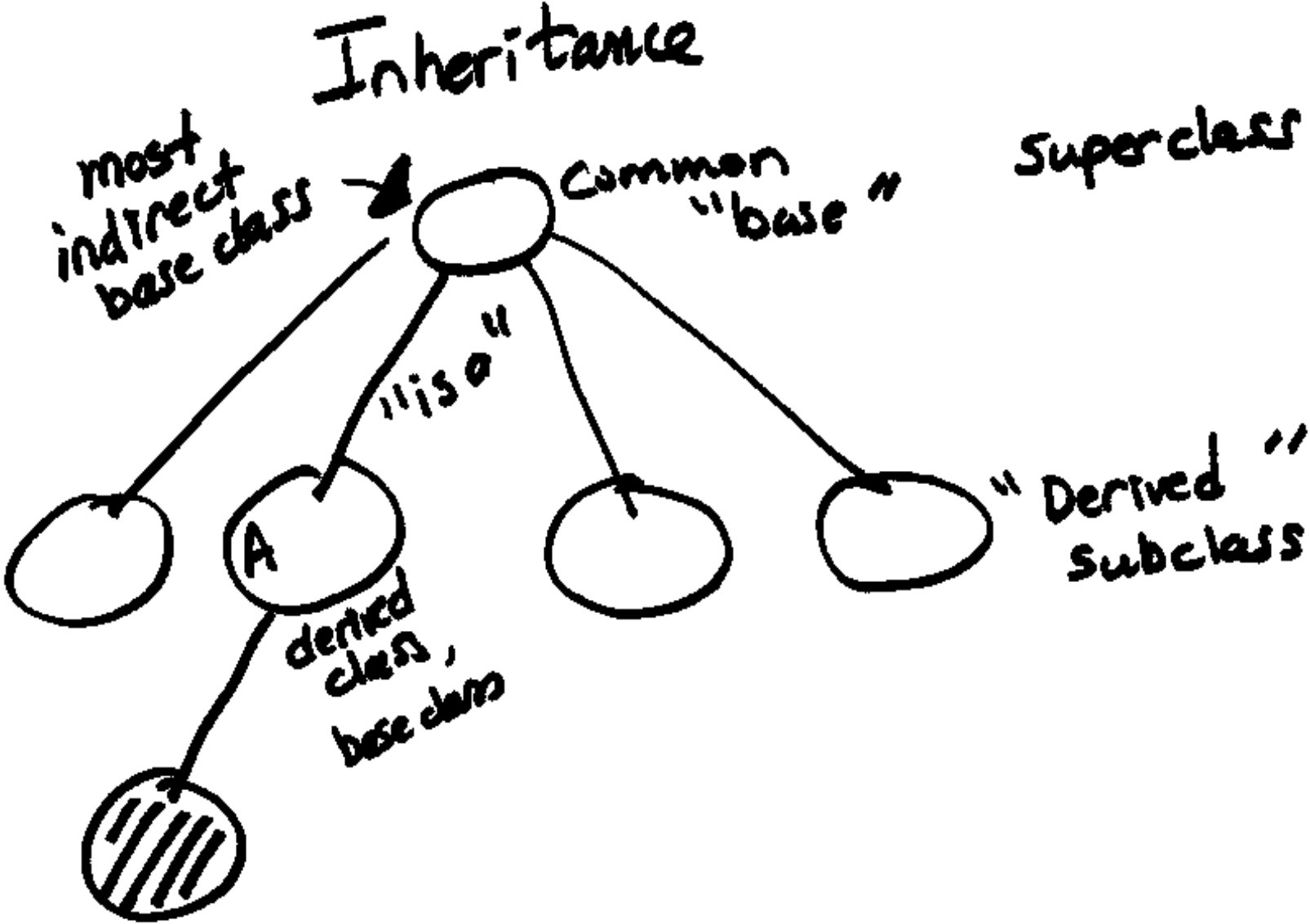
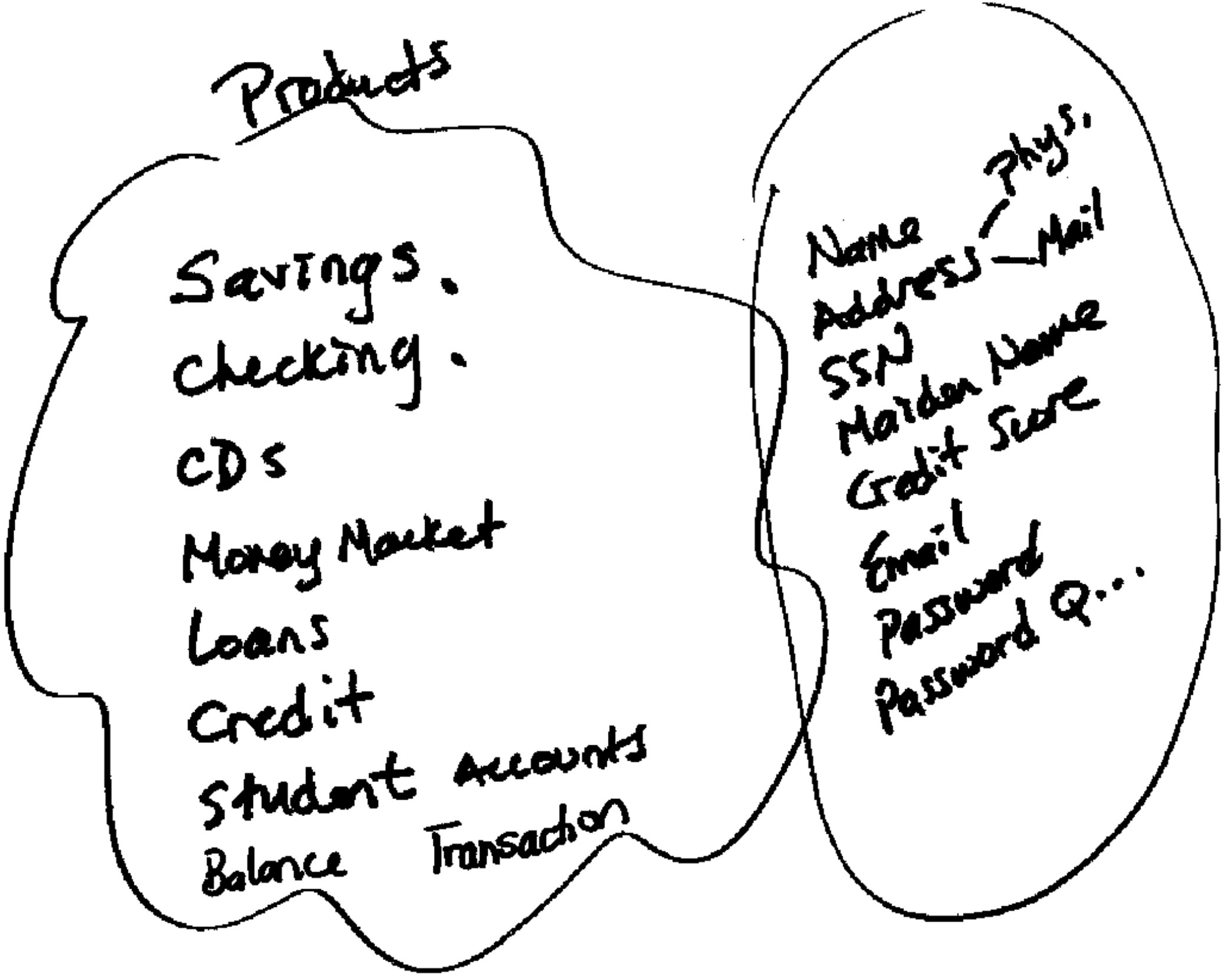
www.cs.pdx.edu/~karlaf

D2L.pdx.edu


Inheritance
most indirect base class
common "base"
Superclass
"is a"
A
derived class, base class
"Derived" subclass


Products
Savings.
Checking.
CDs
Money Market
Loans
Credit
Student Accounts
Balance Transaction
Name
Address - Phys. - Mail
SSN
Maiden Name
Credit Score
Email
Password
Password Q...

Derivation List

```
class Checking : public Account
{   public:
        // No display function!

    protected:

    private:

};
```

OOP

```
Checking obj;
obj.display();
```

Alternate

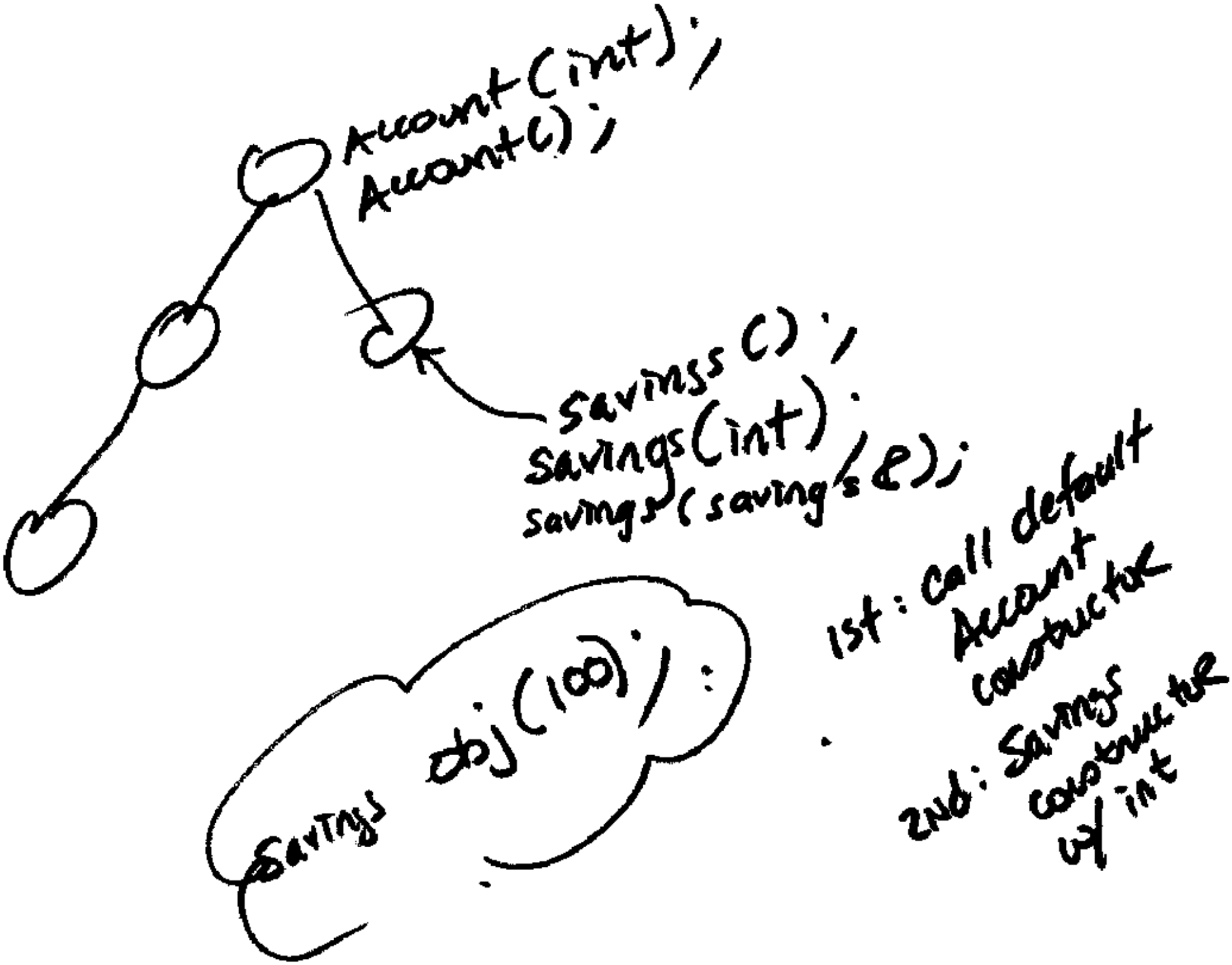
```
class checking : public Account
{ public :
      void display();
```

Client

```
obj.display();
```

```
void checking::
     display()
{
     :
Account::display();
```

scope resolution operator


Account(int);
Account();
Savings();
Savings(int);
Savings(Savings&);
Savings obj(100);
1st: call default Account constructor
2nd: Savings constructor w/ int

Implement Member functions .cpp

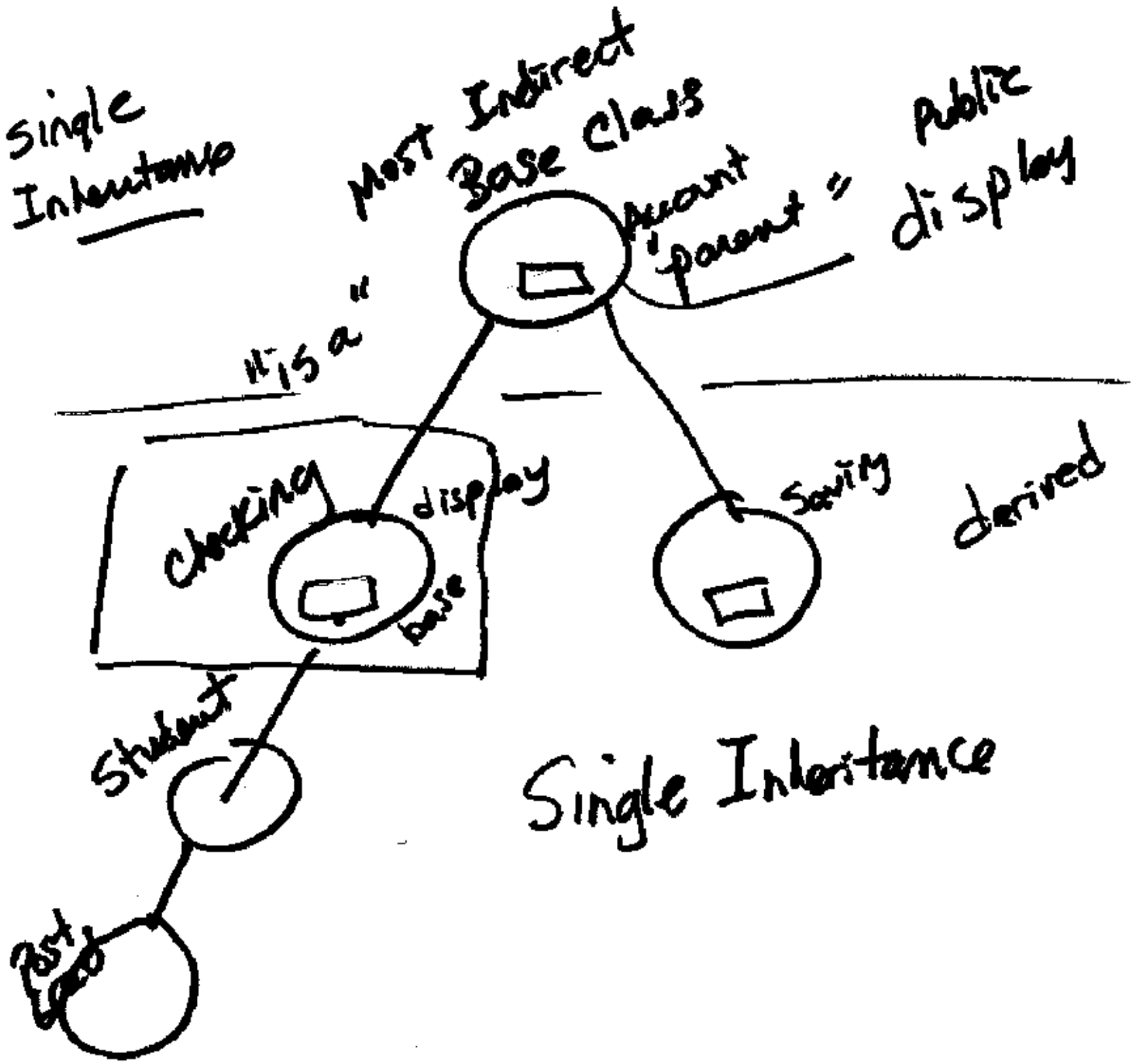
```
Acnd::Account(int value)
    {
    }
Saving::Savings(int value) : datamember(Φ),
                             Account(value)
  {
  }
```

initialization List

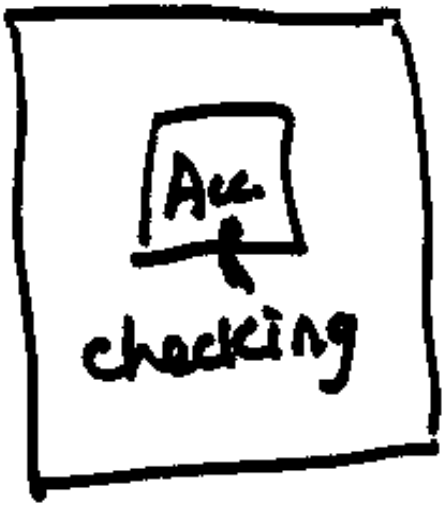
Saving obj(100);

```
Ax(100);
ata2(Φ);
lata1(Φ);
```

order is not important


Single Inheritance
Most Indirect Base class
Account "parent"
public display
"is a"
checking
display
base
Saving
derived
Student
Single Inheritance

checking obj;




```
class Account
{ public:
    Account();
   ~Account();
    void display();
protected: ~~private:~~
    client owner;
private:
    float balance;
};
```

"has a"

```
class savings
{  public:
        savings();
        savings(int);
        .
        .
        .
   protected:
        int datamember;
};
```

```
Savings::Savings(): datamember(0)
{
}
```

Task

person (responsible)
project
Due Date (Day/Time)
Priority

Email

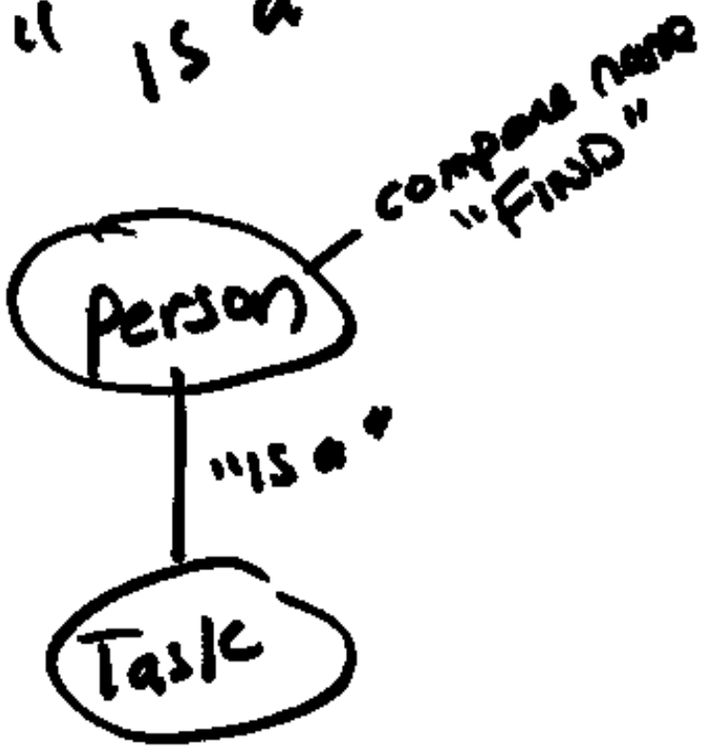
person (To) ← many
person (From) ← 1
" (cc, bcc) ← many

Message
Attachments
Date
Priority

Person

Name - First, Last
Email Address

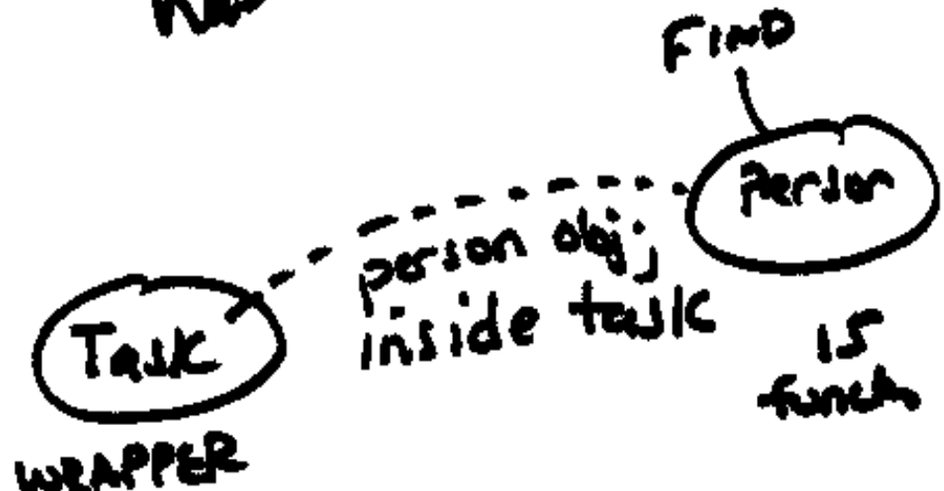
"is a"



List of Task

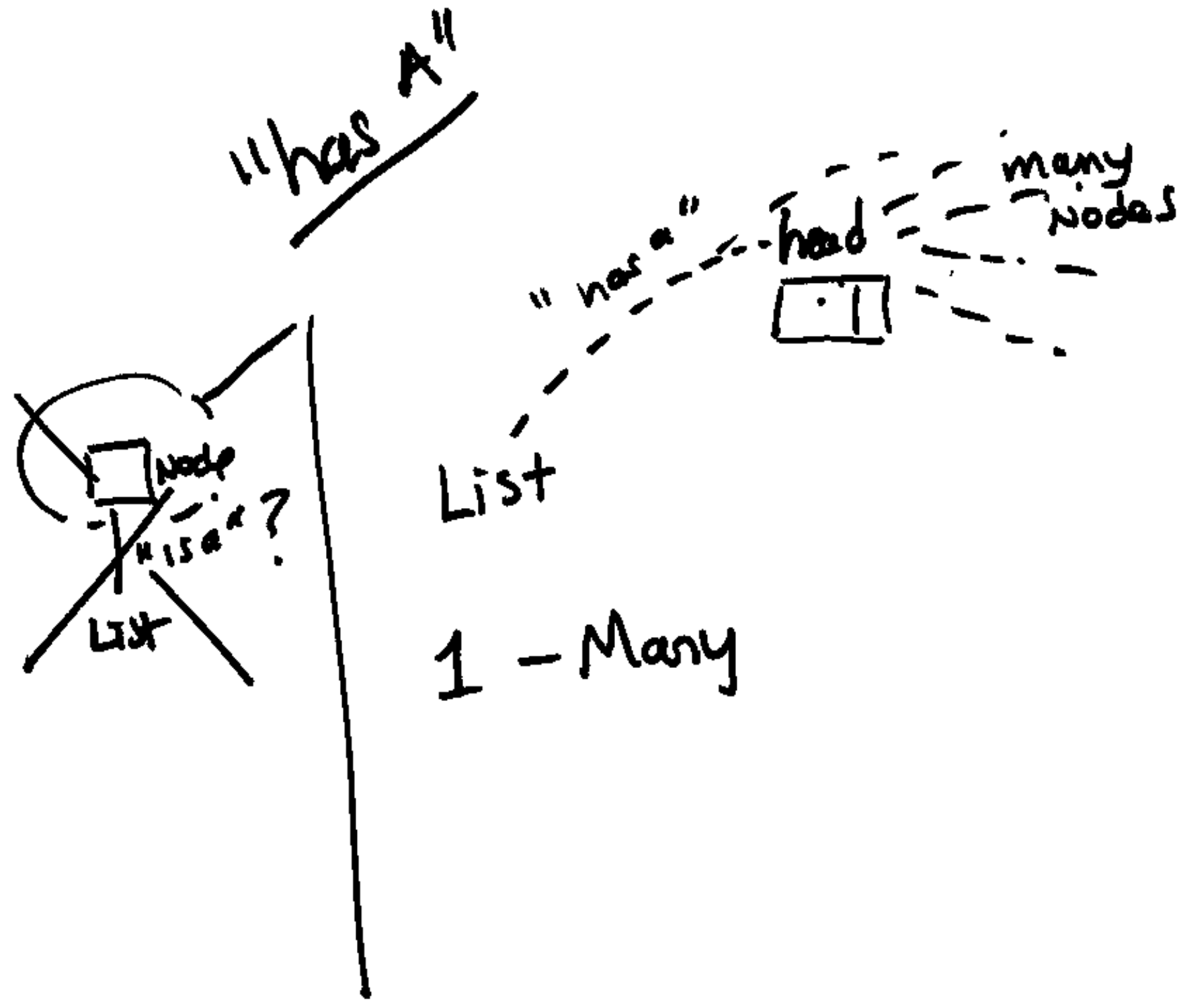
if(head->taskobj.Find("Joe Mike"))

"has a"

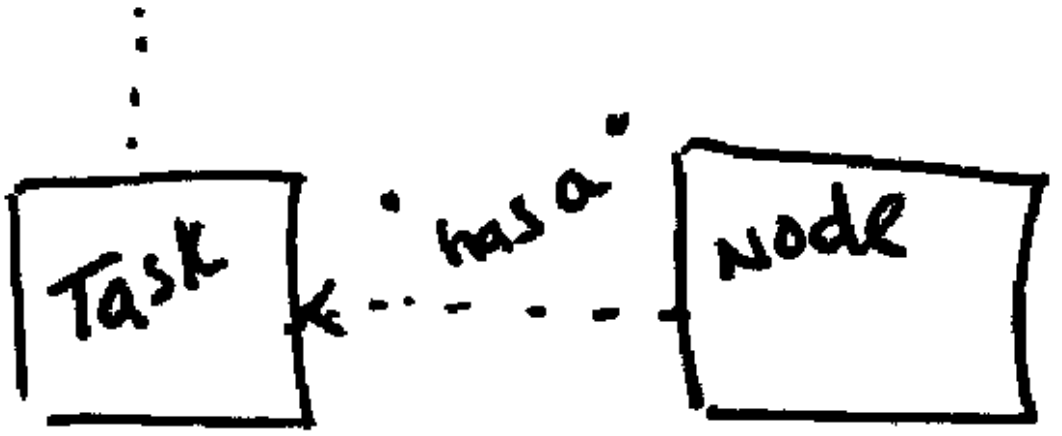


List of Task

if(head->taskobj.wrapper funtion())


"has A"
"has a"
head
many
Nodes
Node
"is a"?
List
List
1 - Many

Versus "has a"




```
struct node
{
    Task a_task;
    node * next;
};
```

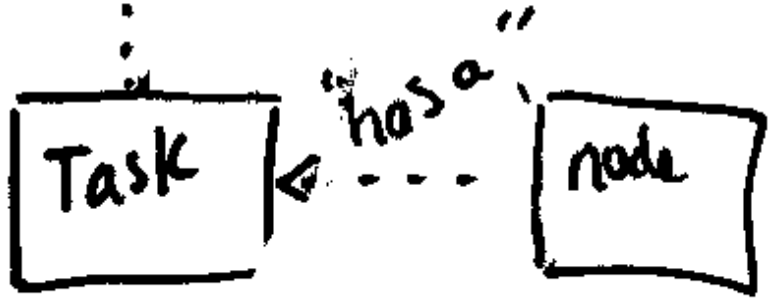
```
head -> a_task . compare( . . . )
```

or

```
head -> a_task . edit( . . . )
```

OR

Using classes




```
class node
{ public:
    node();
   ~node();
    int compare(...)
    int edit(...)
  private:
    Task a_task;
    node * next;
};
```

"wrapper" functions which in turn call the Task class functions thru the data members

head → compare(...)

which in turn will call

a_task.compare(...)

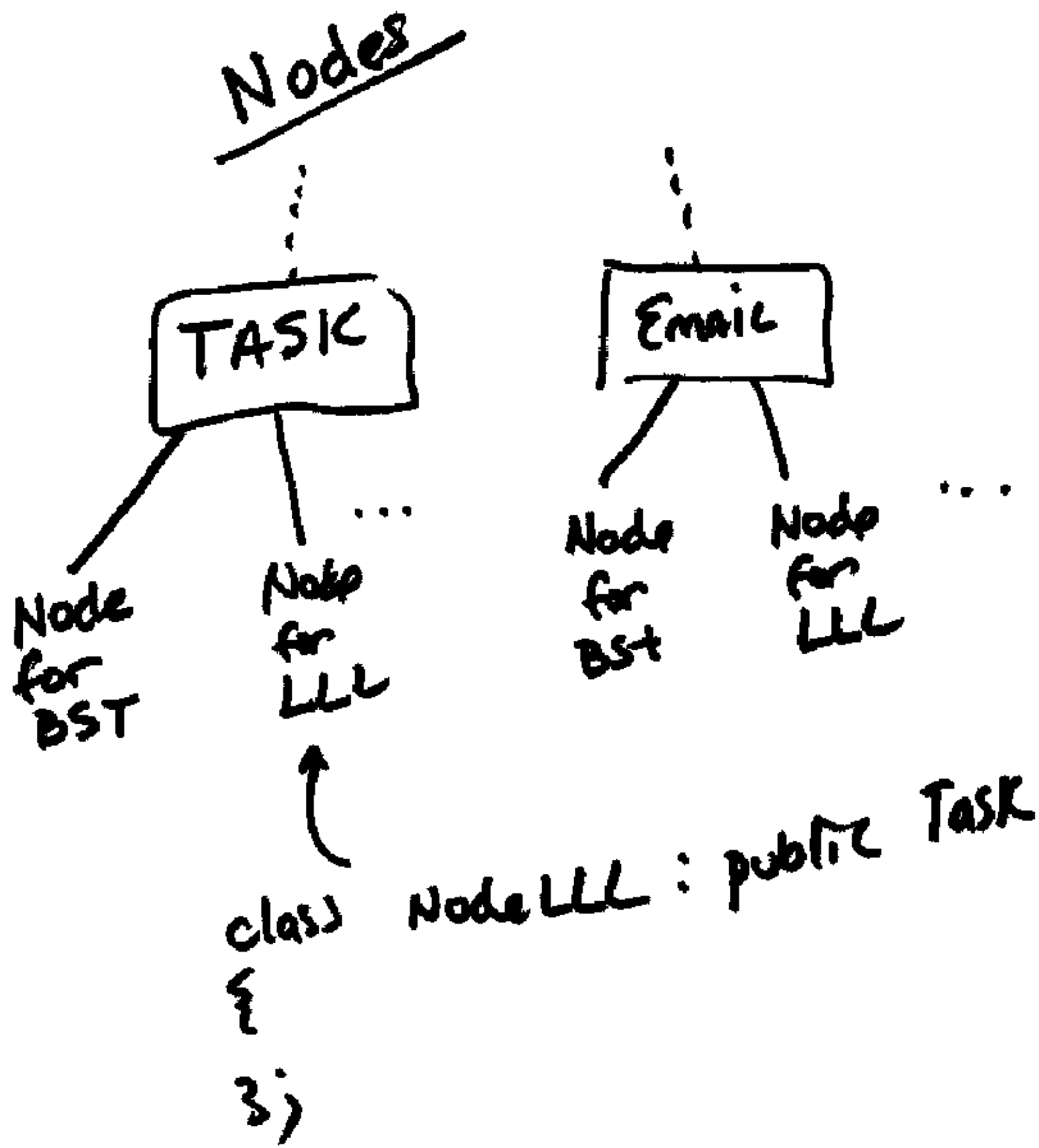
```
class common : public person
{


};
class task : public common
{   public:
        task();   ~task();
        int compare(task &);
        int restore();
        int edit....
};  private:  ← Members for a task

class node : public task
{   public:
        node();
        ~node();
        void setnext(node * ptr);
        node * getnext();
};  private:
        node * next;
```

client:

head -> compare(...);


Nodes
TASK
Email
Node for BST
Node for LLL
...
Node for BST
Node for LLL
...
class NodeLLL : public Task
{
};

Advantage:

Client program (TaskList)

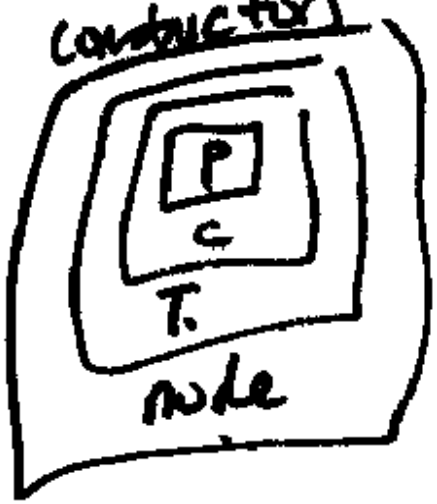
```
node * head;
head = new node;
```

Implicitly calls 3/4 constructor

```
head->compare(...)
```

or

```
head->edit(...)
```



NOT IN The NODE Class ! IN The hierarchy.

SEAMLESS