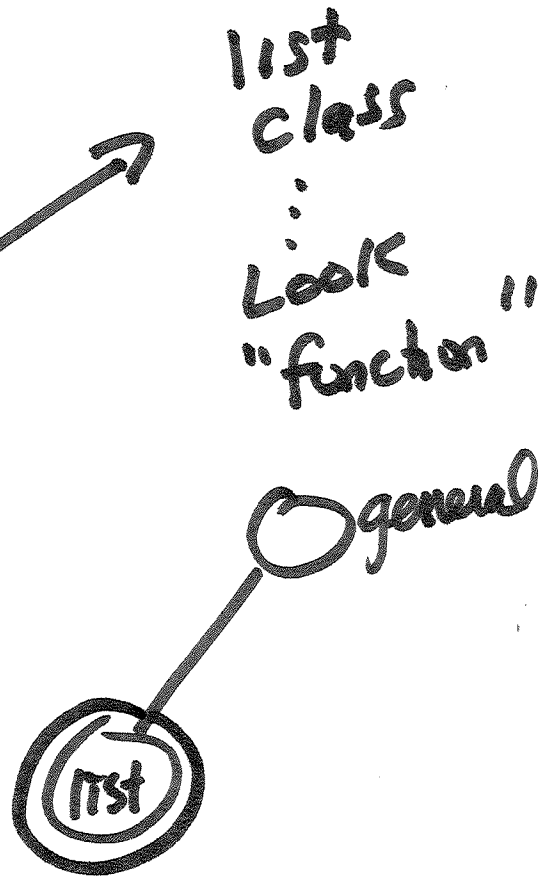
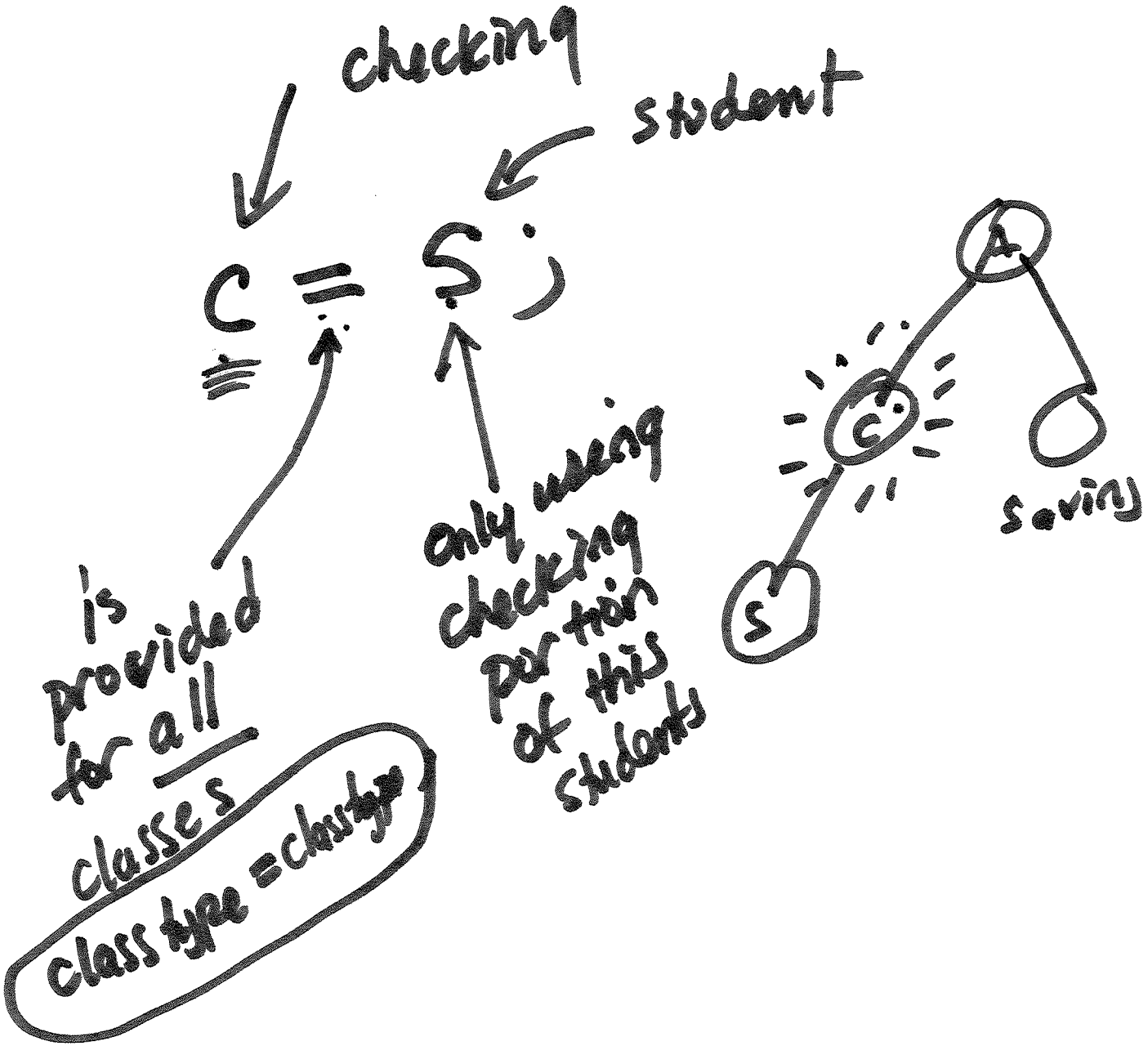
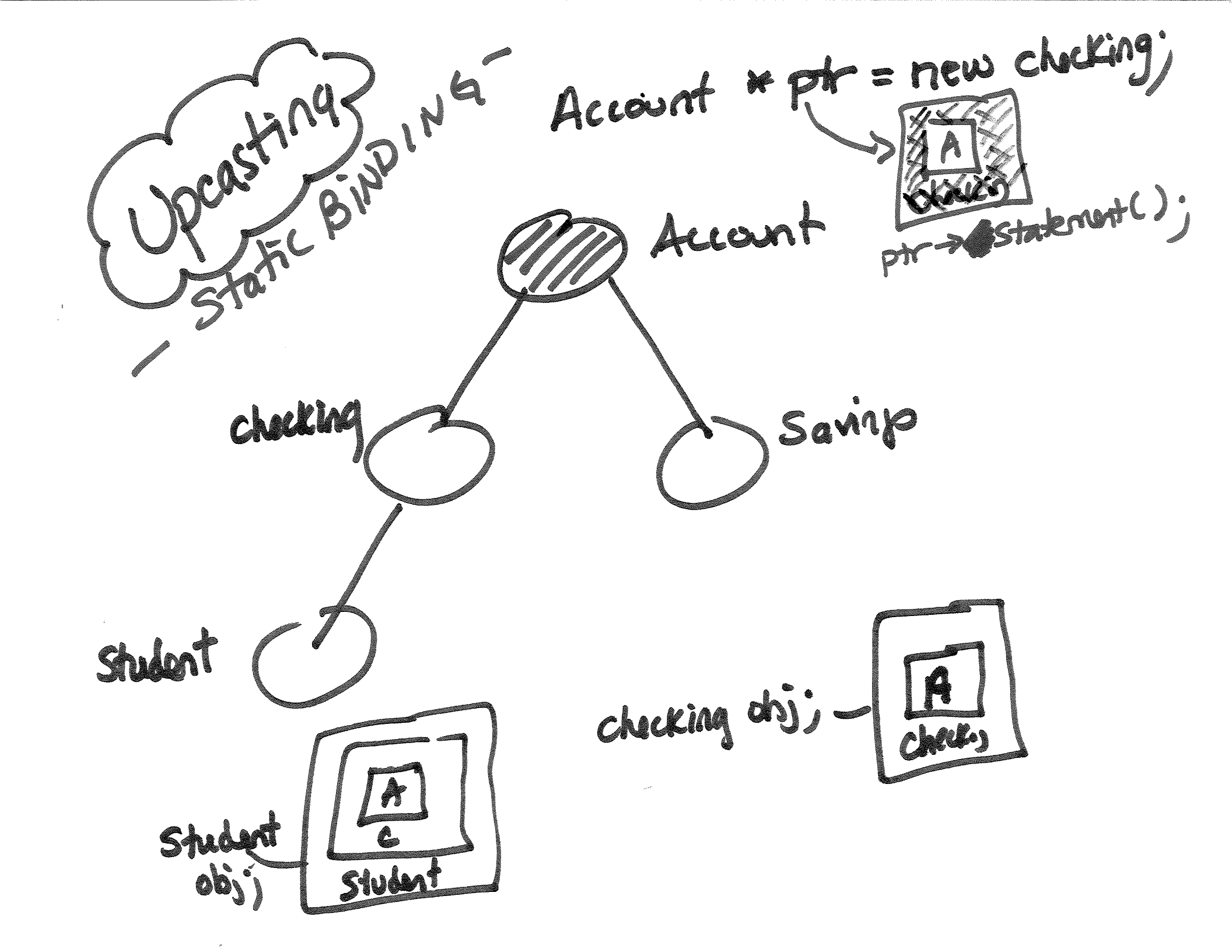
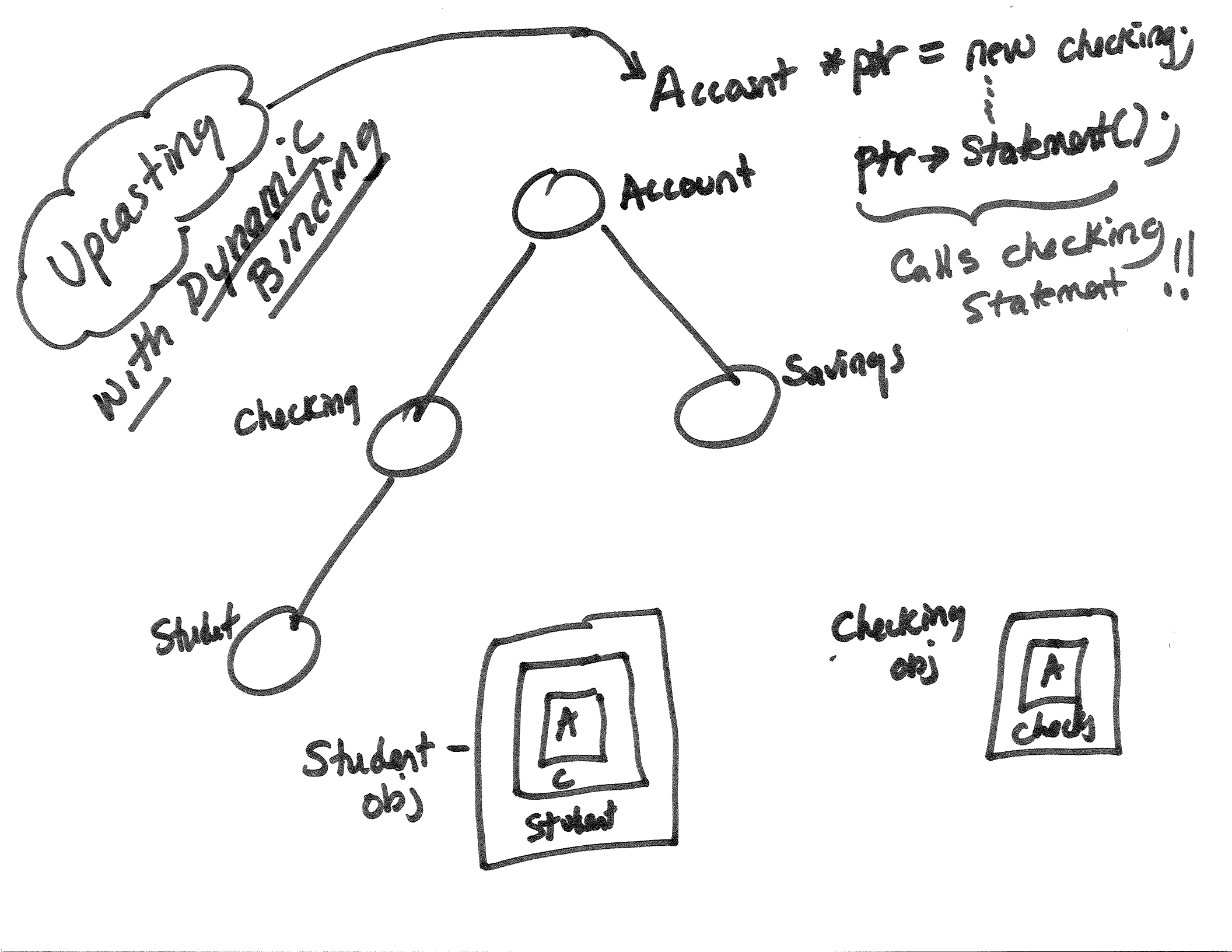
Static Binding

```
list obj;
{
obj.function(...);
```




checking
student
C
S
is provided for all classes
only using checking portion of this students
A
C
S
saving
class type = class type


Upcasting
Static BINDING
Account *ptr = new checking;
A
checking
ptr
statement();
Account
checking
Savings
Student
Student obj;
A
C
Student
checking obj;
A
checking


Upcasting
With Dynamic Binding
Account *ptr = new checking;
ptr -> Statement();
Calls checking Statement !!
Account
Checking
Savings
Student
Student obj
A
C
Student
Checking obj
A
Checks

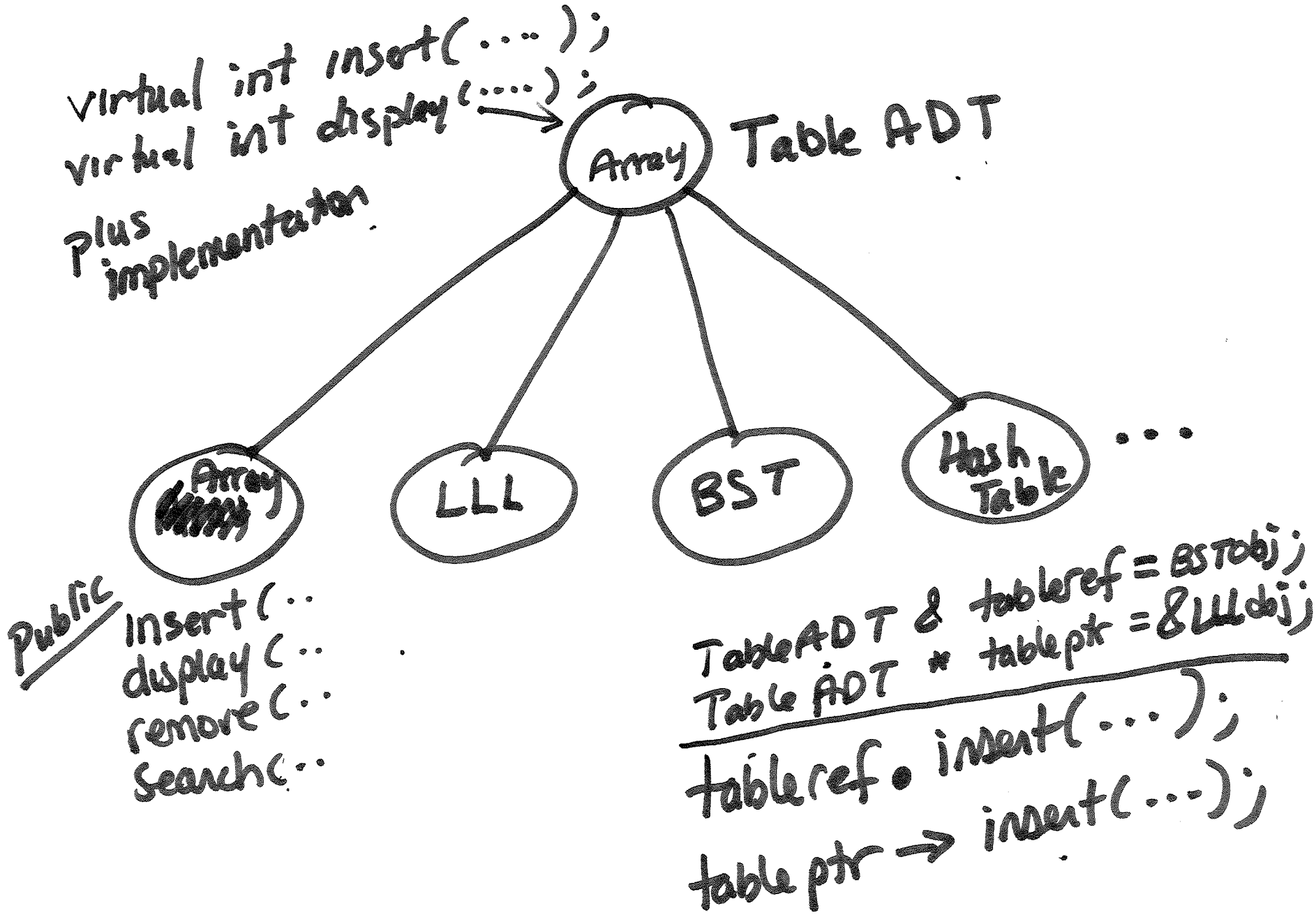
Rules

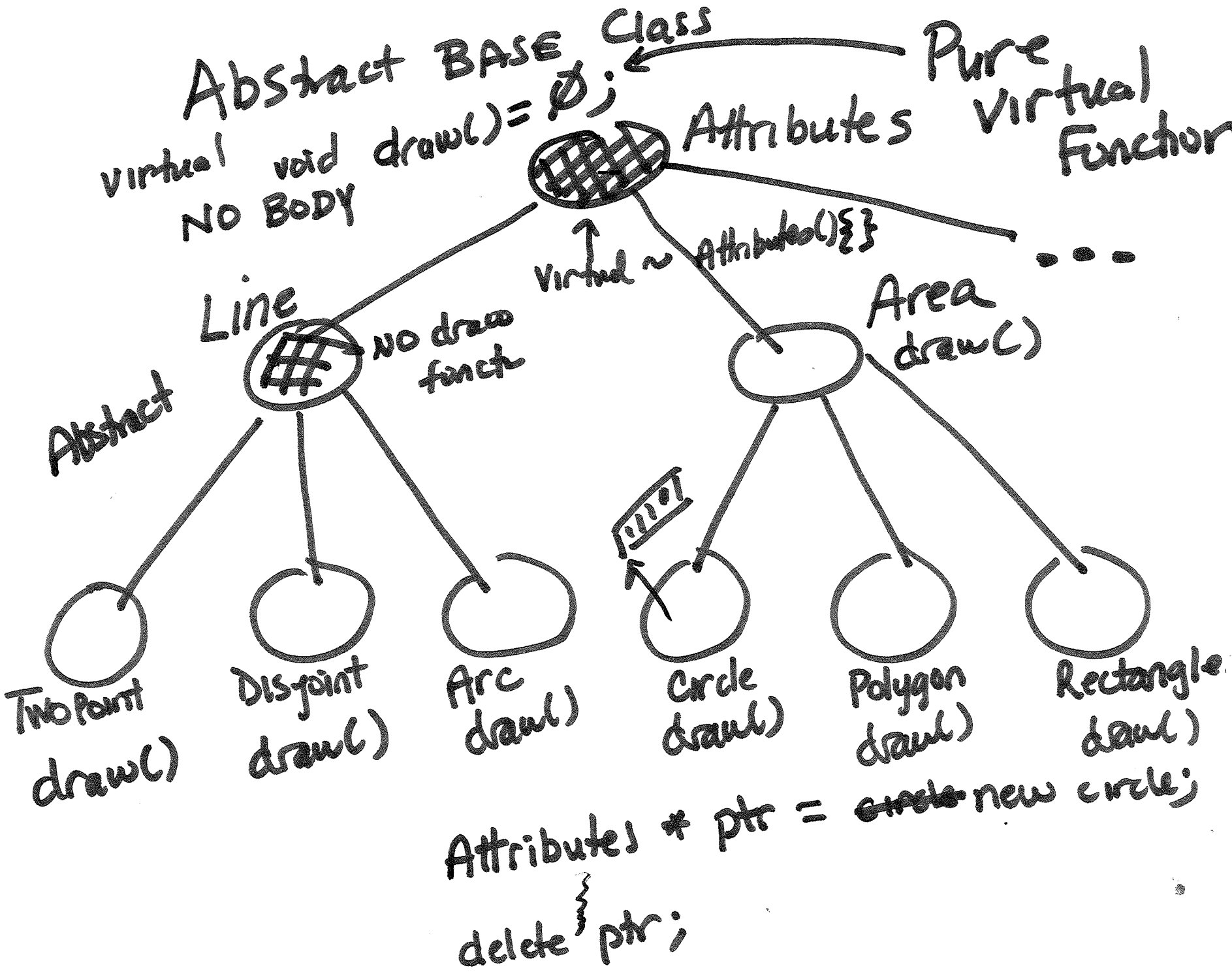
- Inheritance hierarchy
- Use Pointer or Reference to common base class
- Define functions IN base class as **virtual**
- At run time, have pointer or reference-point/refer to any object within hierarchy

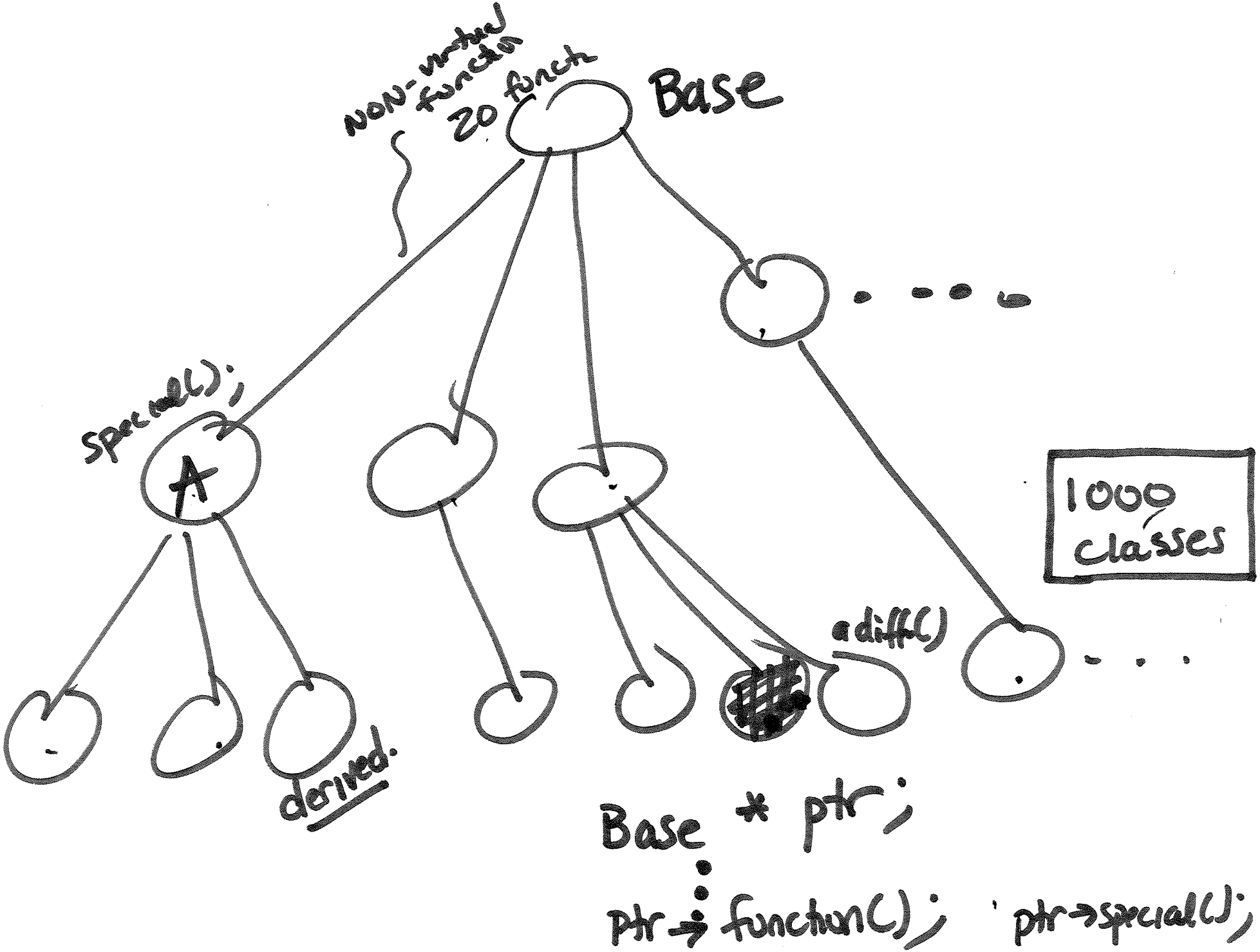
- The functions **MUST** have the same return type AND argument list EXACTLY, within hierarchy.
- Destructors must be virtual

ptr → Statement();

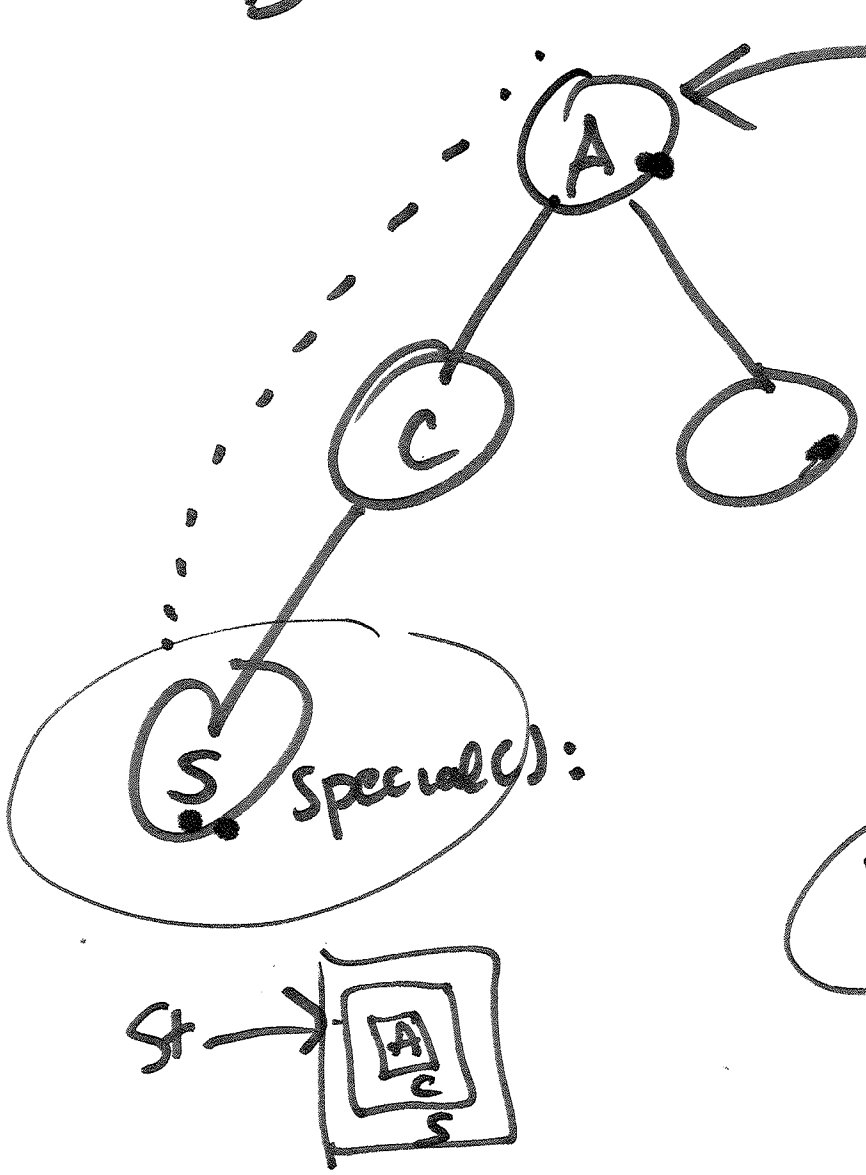
Account ptr


virtual int insert(....);
virtual int display(....);
plus implementation.
Array
Table ADT
Array
LLL
BST
Hash Table
...
Public
insert(..
display(..
remove(..
search(..
TableADT & tableref = BSTobj;
TableADT * tableptr = &LLLobj;
tableref . insert(...);
tableptr -> insert(...);


Abstract BASE Class
Pure Virtual Function
virtual void draw() = 0;
NO BODY
Attributes
virtual ~Attributes(){}
...
Line
Abstract
NO draw func
Area
draw()
TwoPoint
draw()
Disjoint
draw()
Arc
draw()
Circle
draw()
Polygon
draw()
Rectangle
draw()
Attributes * ptr = new circle;
delete ptr;


NON-virtual function
20 funct
Base
special();
A
1000 classes
diff()
derived.
Base * ptr;
ptr → function();
ptr → special();

Downcasting



ptr->special();

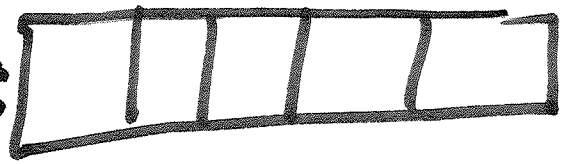
Student* aPtr = ptr;

Pointers to Functions

```
int p1;
int *p2;
int *p3[4];
int *p4();        <-- Prototype!
int (*p5)();
int *(*p6)();
```

p5 is a pointer that could point to a function that takes no args and returns an int.

```
int array[5];
```

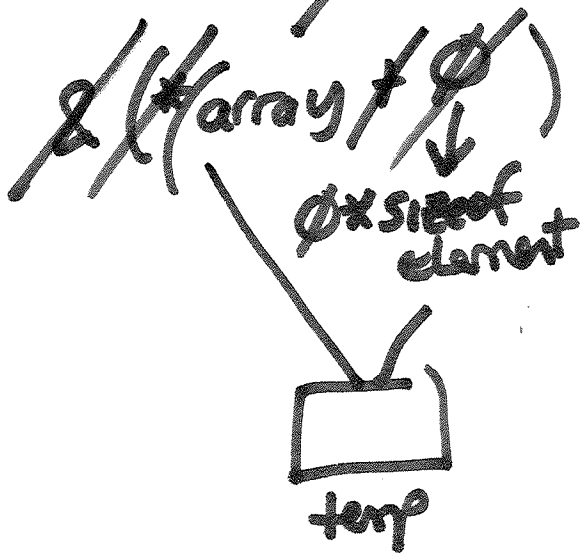



```
array[i] = 10;
```

```
int *ptr = array; // ptr = &array[0];
```

```
*ptr = 10;
```

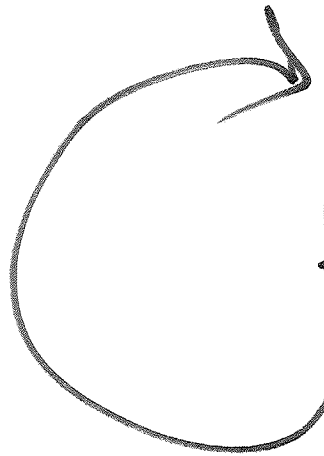
```
ptr[i] = 10;
```



```
int display();
int statenat();
int (*p5)();
```

~~p5 = &display();~~

```
p5 = display;
```



```
i = p5();
```