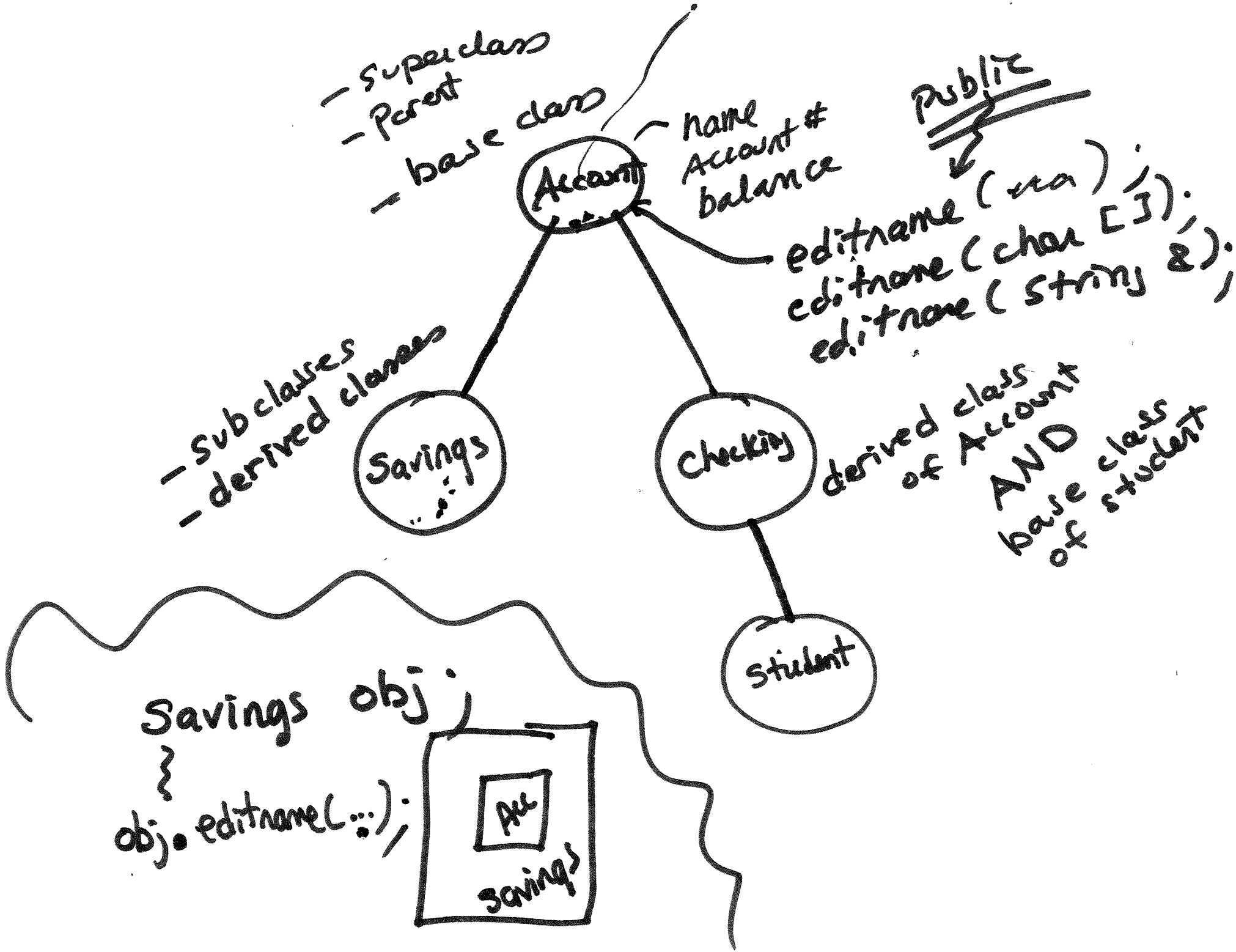
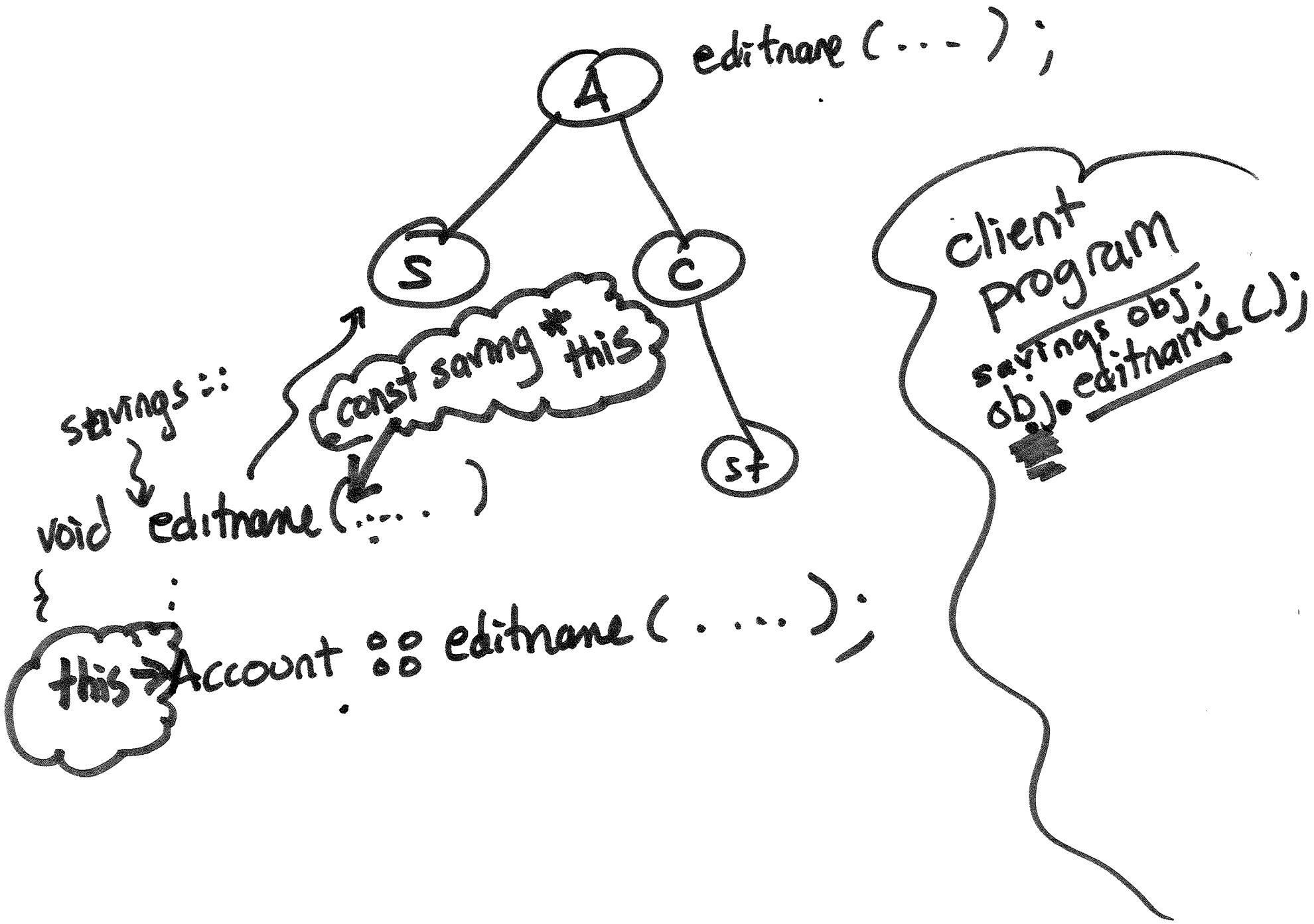

- superclass
- Parent
- base class
Account
- name
Account #
balance
Public
editname (...);
editname (char [ ]);
editname (string &);
- subclasses
- derived classes
Savings
Checking
derived class of Account AND base class of student
student
Savings obj;
{
obj.editname(...);
Acc
Savings


editname(...);
A
S
C
St
const saving *this
savings::
void editname(....)
{
this->Account::editname(....);
client program
savings obj;
obj.editname();

without hierarchy

void saving::
editname() {
personal_info.editname()
}

Savings

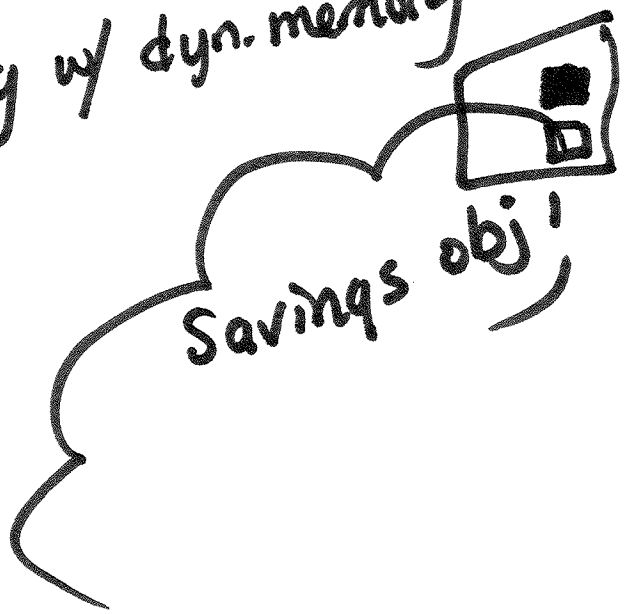
Account
- name
- address
- passwd

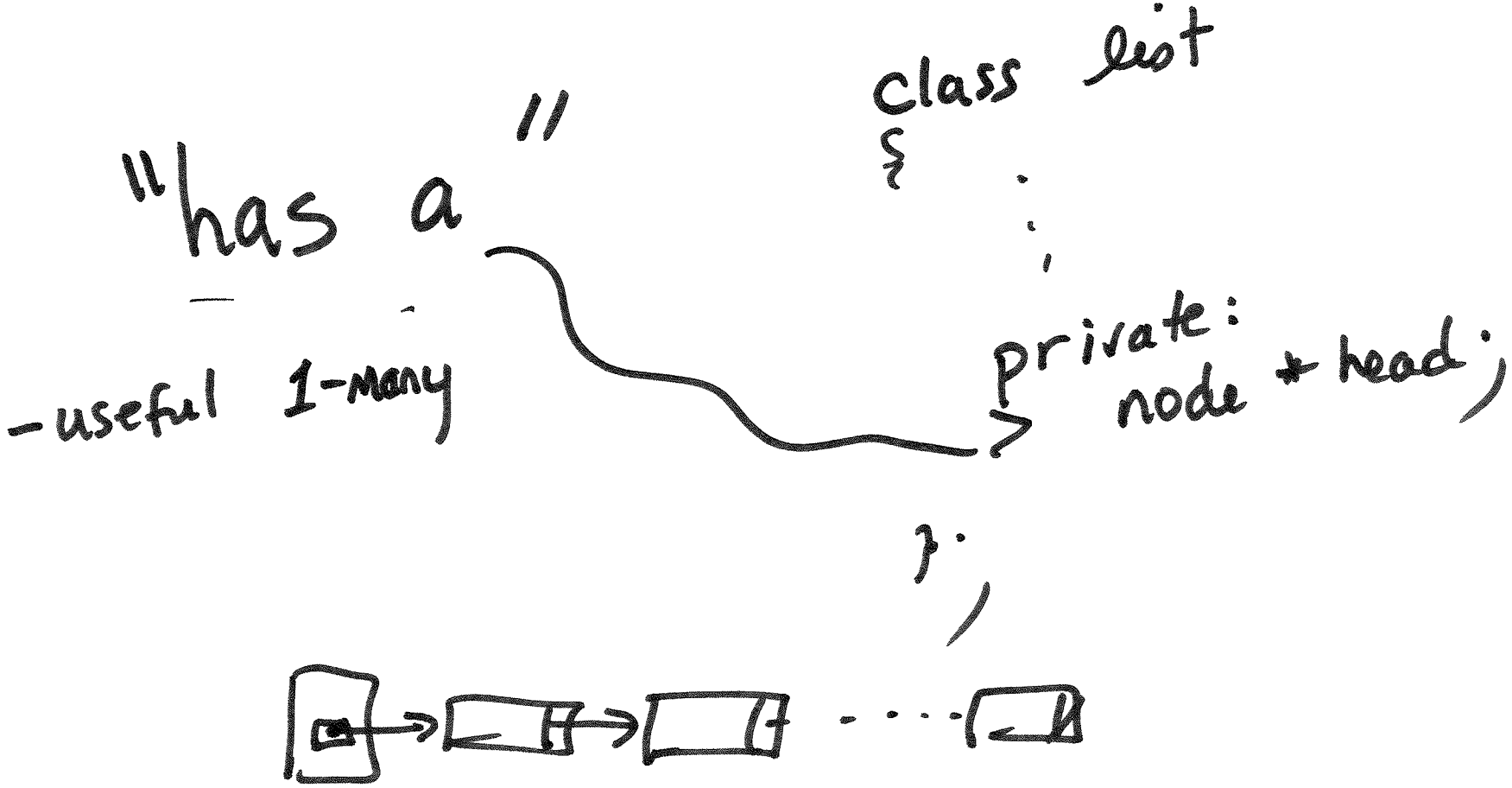
setname()
editname()
outputname()
copyname()
compname ...

Class savings
{ public:
saving(); //Always!
~saving(); // when working w/ dyn. memory
void editname();

private:
Account personal_info;
float balance;

};




"has a"
class list
{
private:
node * head;
};

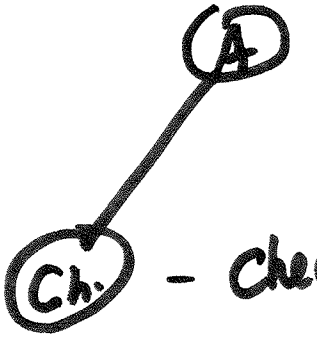
"is a"

Syntax

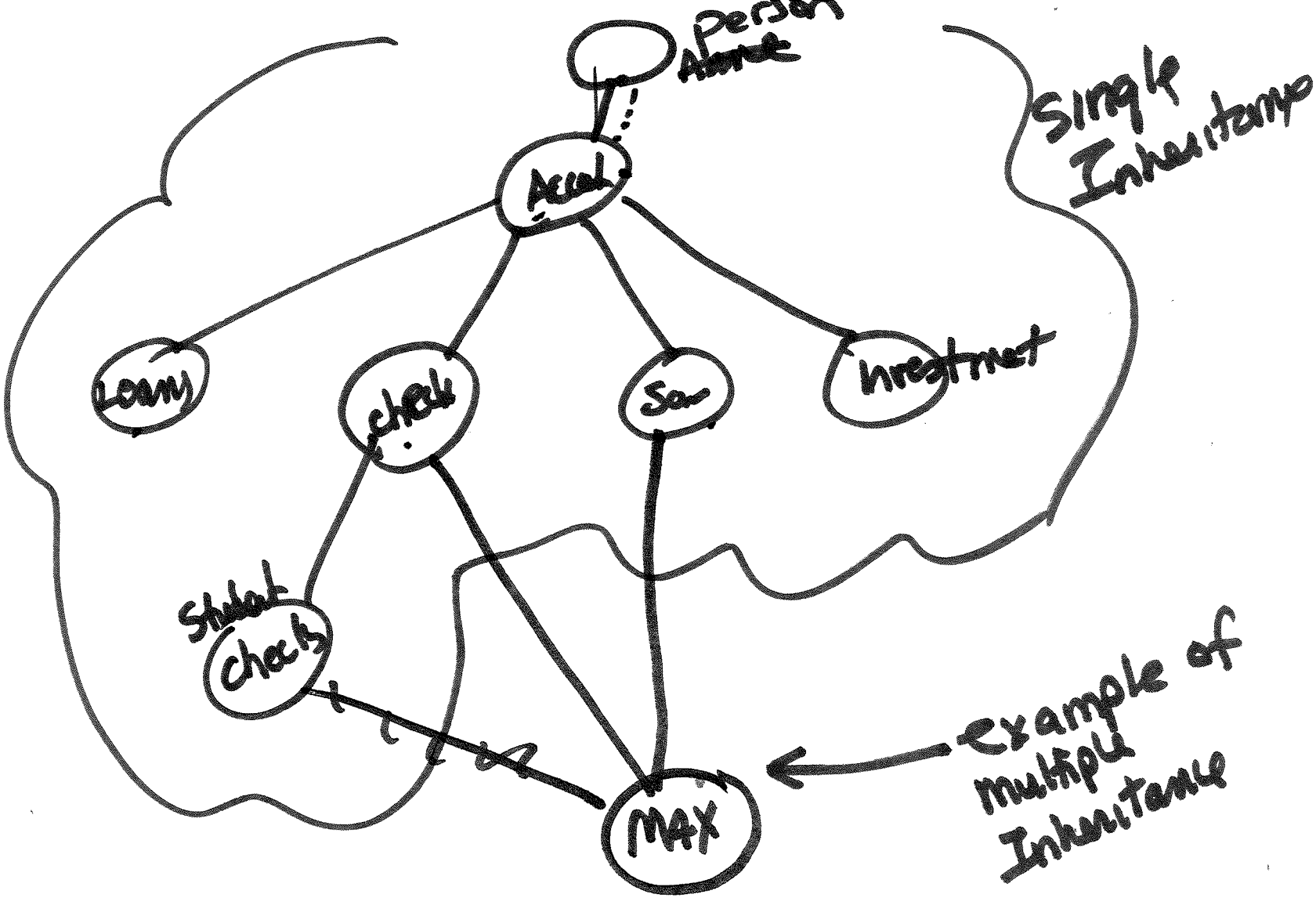
single inheritance

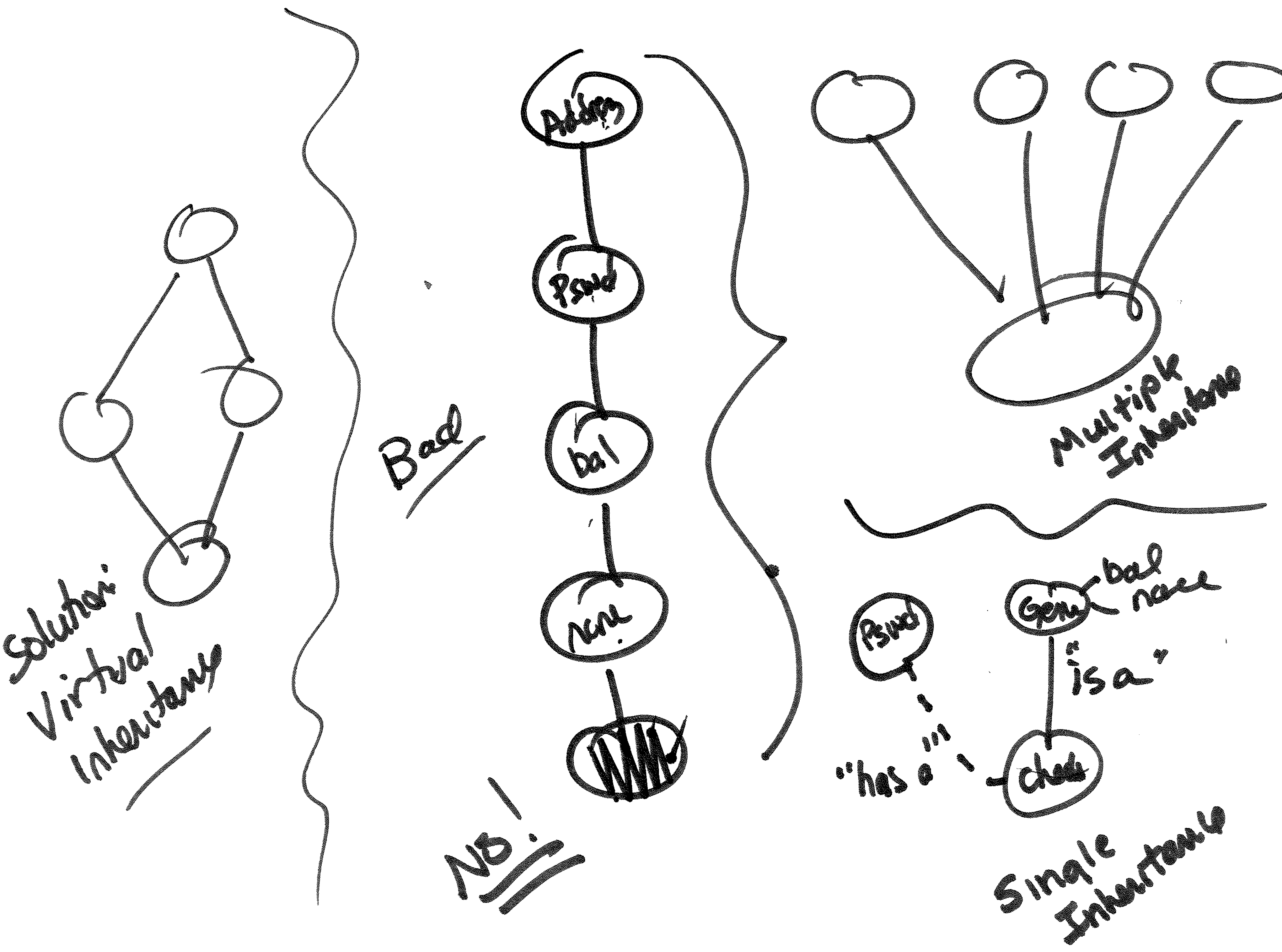
class checking : public class Account
{ ⋮ //class interface
};

checking obj;



- checking **is** an account object plus more
- checking **doesn't** micromanage account
- wherever an account is needed a checking object can be used in its place.
- ~~xxxx~~ chars **From** private ⟹ to protected


Person
Single Inheritance
Loan
check
Saw
Investment
Student check
MAX
example of multiple Inheritance


Multiple Inheritance
bal
name
"is a"
"has a"
Single Inheritance
NO!
Bad
Solution: Virtual Inheritance