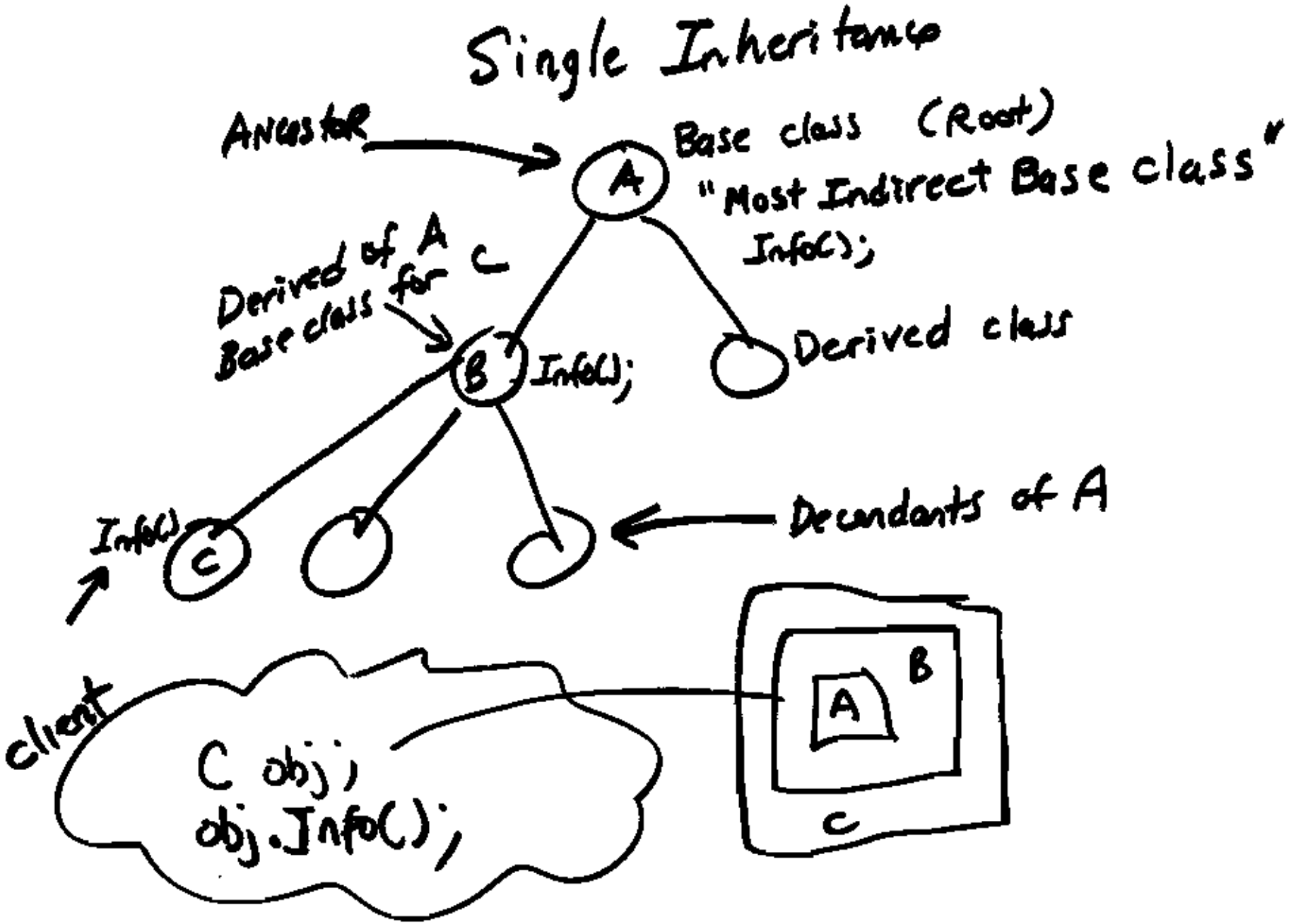
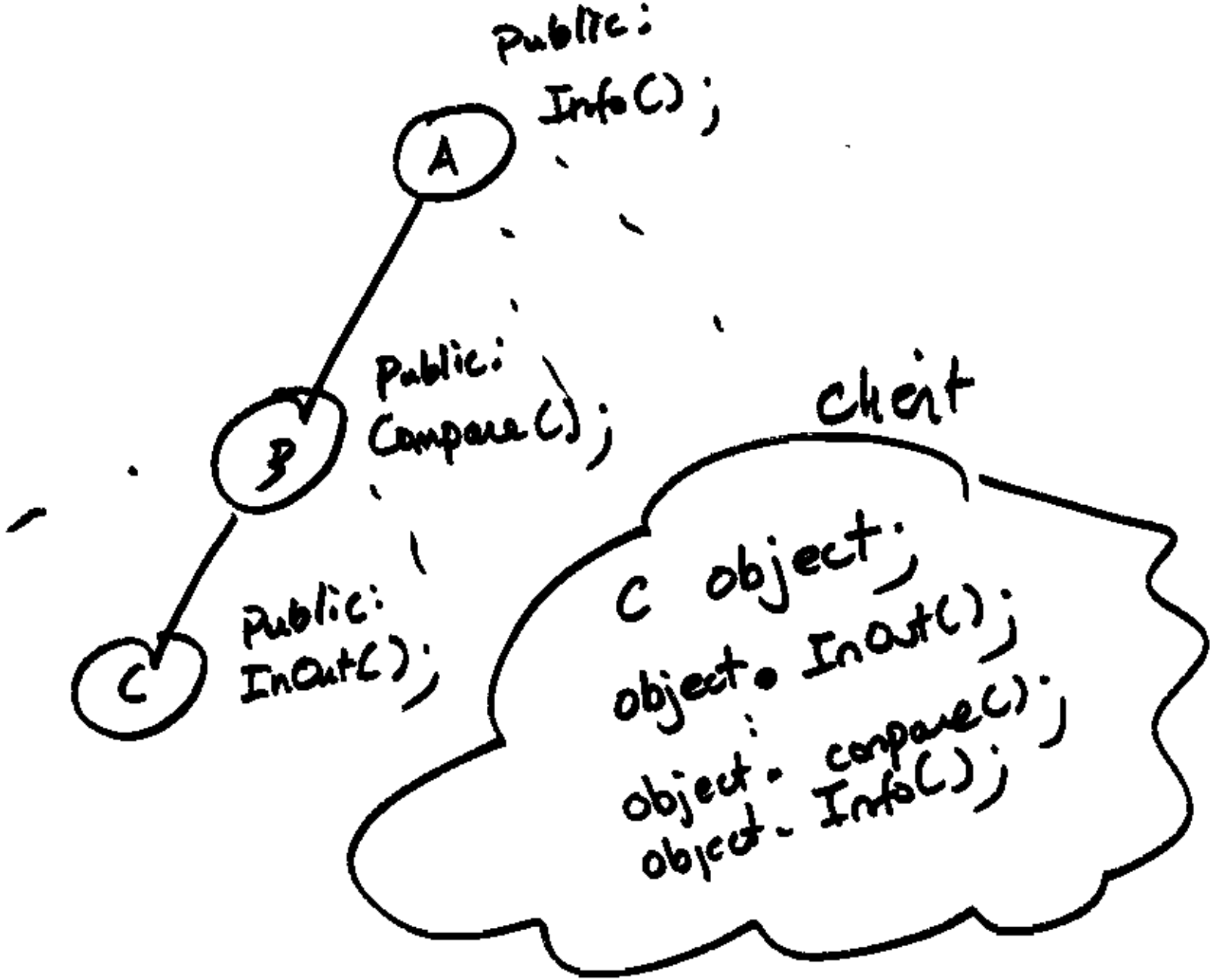
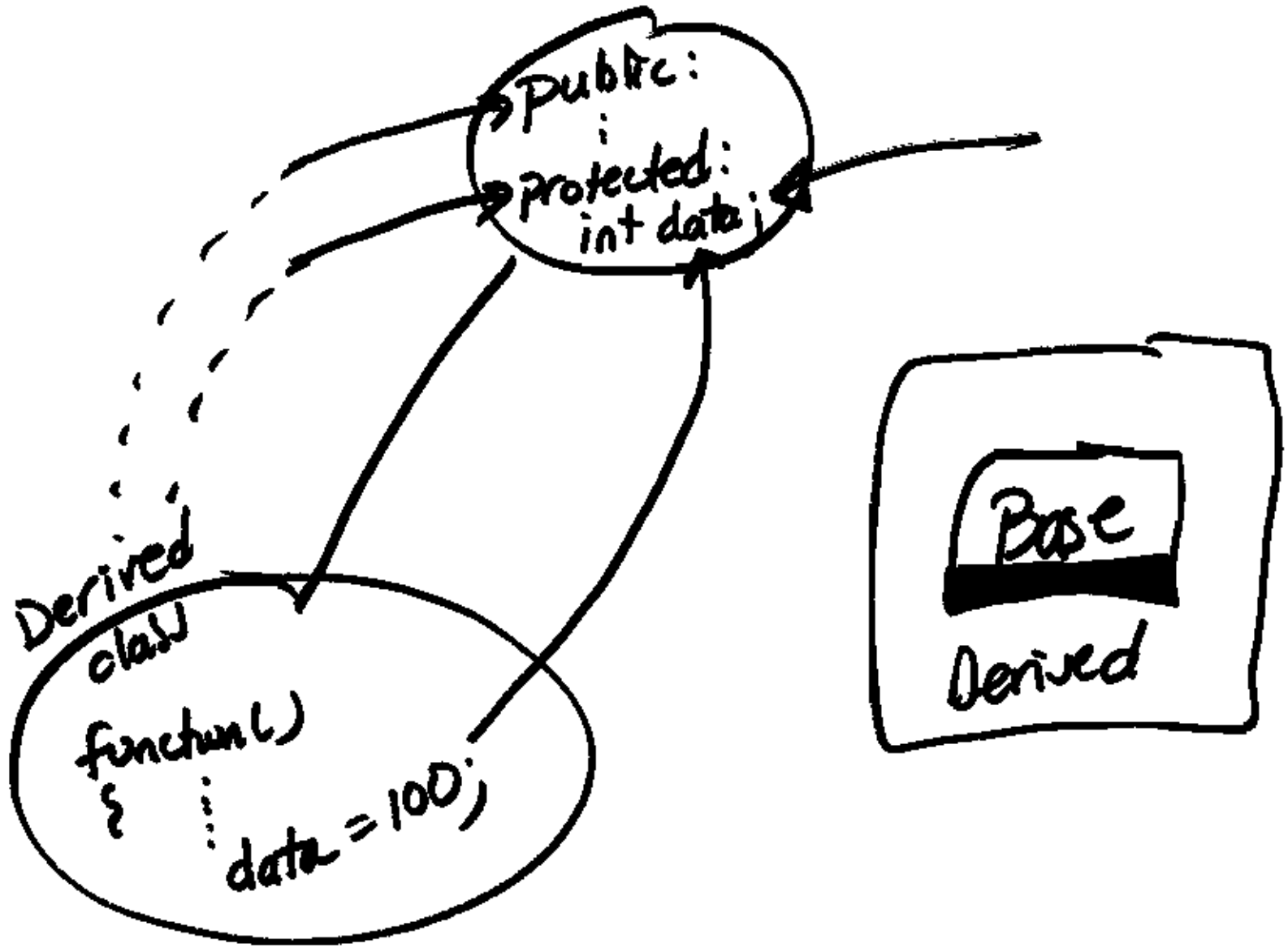

Single Inheritance
Ancestor
A
Base class (Root)
"Most Indirect Base class"
Info();
Derived of A
Base class for C
B
Info();
Derived class
Info()
C
Decendants of A
client
C obj;
obj.Info();
A
B
C

```
void C::Info()
{
    B::Info();      // OK
    A::Info();      // Breaks OOP Rules
```


Public:
Info();
A
Public:
Compare();
B
Public:
InOut();
C
client
C object;
object. InOut();
object. compare();
object. Info();


public:
protected:
int data;
Derived
class
function()
{
data = 100;
Base
Derived

```
base::base()
{ :
}
```

```
derived::derived(char[])
{ :
}
```

name

```
derived obj("filename");
```

```
base::base()
{
  //Called!
}
```

```
base::base(char fname[])
{
    // NOT USED!
}
```

```
derived::derived(char name[])
{
}
```

// Still Calls Default base! Automatically

client

```
derived obj("filename");
```

```
base :: base ()
{
    // NOT called
}
```

```
base :: base (char fname[3])
{
    // Used
}
```

```
derived :: derived (char name[])
  : base (name)
    //initialization List
{
    :
}
```

client

```
derived obj ("filename");
```

.h

```
class derived : public base
{   public:
        derived();  // default
        derived(char [I]);

    protected:
        const int size;
        node * root;
        :
};
```

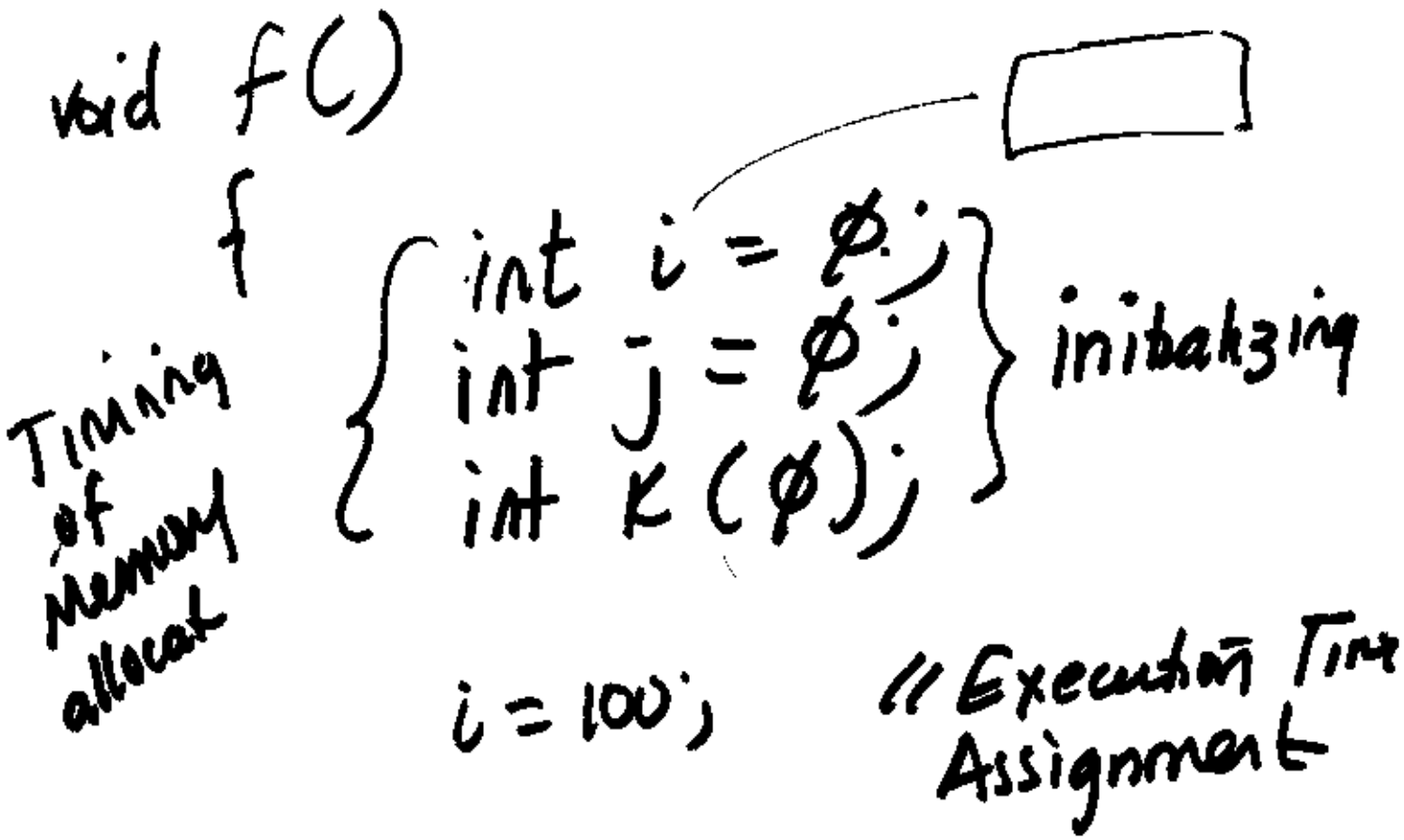
INITIALIZATION LIST

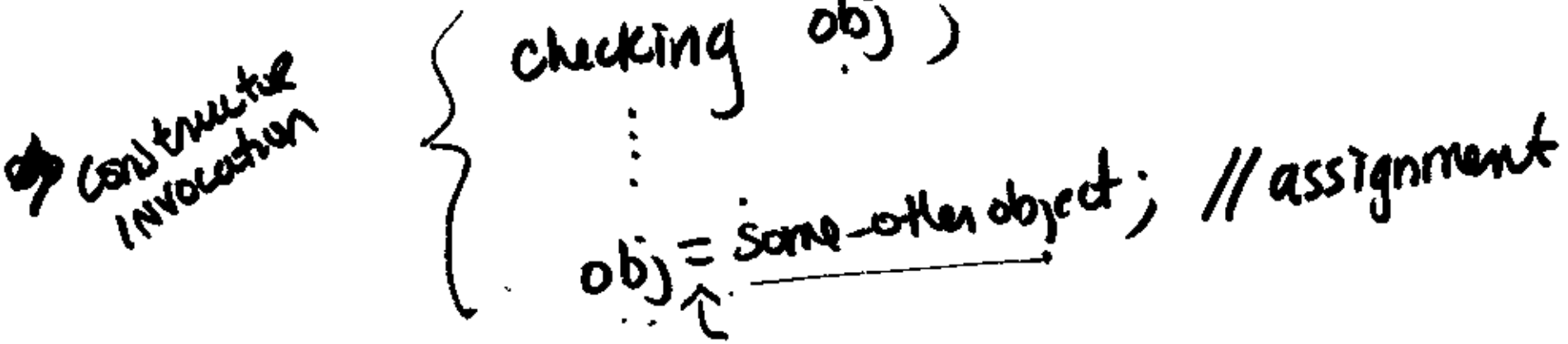
.cpp

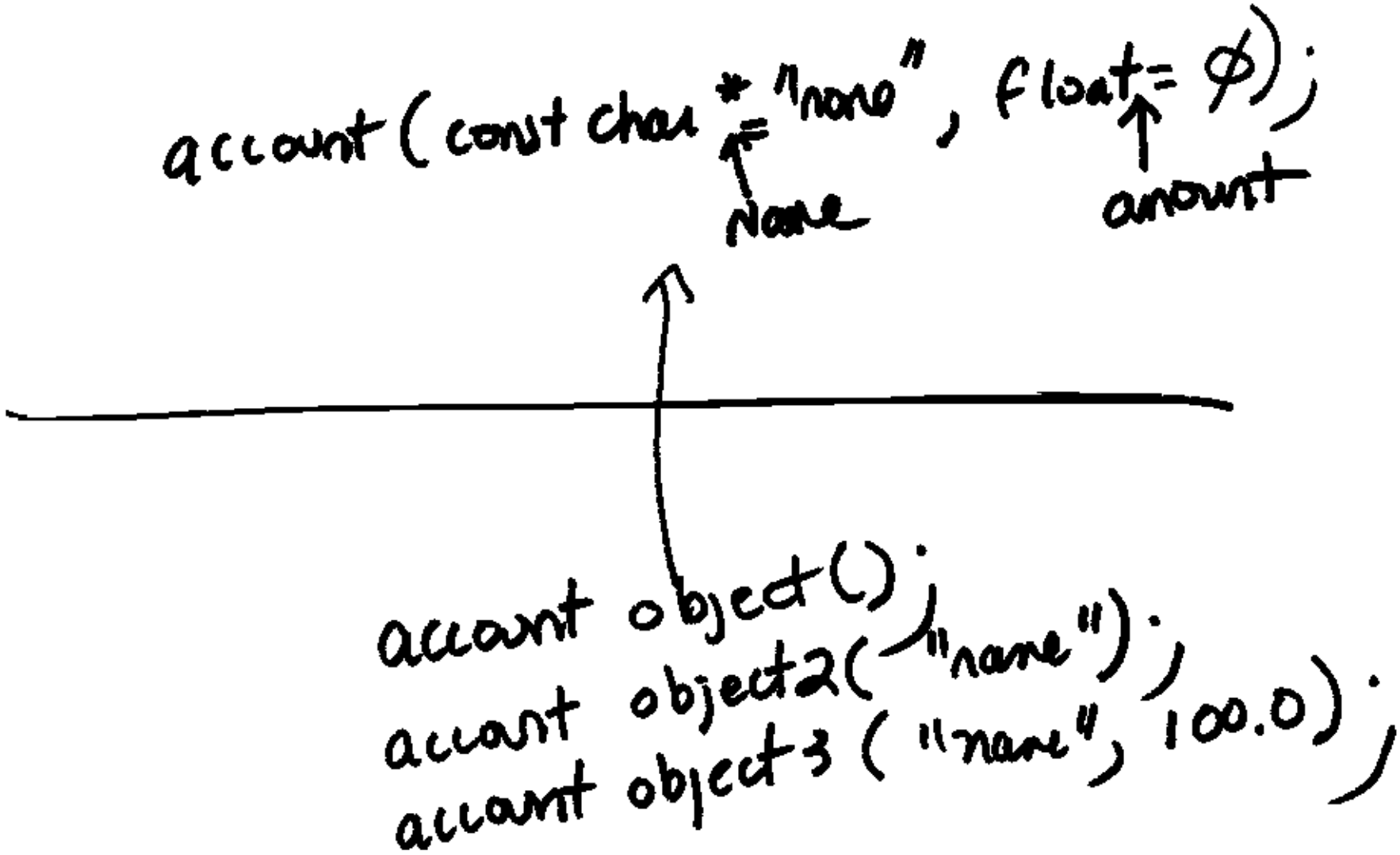
```
derived :: derived() : size(100), root(0)
{                                      ↑
                                       NULL
}
```

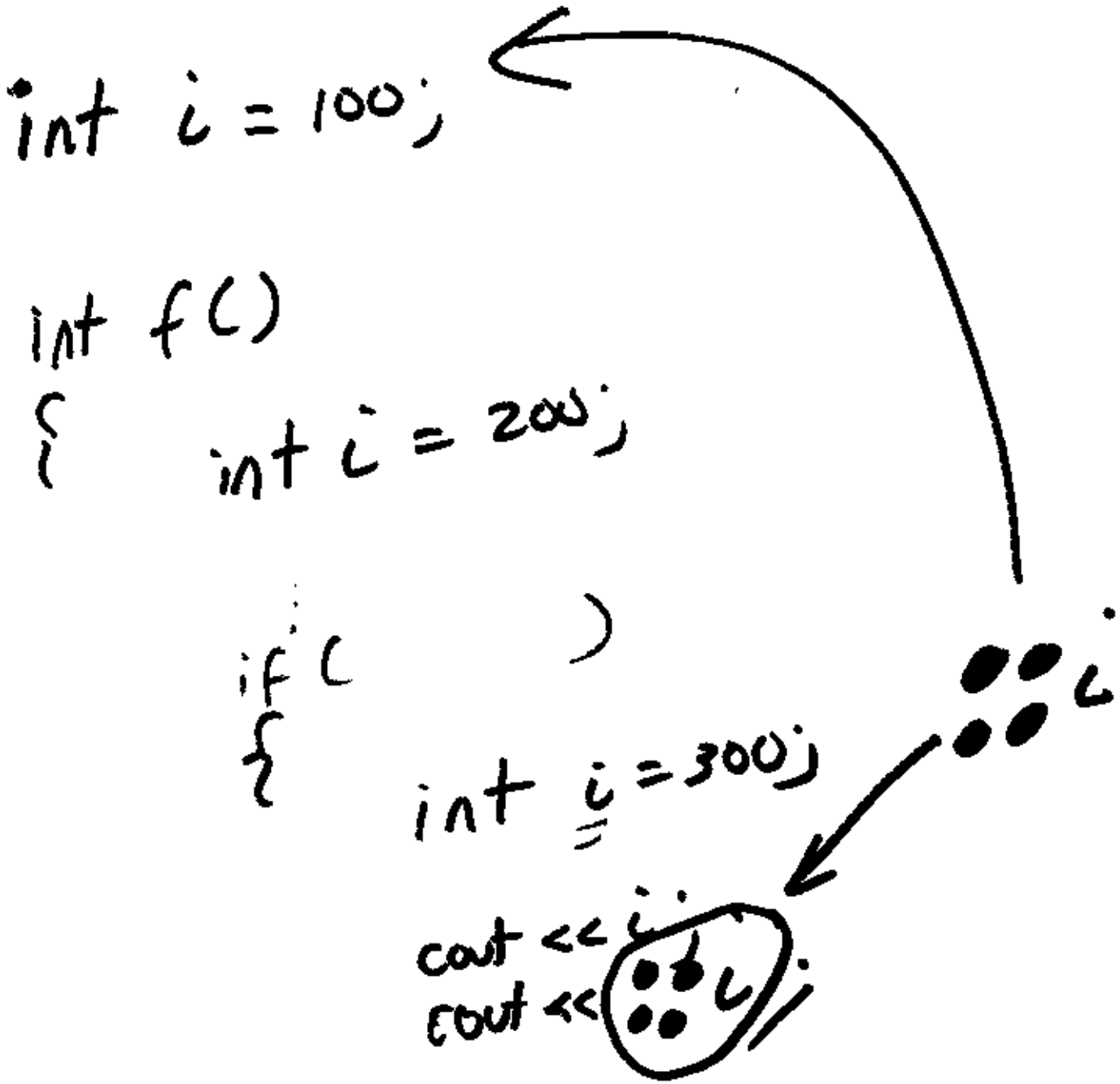
.cpp

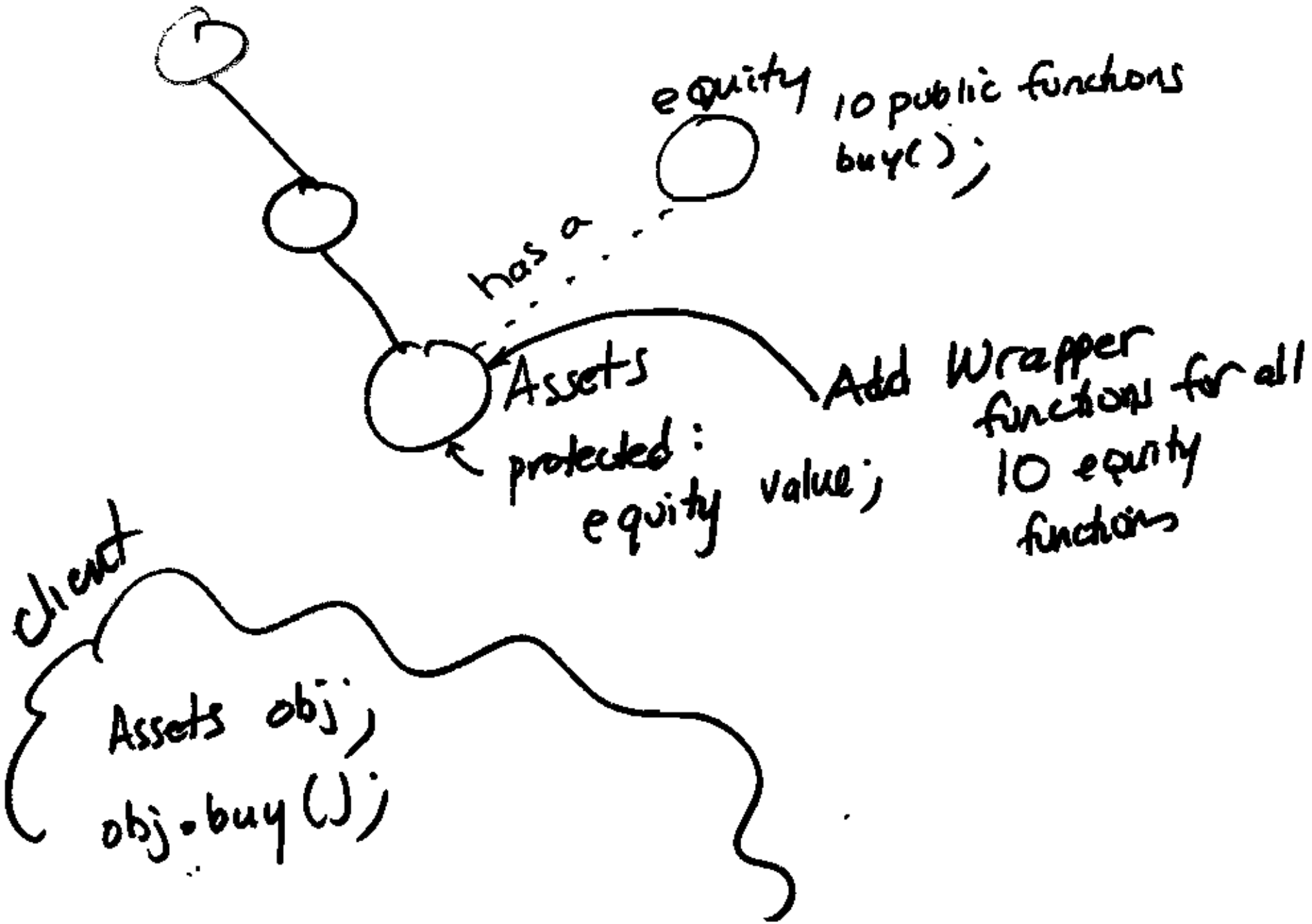
```
derived :: derived (char name[]) :
                root(0), size(100), base(name)
{
}
```

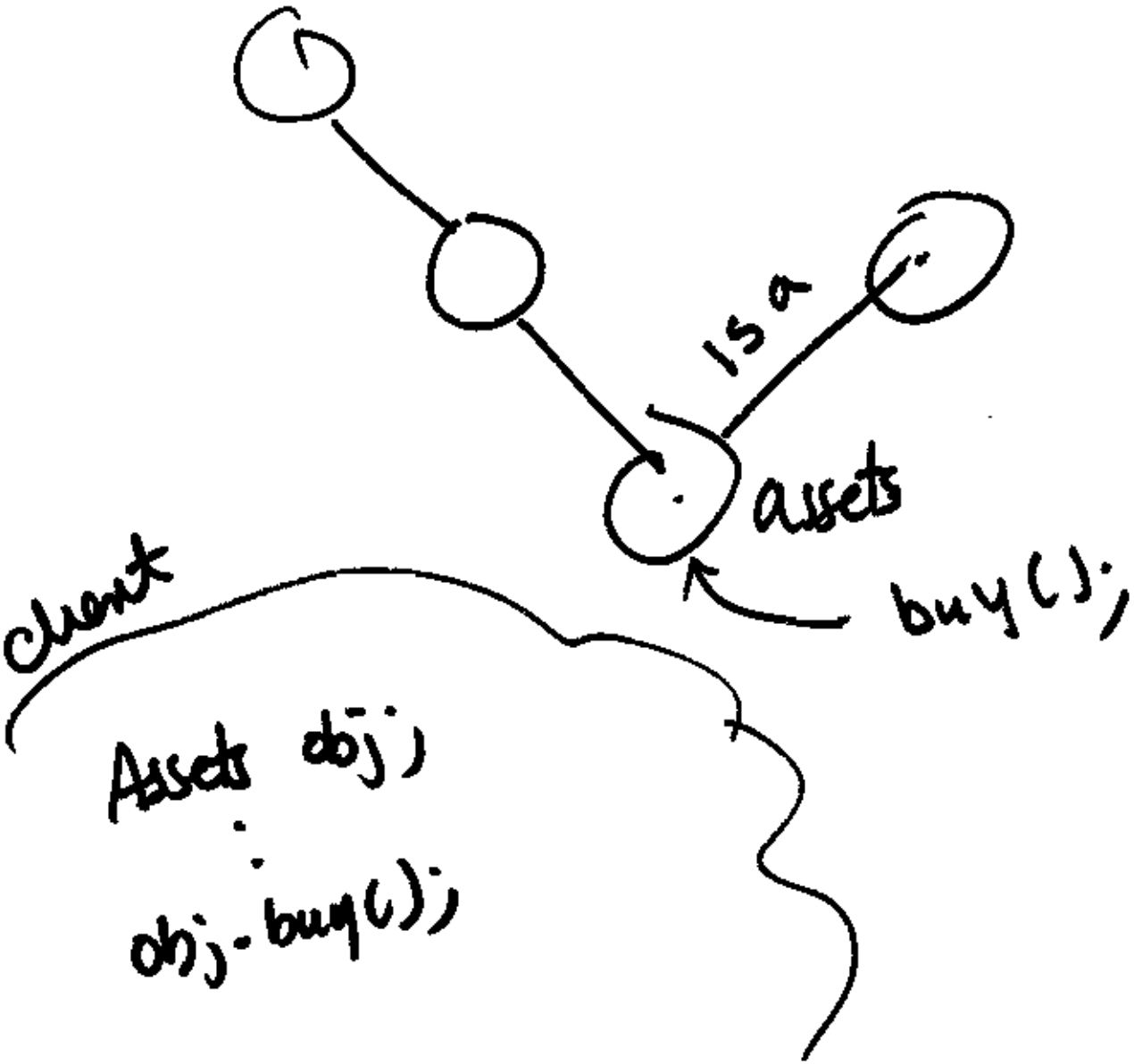

void f()
{
int i = 0;
int j = 0;
int K(0);
initializing
Timing of Memory allocat
i = 100;
// Execution Time Assignment

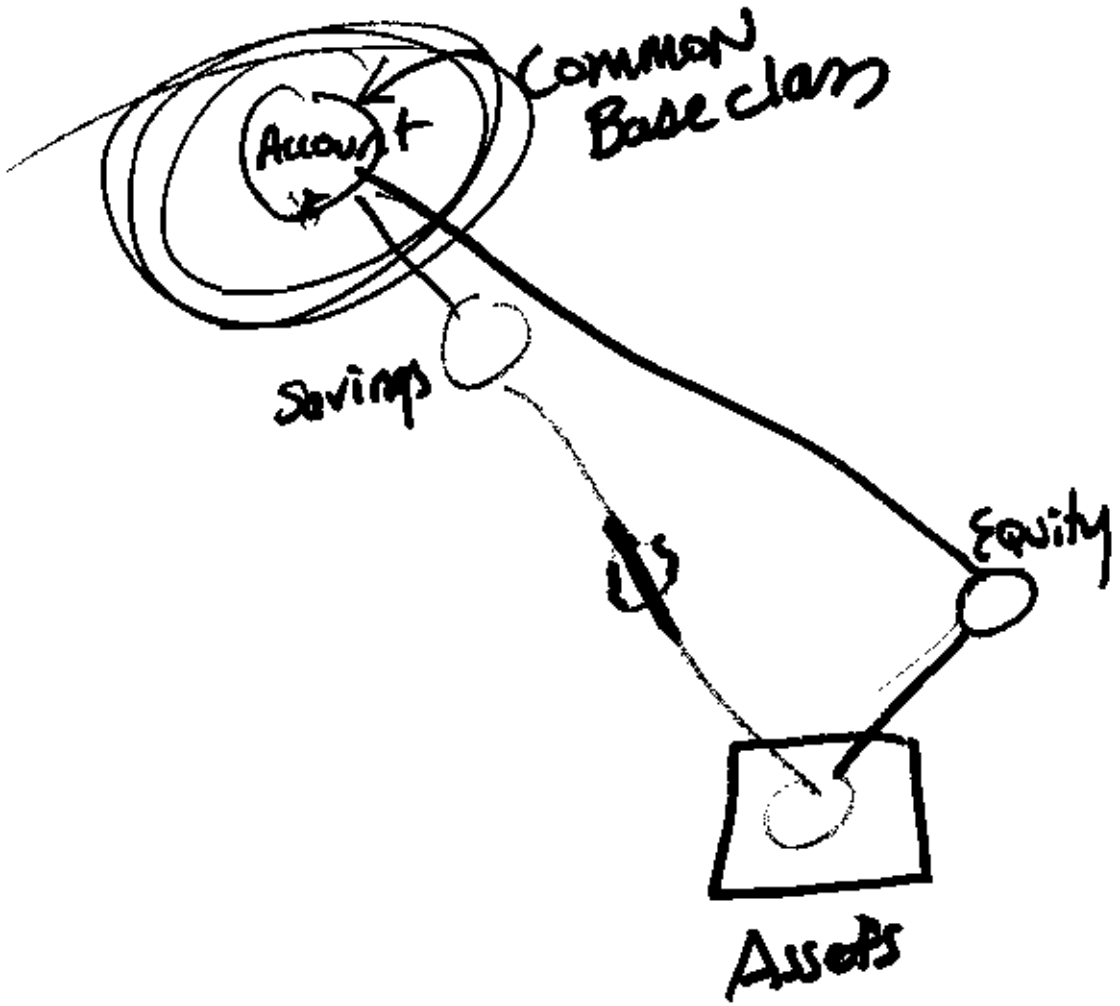

Constructor invocation
checking obj)
obj = some-other object ; // assignment

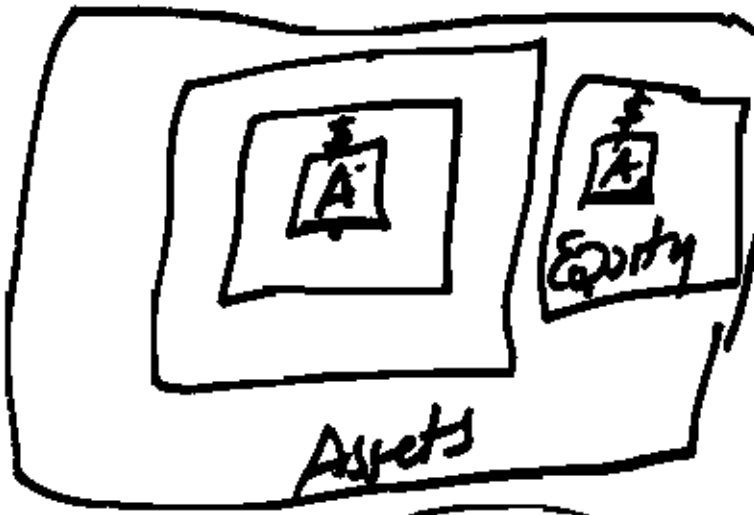

account(const char *="none", float= 0);
name
amount
account object();
account object2("name");
account object3("name", 100.0);


int i = 100;
int f()
{
int i = 200;
if (
)
{
int i = 300;
cout <<
cout <<
i
i
i


equity
10 public functions
buy();
has a
Assets
protected:
equity value;
Add Wrapper functions for all 10 equity functions
client
Assets obj;
obj.buy();


is a
assets
buy();
client
Assets obj;
obj.buy();





Virtual Inheritance

Virtual Inheritance

```
class Account { ... };
class Savings : virtual public Account { ... };
class Equity : virtual public Account { ... };
class Assets : public Savings, public Equity { ... };
```

Dynamic Binding