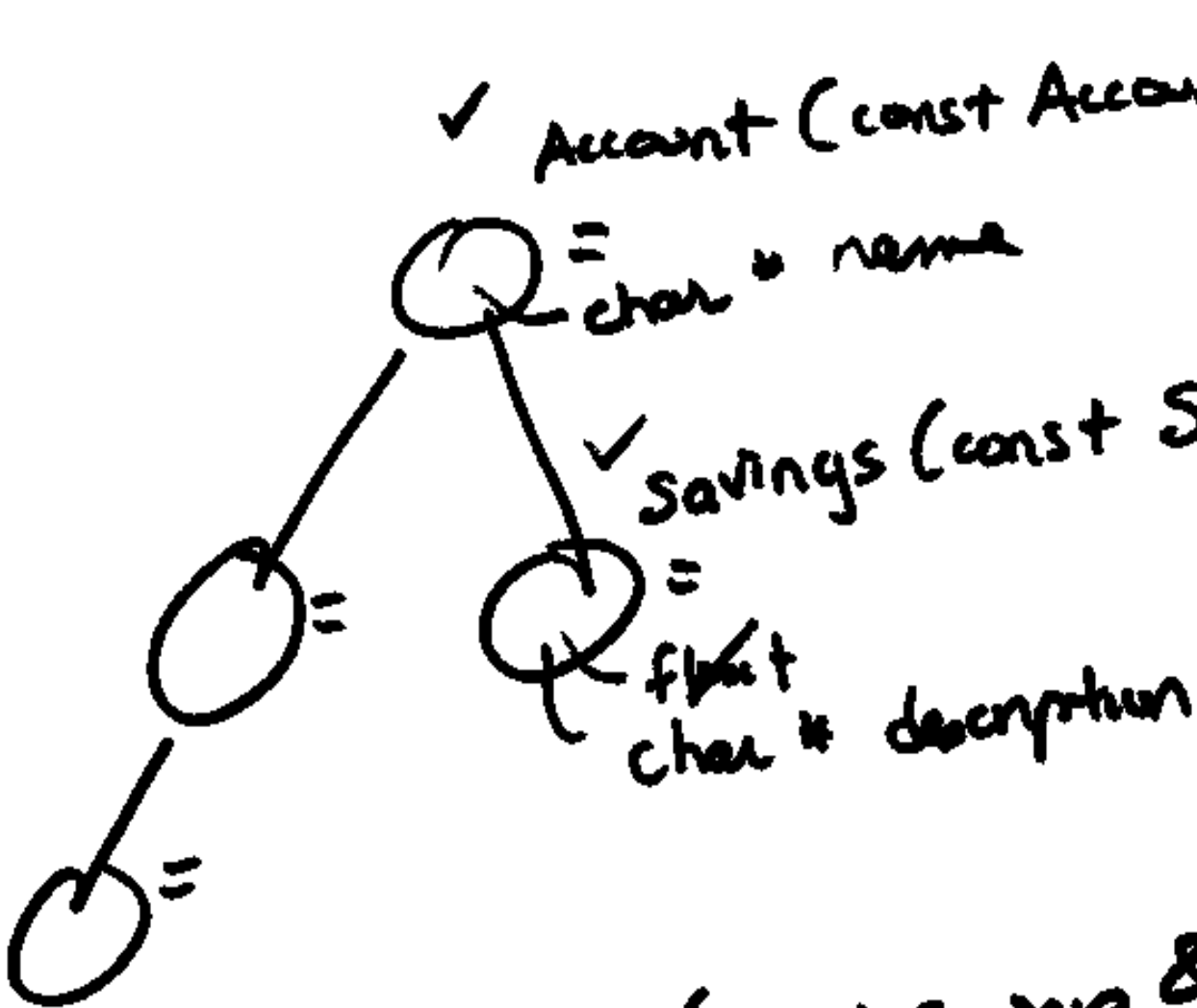
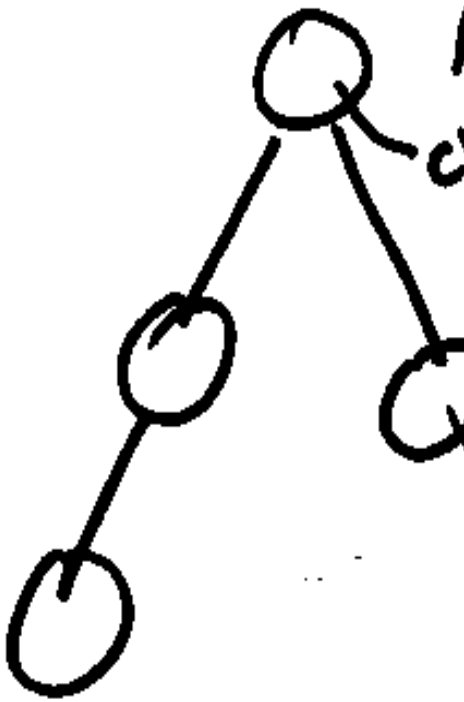
TODAY

- Operator Overloading as it relates to
  1. Inheritance
  2. Dynamic Binding
- User Defined Type Conversions
- Exception Handling
- PREP for MIDTERM!



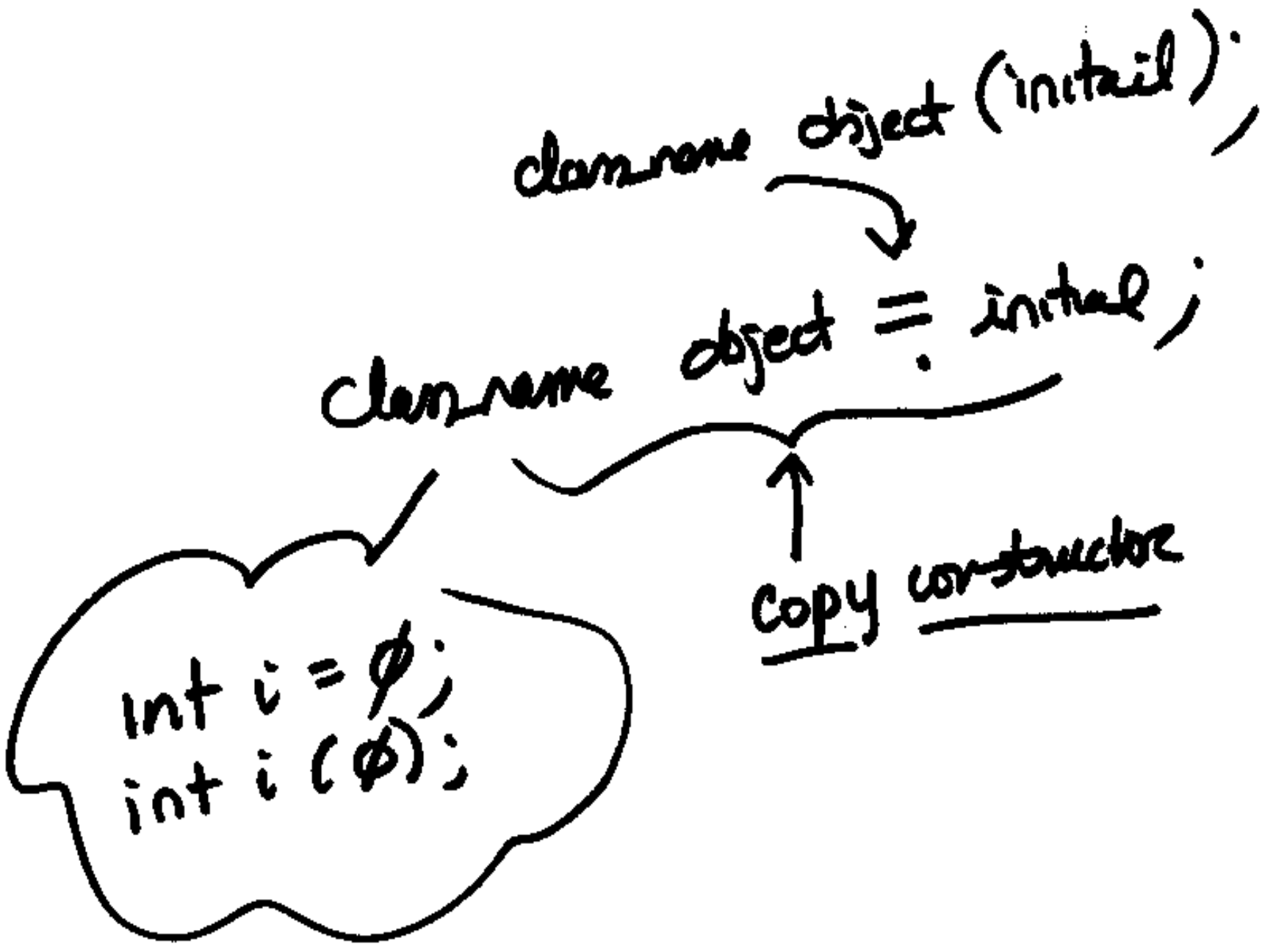

```
Savings::Savings(const Savings & source) : Account(source)
{
    description = new char[strlen(source.description) + 1];
    strcpy(description, source.description);
}
```



Account & operator= (const Account &);
char * name

Savings& operator= (const Savings &);
char * description;

```
Savings& savings:: operator= (const savings & src)
{
    Account:: operator= (src);
    .
    .
    .
or  static_cast<Account &>(*this) = s;
```

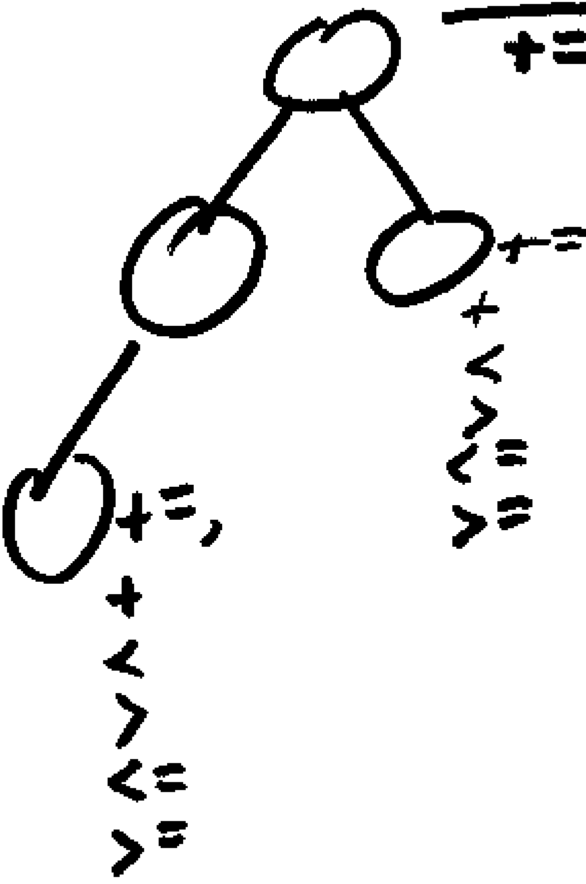

class_name object (initial);
class_name object = initial;
Copy constructor
int i = 0;
int i (0);

Members

+=

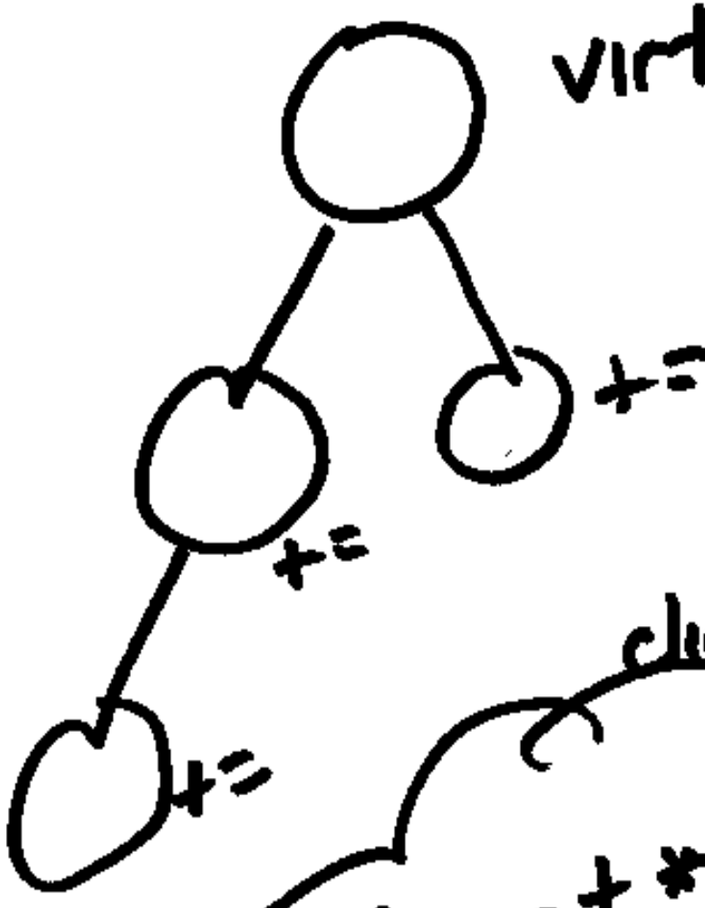
friend

+



With dynamic binding

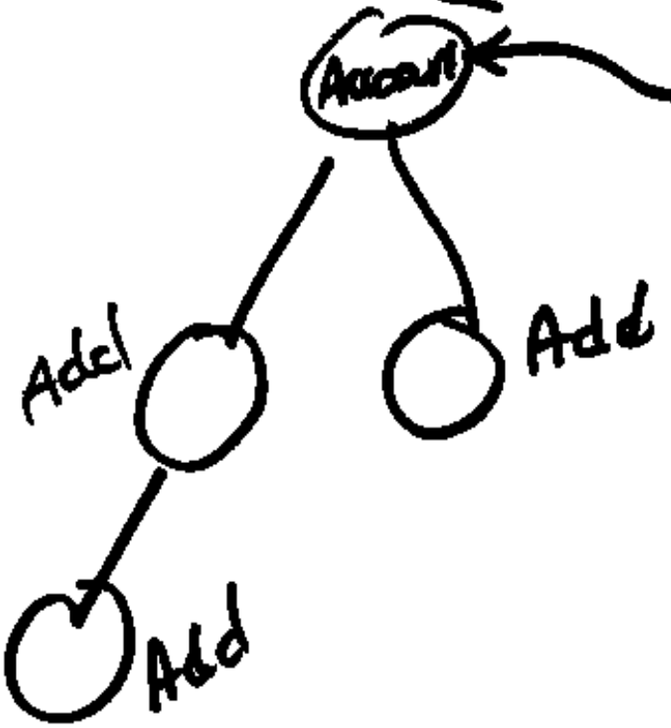
virtual Account& operator += (const Account&);



client

```
Account * ptr;
ptr = &savingobj;
}
ptr += obj;
```

friend Account operator+(const char *, const Account &);



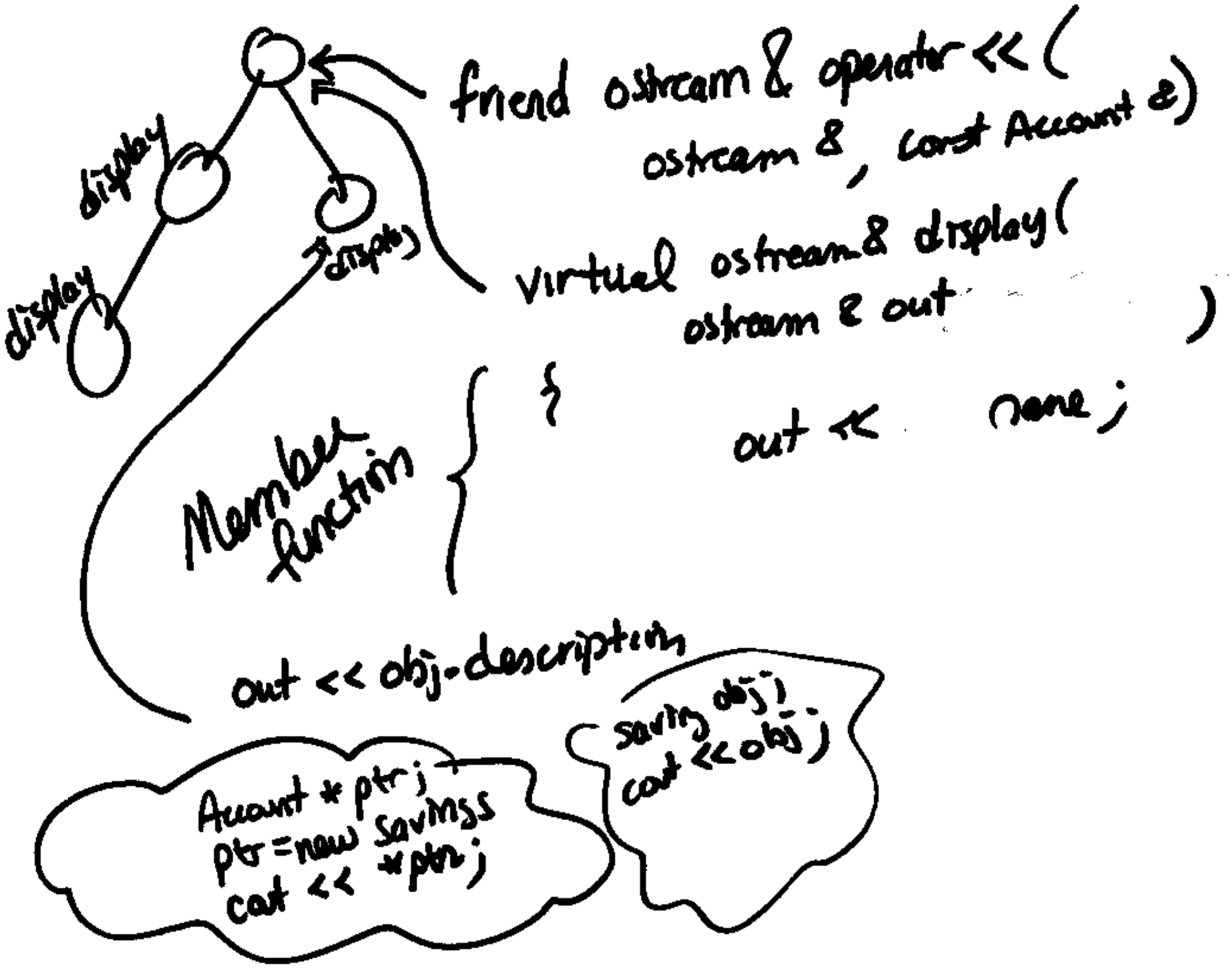
① Friends CANNOT be Virtual

② "Virtual HELPER MEMBER FUNCTION"

"Miller" + *ptr

IN a Larger context

```
Account operator+ (const char* str,
                   const Account& op2)
{
    Account* temp = new ____
    ~~temp->Str, op2)~~
    temp->Add(str, op2)
    return *temp)
}
```


friend ostream & operator << (
ostream &, const Account &)
virtual ostream& display(
ostream & out
)
{
out << name;
display
display
display
display
Member function
out << obj-description
Account * ptr;
ptr = new Savings
cout << *ptr;
saving obj;
cout << obj;

```
ostream & operator << (
        ostream & out,
        const Account & obj)
{ return obj.display(out);
```