07-12-2017

Point p1, p2

p1.

p1 + p2

Operator overloading:

```
class Point {
    int x, y;
Public:
    Point () { x = y = 0; }
    Point (int x, int y)
    { this -> x = x;  this -> y = y; }
```

overload + operator:

```
Point   Point :: operator + (const Point &obj)
{
    Point temp;
    temp.x = this -> x + obj.x;
    temp.y = this -> y + obj.y;
    return temp;
}

bool Point :: Operator < ( cont Point & obj )
{
    if ( this -> x < obj.x && this -> y < obj.y)
        return true;
    return false;
}
```

prefix (++) (--)

a = ++b

void

✓ Point operator++ ( )

{

++(this->x);

++(this->y);

return (*this);

}

f, (int, int)

f, (int)

d = b++

postfix (++) (--)

Point operator++(int) //(int dummy)

{

Point temp = (*this);

this->x++;

this->y++;

return temp;

}

a = b++

Stream operators

- input stream → cin → istream
- output stream → cout → ostream class → L → R
- → can not be overloaded in class.

Friend function: function which has access to class private members.

Friend class

Friend Function

cout << R2 ; cout << S.P1 ;

friend ostream& operator << (ostream & out, const Point & obj);

friend ostream& operator << (ostream & in, const Point & obj)
{
    cout << obj.x;
    cout << obj.y
    return out;
}

friend istream& operator >> (istream & in, const Point & obj);

instream & operator >> (istream & in, Point & obj)
{
    in >> obj.x;
    in >> obj.y;
    return in;
}

cin >>

objects can be converted in → primitive DT
Custom DT

object conversion :
Jis main convert krna h.

return by default

operator int ()
{
    return x+y;
}

int i
Point P(4,4);
i = P

```
operator double()
{
    return (double)(x+y);
}
```

Subscript Operator => returns ref of location

14-12-2017

Inheritance: → feature → Reusability

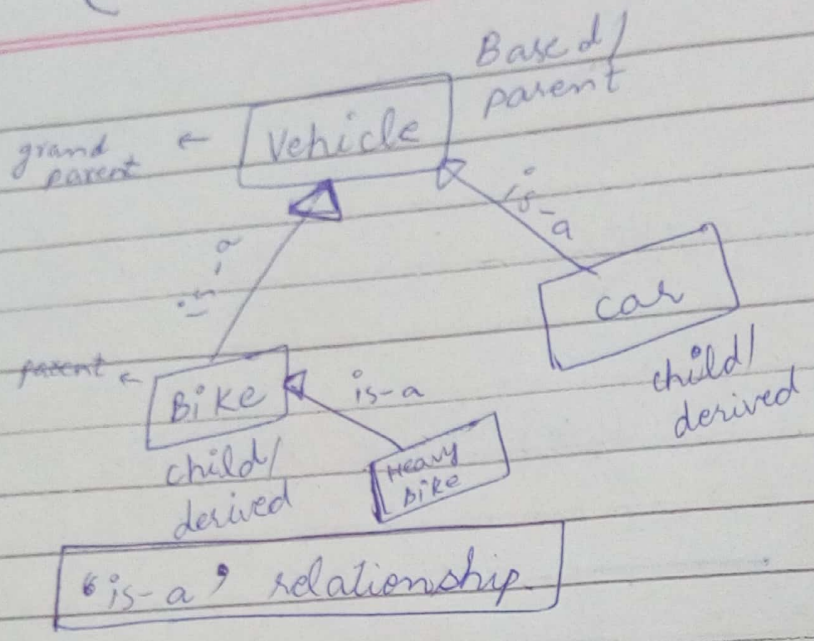
Inheritance allows a new class to be based on an existing class. New class inherits all the member variables and functions except the constructor and destructor of the class it is based on.

→ Base class / Parent Class
→ Drive class / child class

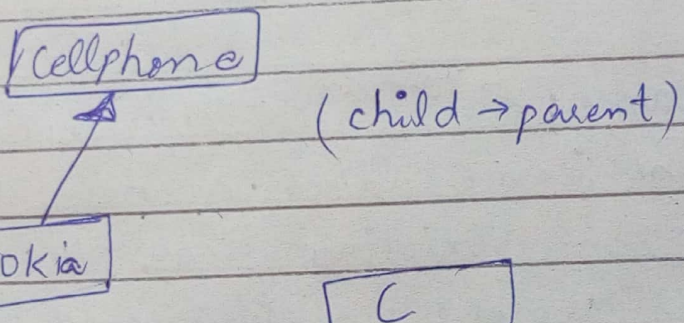
Generalization	Specialization
↓	↓
parent class / Base class e.g. Vehicle	Child class / drive class e.g. bike, car

I ate pizza in breakfast

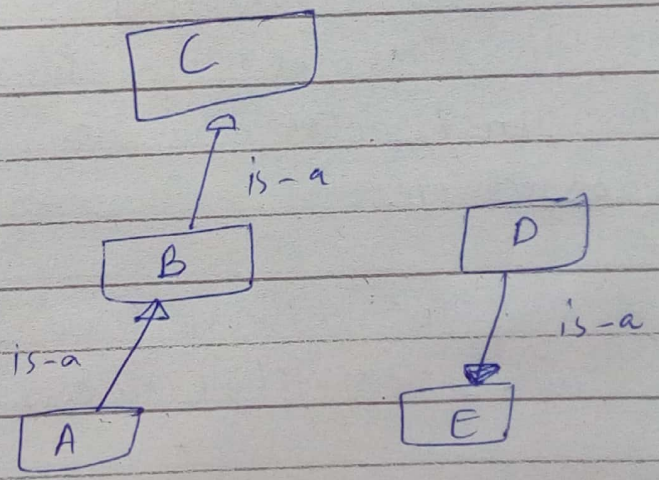
I want a [illegible]
Im your bae :(
before anyone else <3



'is-a' relationship



A is a B
B is a C
D is a E



Static Member is also inherited.

child Parent
class A : public B
{
  == private
  == protected
}

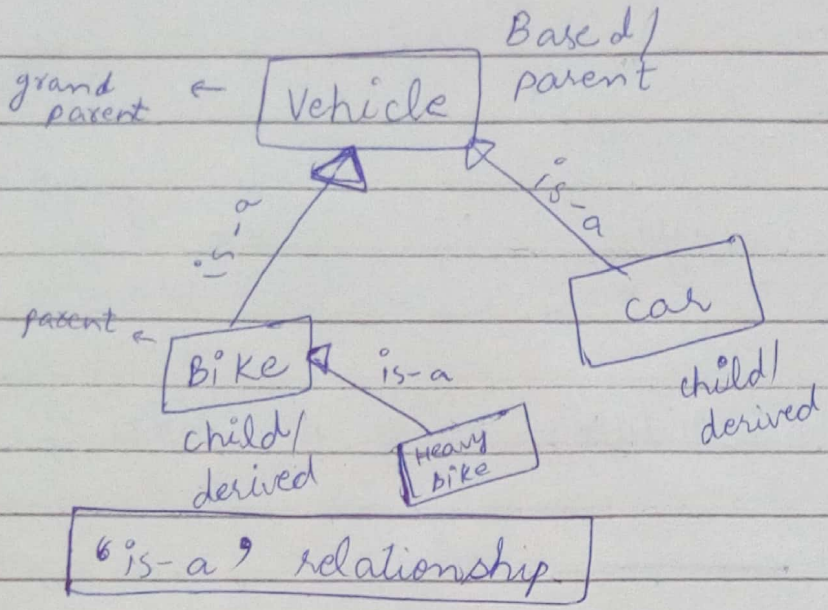
① 15.1 - 15.3 = Checkpoints

② Final
Assignment
Quizzes

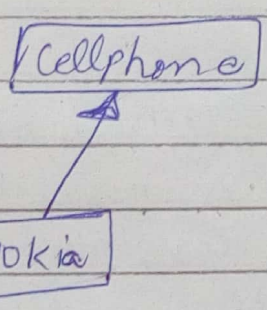
③ How can we call parameterize constructor when child class constructor invoked.

I ate pizza in breakfast

I want a ring too :(
Im your bae :(
before anyone else <3

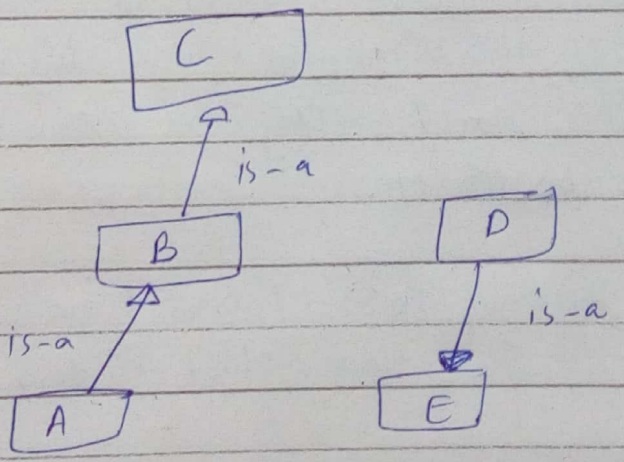


'is-a' relationship



(child → parent)

A is a B
B is a C
D is a E



Static Member is also inherited.

child Parent
class A : public B
{
private
protected
}

① 15 1-15 3 checkpoints

② Final
Assignment
Quizzes

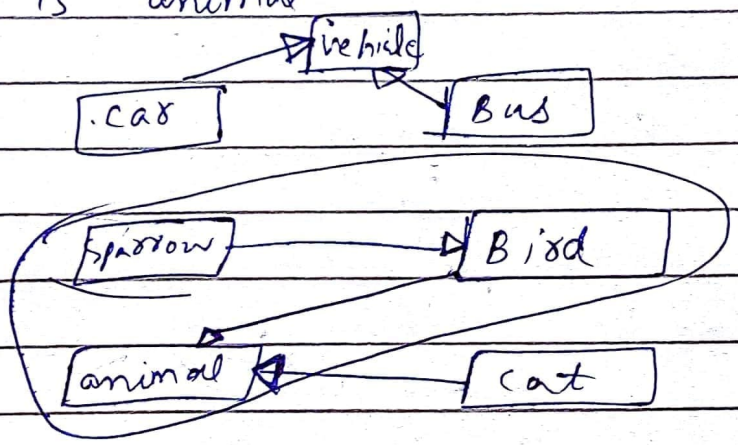
③ How can we call parameterize constructor when child class constructor invoked.

19-12-2017

Car :

Parent Constructor
Child "
Child Destructor
Parent "

Car is vehicle
Bus is a vehicle
Sparrow is a bird
Bird is animal
Cat is animal



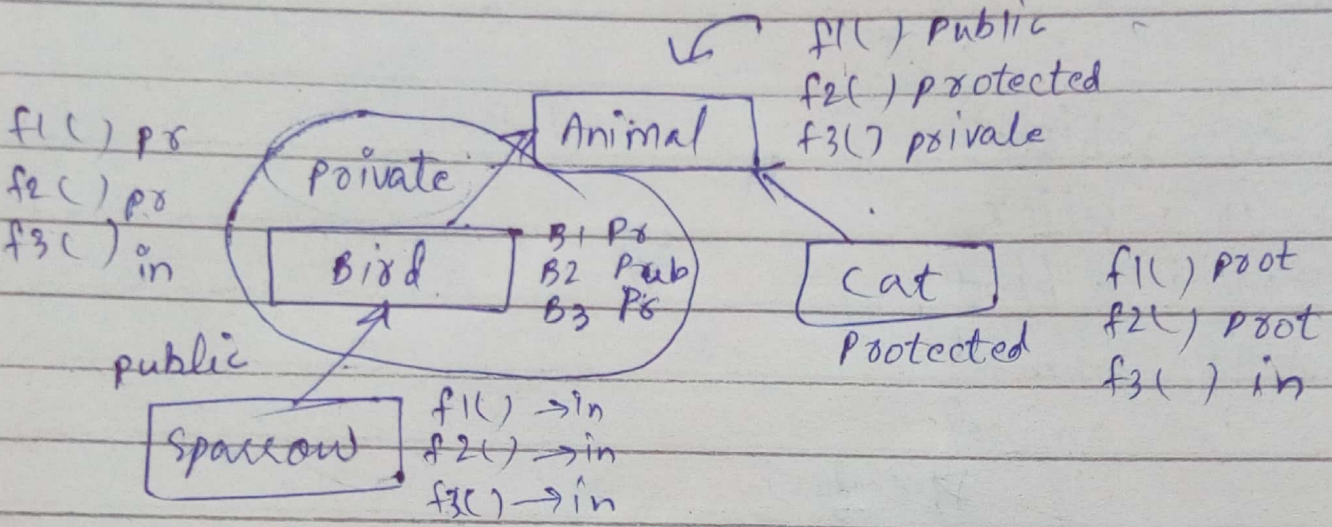
Chain inheritence
multi level inheritance

Access Specifier:

1) Public
2) Private
3) Protected:

Protected members of a base class are like private members, but they may be accessed by drive classes. Base class access specification determines how private, public or protected base class members are accessed when they are inherited by the derive classes.

private members of base class are not accessible to child class. therefore protected members are used.



Func in parent

↓

(Function redefinition in child)

Static Binding

↓

If func is redefined, compiler decides to call which class' function, by the calling object type.

Function redefine vs Func Overloading

↓ ↓

(parameters same/diff) (diff parameters)

↓

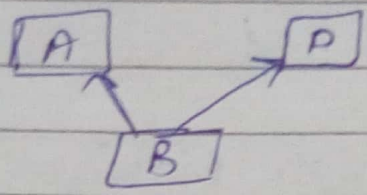
In case of inheritence

CP:
15.4 - 15.8

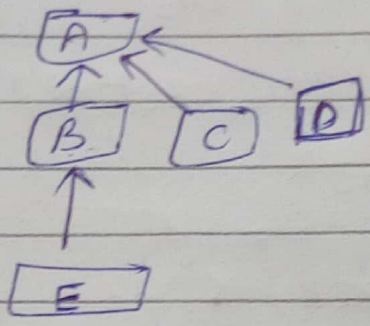
Scanned by CamScanner

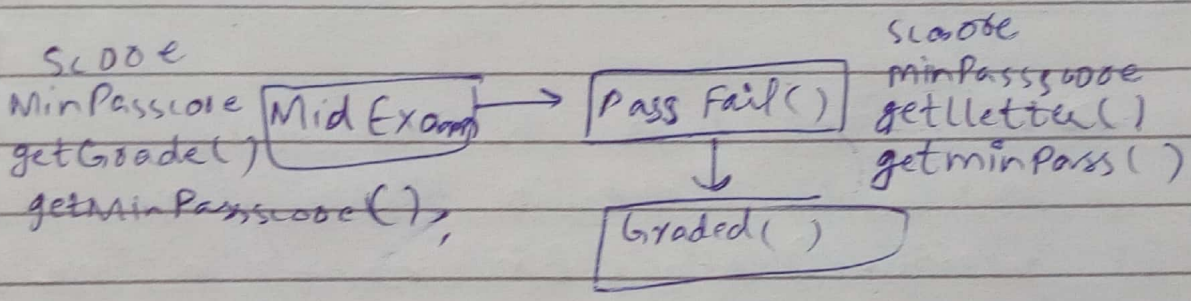
22-12-2017

More than one parent of a class → multiple inheritance



- Multilevel Inheritance :





In long hierarchy, then func called by object uses the func inherited from immediate parent.

→ Parent class pointer can point to child class pointer. GoadedAct *gptr = new midExam(); ✓

→ Child can not point to parent pointer.

MidExam * ptr = gtr; ✗

in parent

- gpto → getLetterGrade(); ✓
- gpto → getMinPassScore(); X

x ↓ in child ↖ static Binding

→ Casting (child pointer can not pointed to parent)
↓ to do this

MidExam * Mpto = static_cast < MidExam *) gpto

child = parent

Now Mpto = gpto ✓

before Mpt = gpto X

Virtual → if parent : func is redefined in child make it virtual in parent. Dynamic Binding → run time
↳ according to class it is pointing to.

— = & —

"Jahan virtual kity hain us se neechy waly ko lye hoga virtual."

Polymorphism: child's reference func is used.

---

void displayGrade (GradedAct &obj)

A B C P F Pass F

Chap 15:

- To call base class constructor → constructor delegation
- Base class constructor is executed before the derived class constructor.
- ~~Derived~~ Based Class Arguments:
  - → Derived class constructor parameters
  - → Literal values
  - → Global variables
  - → Expressions involving any of these

- Redefinition:

When derived class has the same function as the base class's member function.

- When an object of the derived class calls the function, it calls derived class's version.
- If a class inherits the redefined function, it inherits the one from immediate parent.

e.g. PassFailExam class is derived from the PassFailActivity, it ~~the~~ inherits the redefined getLetterGrade function.

- Polymorphism and Virtual members:

↳ The ability to take many forms.

Polymorphism allows an object reference variable or an object pointer to reference objects of different types to call the correct member functions, depending upon the type of object

being referenced.

void displayGrade (GradedActivity & activity)
{ == } PassFailActivity
class FinalExam → getletterGrade() redefined
class GradedActivity → getletterGrade()

Now if:

```
PassFailActivity test;
test.displayGrade (test);
```

Output:

```
GradedActivity :: getletterGrade ();
```

→ Static Binding: The process of matching a function call with a function at compile time.

• To remedy this getletterGrade() can be made virtual → dynamic binding → C++ determines which function to call at run time.

```
virtual char getlettergrade ();
```

This tells the compiler tho expect the funtion to be redefined in derived class. The compiler does not bind calls to the function with the actual function.

• By • value → static Binding

Base Class Pointers:

Pointers to base class may be assigned the address of derived class.

```
GradedActivity *exam = new PassFailExam ();
```
(GradedActivity: parent; PassFailExam: child)

```
exam -> getScore();
exam -> getLetterGrade(); (-> passFail)
```

It can only point to members that are part of GradedActivity class. i.e. setScore, getScore, letterGrade
But
```
exam -> getPointsEach()  (-> ERROR!)
```

'Is-a' relationship doesn't work in Reverse:

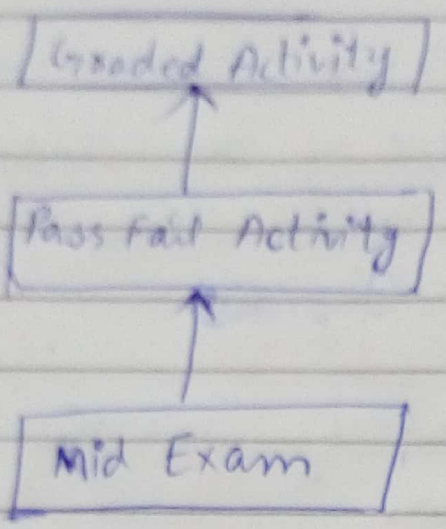
```
GradedActivity *gaptr = new GradedActivity();
FinalExam * fptr = gaptr  (ERROR!)
```
However,
```
FinalExam *fptr = static_cast<FinalExam *>(gaptr);
```

fptr now points to base class, but, we can access function of only base class.

```
fptr -> getPointsEach()  (ERROR!)
```

02-02-2018

- Multilevel Inheritance / Chain inheritance:



- Polymorphism → reference / pointer
  Function redefinition
- virtual functions redefine → function overriding
  ↓
  dynamical bounding

'override' keyword → function overrided.
↓
(same parameters) means it is defined virtual in upper functions.

'Final' keyword → function can not be over-rided further.

Virtual Destructors:

① If any functions is virtual in a class, make its destructor also virtual.

② Pointers are involved in a class and it is inherited, make virtual destructor.

Abstract base class:
Pure virtual destructor:

virtual void Draw() = 0;

→ function of this type of definition → pure virtual function → abstract class → class's object can not be created.

- Further classes in hierarchy will also become abstract, unless defined.