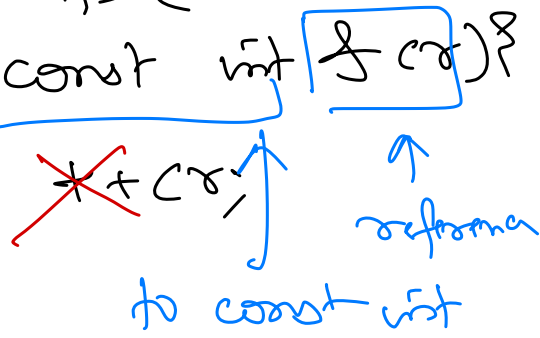
Constant Member function

```
void f1 ( int v) {
    ++v;
}

void f2 ( int &r) {
    ++r;
}

int main () {
    int a = 5;
    f1 (a); std:: cout << a ; => 5
    f2(a); std:: cout << a; => 6
    return 0;
}
```

```
void f3(
    const int &cr){
    ++cr;
}
```



```
class X {
    :
public:
    :
    void setVal(int val) {
        this->val = val;
        ++this;
    }
    int getVal() const {
        return val;
    }
    :
};
```

this ← points to the object for which member function is called.

this is a constant variable

return this->val;

getVal is Constant member function

this will point to a constant object

① for constant object or reference to constanst object or pointer to constant object, only constant member functions can be called.

② In a constant member function only content member functions can be called for the object pointed by "this".

```
:
int getVal() const {
    X obj;   obj.setVal(100);
    return val;
}
```

On day 10))

```
BigInt operator + ( BigInt obj2)
```

The correct way is.

```
BigInt operator + (const BigInt& obj2) const
```

Static Members

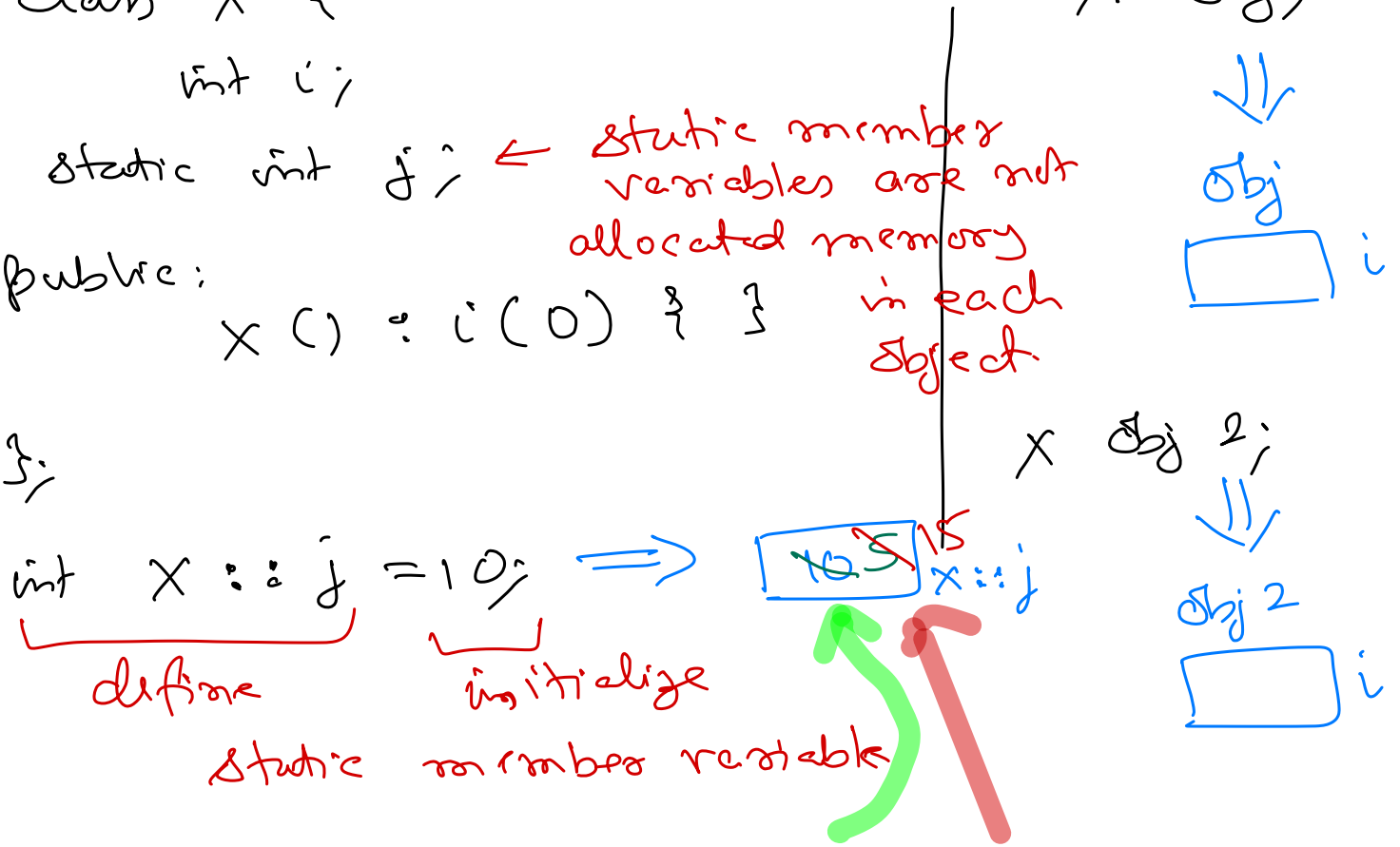
```
class X {
    int i;
    static int j;
public:
    X() : i(0) { }
};

int X::j = 10;
```

static member variables are not allocated memory in each object.

define

initialize

static member variable



① Static member variables can be accessed without an object.

```
X::j = 5;
obj.j = 15;
```

② static members are still member of class and hence access specifiers apply to them.

```
class X {
    int i;          ← non-static member variable
    static int j;   ← static member variable
public:
    X() : i(10) { }
```

```
int getI() const { return i; }   <- non-static member function

static int getJ() { return j; }
```

static member function

};

---

non-static members can only be accessed for an object.

static members can be accessed without an object of the class.

non-static member function receives this pointer

Static member function do not receive this pointer.

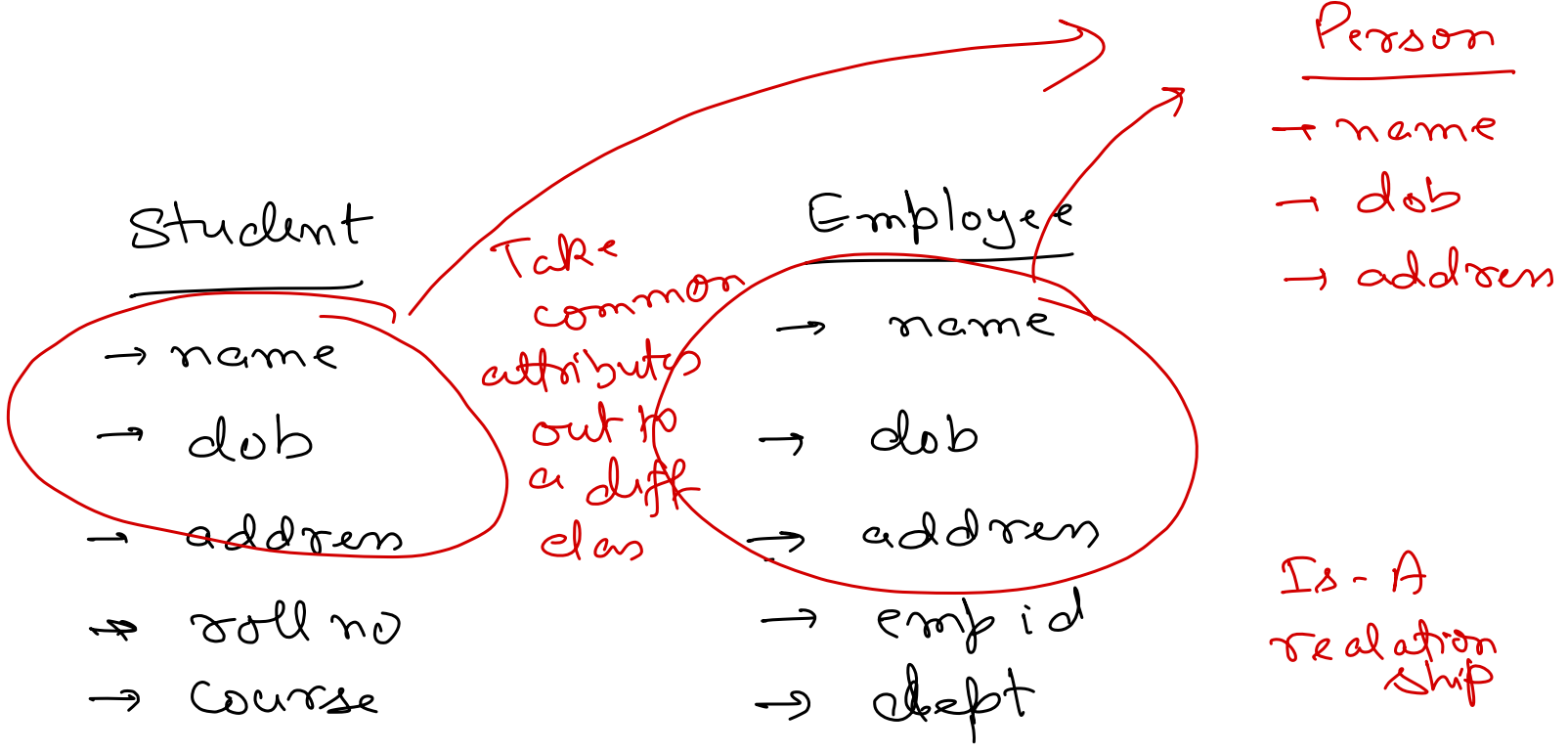
Inheritance

Student

- name
- dob
- address
- roll no
- course

Employee

- name
- dob
- address
- emp id
- dept


Student
→ name
→ dob
→ address
→ roll no
→ course
Take common attributes out to a diff class
Employee
→ name
→ dob
→ address
→ emp id
→ dept
Person
→ name
→ dob
→ address
Is-A relationship

```
class Student {
    char name[100];
    int rollNo;
public:
    void setName(char name[100]);
    void setRollNo(int rollNo);
    int getRollNo();
};
class Employee {
    char name[100];
    int empid;
```

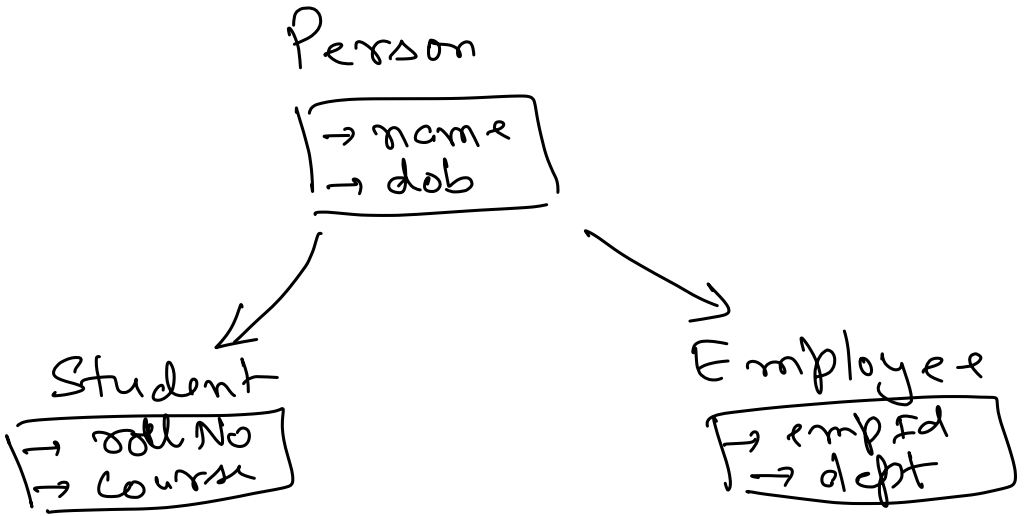
public:

void setName(char name[100]);

void setEmpId(int empId);

int getEmpId();

};



```
class Person {
    char name[100];
public:
    void setName(char name[100]);
};
```

```
class Student : public Person {
    int rollNo;
public:
    void setRollNo(int rollNo);
    int getRollNo();
};
```

Derived class

inheritance

Base class