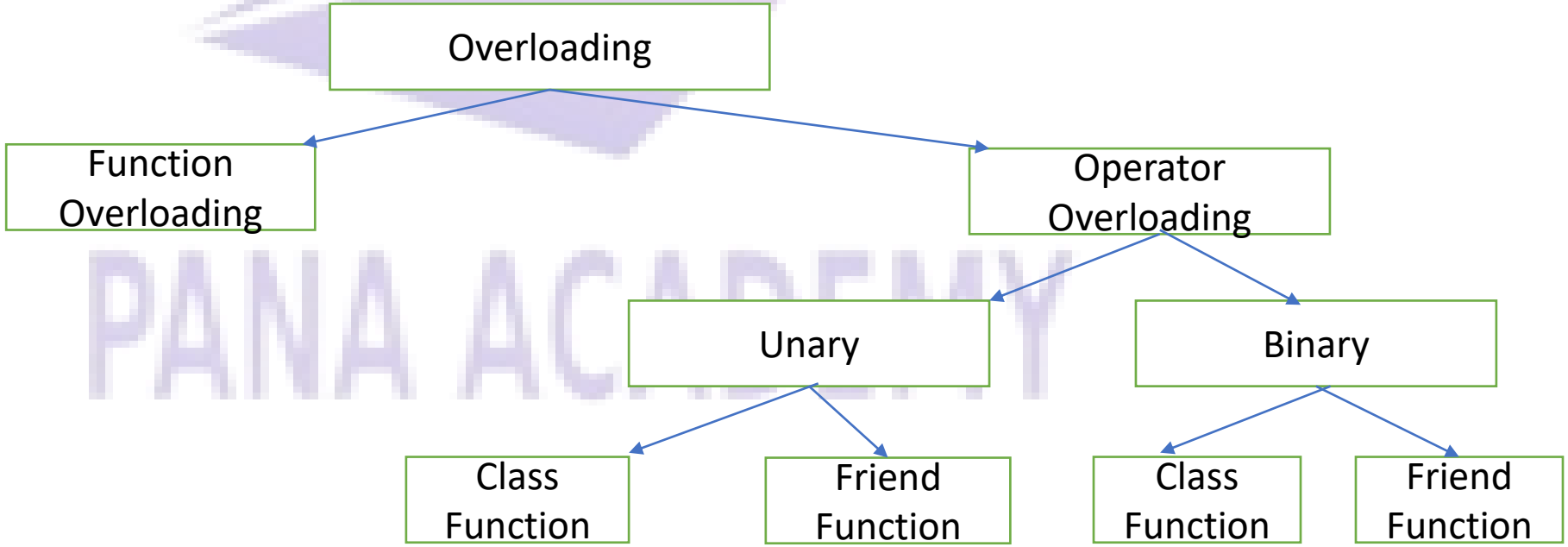
```
Friend return_type operator op (args);
```

Rules for operator overloading:

- Only existing members can be overloaded, we can not create our own operator to overload.
- The overloaded operator must have at least one operand of the user-defined type.
- It follows the syntax rules of the original operator. This means we can not change the basic meaning of operators.
- Some operators can not be overloaded:
  - Member access operator (.)
  - Pointer to member access operator (.*)
  - Scope resolution operator (::)
  - Size operator (size of)
  - Ternary operator (?:)
- We can not use friend function to overload some operators.

Overloading unary operator:

- Unary operator: operators which contains only one operator and one operand i.e. +a.
- Binanry operator: operator which contains one operator and two operands i.e. a+b.
- Unary operators are the increment and decrement operators ++ and --



Data Conversion:

- Conversion of data of one type to another type.
- Types:

1. Basic/primitive data type- int, float, char.
2. User-defined data type: class, structure.

Combinations:

a. Basic to basic: automatic { int x, float y=3.14; x=y; cout<<x; print x=3;}
b. Basic to user-defined.
c. User-defined to basic
d. User-defined to user-defined

- Basic to user-defined:

Test t1;

Int x=6;

T1=x; //basic to user-defined conversion.

- It is done by the constructor with one argument of basic type as follows:

```
Class test
{
        private:
        ----------
        public:
        test (data_type)
        {
        // conversion code;
        }
};
```

- User-defined to Basic:

int x;

test t1;

x=t1; // user-defined to basic

-We need type casting operator function.

-Typecast function:

```
operator data_type()
        {
                return (datatype-value);
        }
```

- User-defined to User-defined:

test t1; // test t1,t2; t1=t2; this is the possible objects are in the same class. Using copy constructor

sample s1;

t1=s1; // not possible because there are two different classes. User-defined to user-defined.

- Ways two- constructor and type casting operator function.

Introduction:

- Inheritance is a relationship between two or more classes where derived class inherits the properties of base class is called inheritance.
- Base class: it is a class whose properties are inherited by another class. It is also called as super class or parent class.
- Derived class: it is a class that inherits the properties from base class. It is also called as sub class or child class.
- Eg:

Base class	**Derived class**
student	Graduate student, under graduate student
shape	Triangle, rectangle, circle, cube
employee	Faculty, staff

Defining derived class

- Syntax:

Class derived_class_name: visibility mode (optional access specifier) base class_name

{

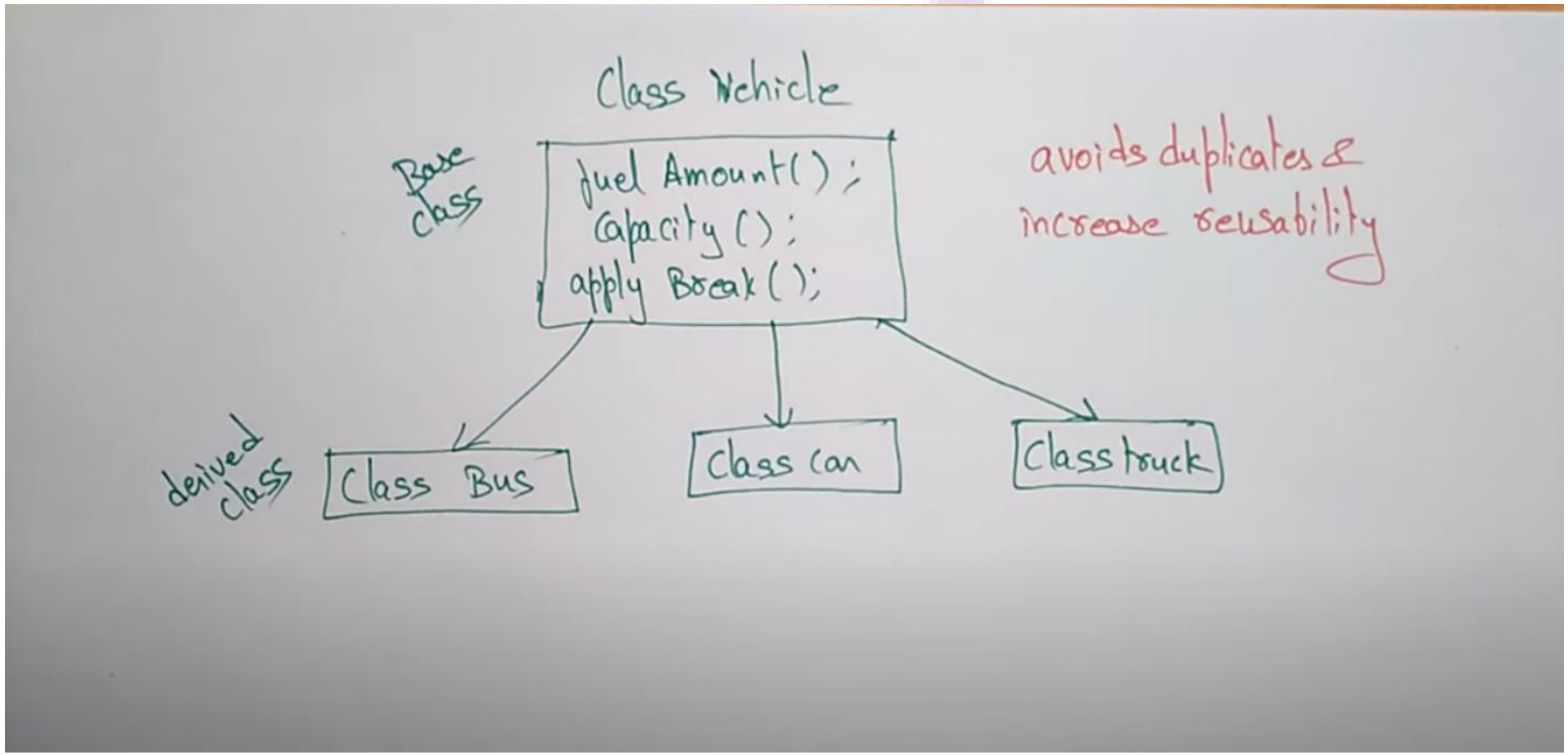
---------- members of derived class

}

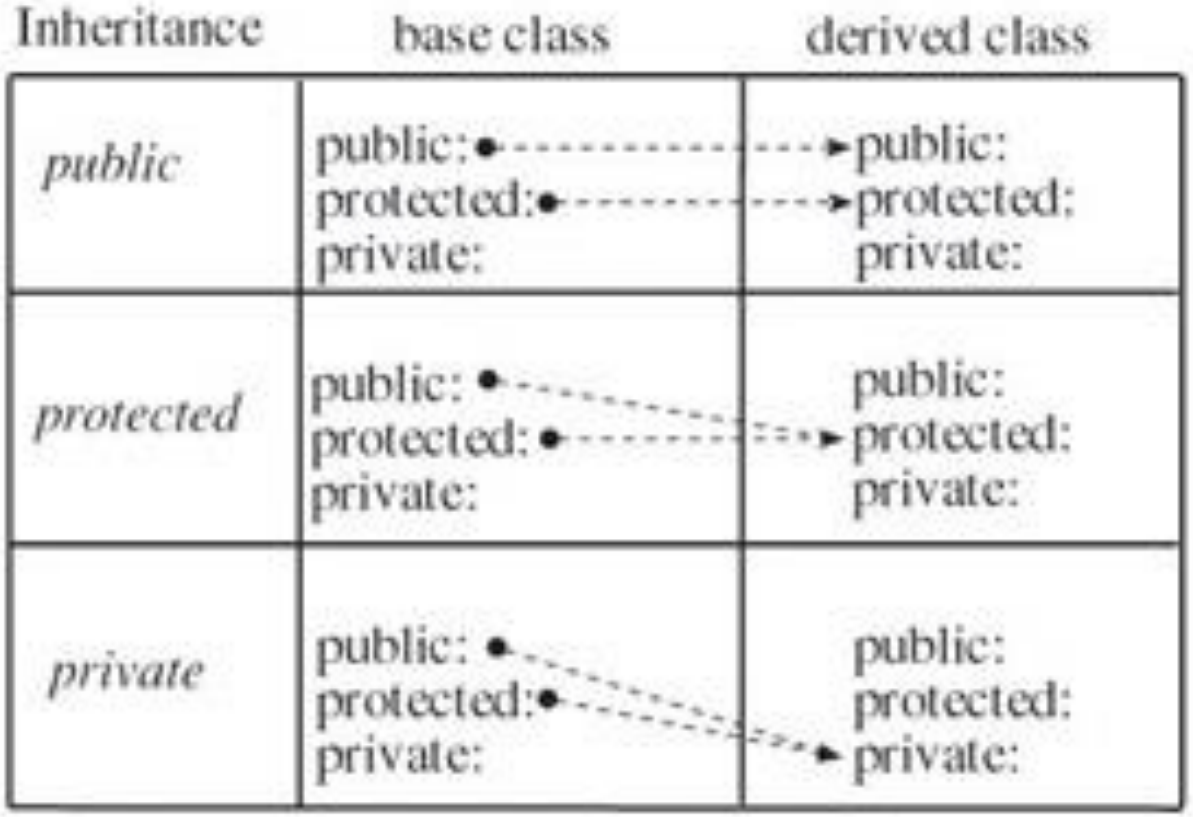
*Why and when to use inheritance:

Eg: consider a group of vehicles (classes:bus, car, and truck)

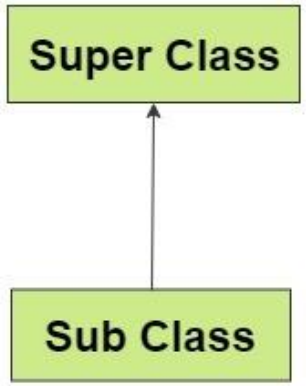
Methods-> fuelamount(), capacity(), applybrake(); these three methods are the same for three classes. These increases chance of error and data redundancy to avoid these type of situation inheritance is used.


Class Vehicle
Base class
fuel Amount();
Capacity();
apply Break();
avoids duplicates &
increase reusability
derived class
Class Bus
Class Car
Class Truck

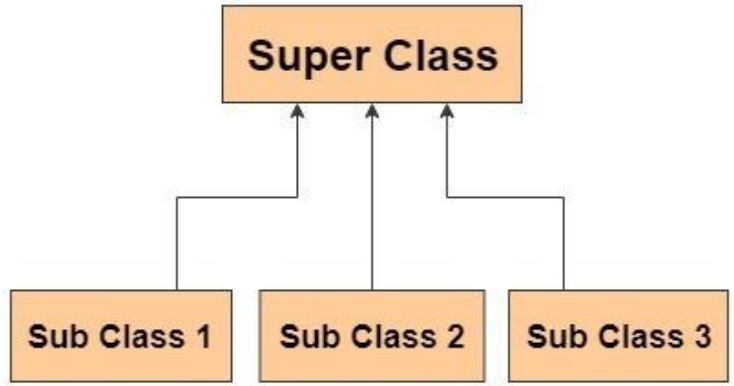
Access Specifier:



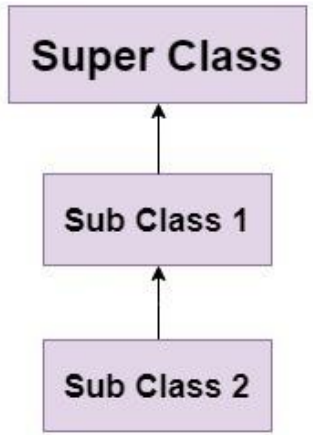
Single Inheritance



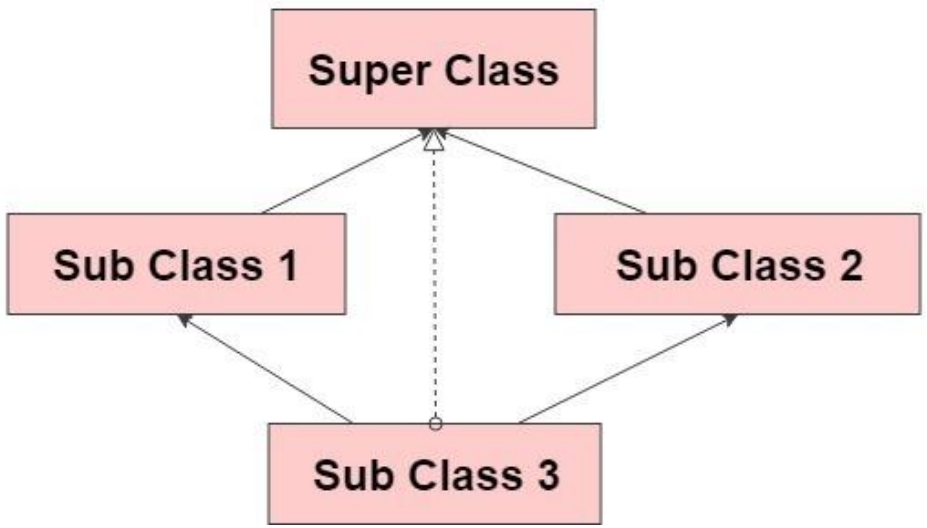
Hierarchial Inheritance



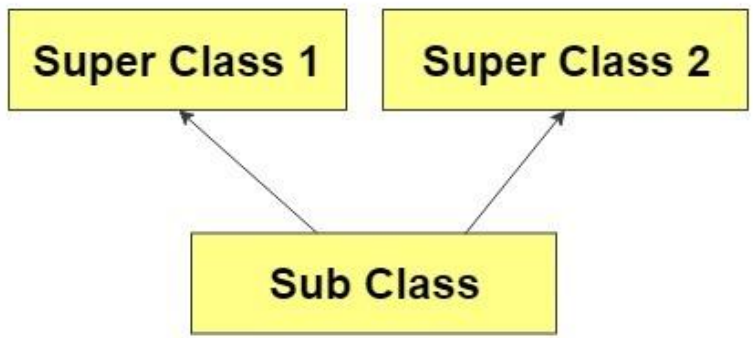
MultiLevel Inheritance



Hybrid Inheritance



Multiple Inhertance



Constructor and Destructor in derived class:

- Constructors are **special member methods invoked during object creation** that are used for **initialization**.
  - Have the **same name as the class.**
  - **No return type** is specified
  - Can be **overloaded**
  - Go into the **public section** of the class declaration.
- Destructors are **special member methods** that are called on each instance of an object used to **release the memory and other resources associated with an instance** after it is no longer needed.
  - Have the **same name as the class preceded with a tilde(~)**
  - **Invoked automatically when an object is destroyed**
  - Have **no return type and no params**
  - Only one destructor is allowed per class, so there is **no overloading** of Destructors
  - **Called when a local object goes out of scope or when you delete a pointer to an object.**
  - When an object goes out of scope, the **destructors are called on each instance of an object in the order that the objects were created**

Fill in the Blanks

1. The _____ keyword is used to overload an operator.
2. The conversion of a basic data type to a class type is accomplished by defining a _____ of the class.
3. To convert class type to basic type an overloaded _____ function needs to be defined.
4. An operator can be overloaded either using _____ or using _____ .
5. The operators (), [], -> and = can be overloaded only as _____ functions.

Multiple Choice Questions

1. Which of these operators can be overloaded?
   (a) scope resolution (b) `sizeof` (c) array subscripting (d) conditional operator
2. Which of these operators can only be overloaded as friend function?
   (a) ++ (b) -- (c) >> (d) +
3. Which of these keywords is used to overload an operator?
   (a) `overload` (b) `operator` (c) `op` (d) none of these
4. The operator function defined as a friend function of the class is known as
   (a) friend function (b) friend operator function (c) member function (d) all of these
5. Which of these statements is not true for operator overloading?
   (a) The number of operands required with the operator can be changed.
   (b) New operators cannot be created.
   (c) The implementation of the operator can be changed and not the syntax for using it.
   (d) All overloaded operators except the assignment operator '=' can be inherited by the derived classes.

Fill in the Blanks

1. _____ is a mechanism of deriving a new class from the old class in such a way that the new class inherits all the members of the old class.
2. The class that is inherited by the new class is called _____ .
3. The _____ class inherits the data members of the old class.
4. The _____ of the base class in the derived class definition determines the way the derived class inherits the base class.
5. When a derived class is inherited from a single base class, it is referred to as _____ .

Multiple Choice Questions

1. In which of these inheritances is a derived class inherited from more than one base class simultaneously?
   (a) multiple inheritance (b) single inheritance (c) multilevel inheritance (d) hierarchical inheritance
2. In which of these inheritances is a derived class inherited from a single base class?
   (a) multiple inheritance (b) single inheritance (c) multilevel inheritance (d) hierarchical inheritance
3. In which type of inheritance does the derived class have all the members of its direct base class as well as its indirect base class?
   (a) multiple inheritance (b) single inheritance (c) multilevel inheritance (d) hierarchical inheritance
4. Which type of class prevents its duplication?
   (a) derived class (b) base class (c) indirect base class (d) virtual base class
5. In which of these types of nesting does an enclosing class have an object of another class as its member?
   (a) aggregation (b) inheritance (c) multilevel inheritance (d) hierarchical inheritance

- https://www.geeksforgeeks.org/c-plus-plus-gq/operator-overloading-gq/
- https://www.geeksforgeeks.org/c-plus-plus-gq/inheritance-gq/
- 13. operator overloading: 1. operator 2. constructor 3. casting operator 4. member function , friend function 5. member
- 1. c 2. c 3. b 4. b 5. a
- 14. Inheritance: 1. Inheritance 2. base class 3. derived class 4. Access specifier 5. single inheritance
- 1. a 2. b 3. a 4. d 5 a