Modular Design

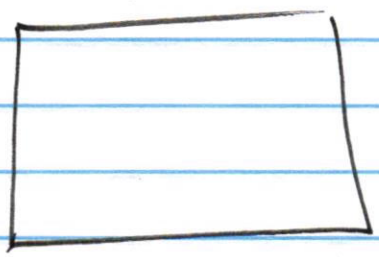
a set of modules

+

call relationships between modules

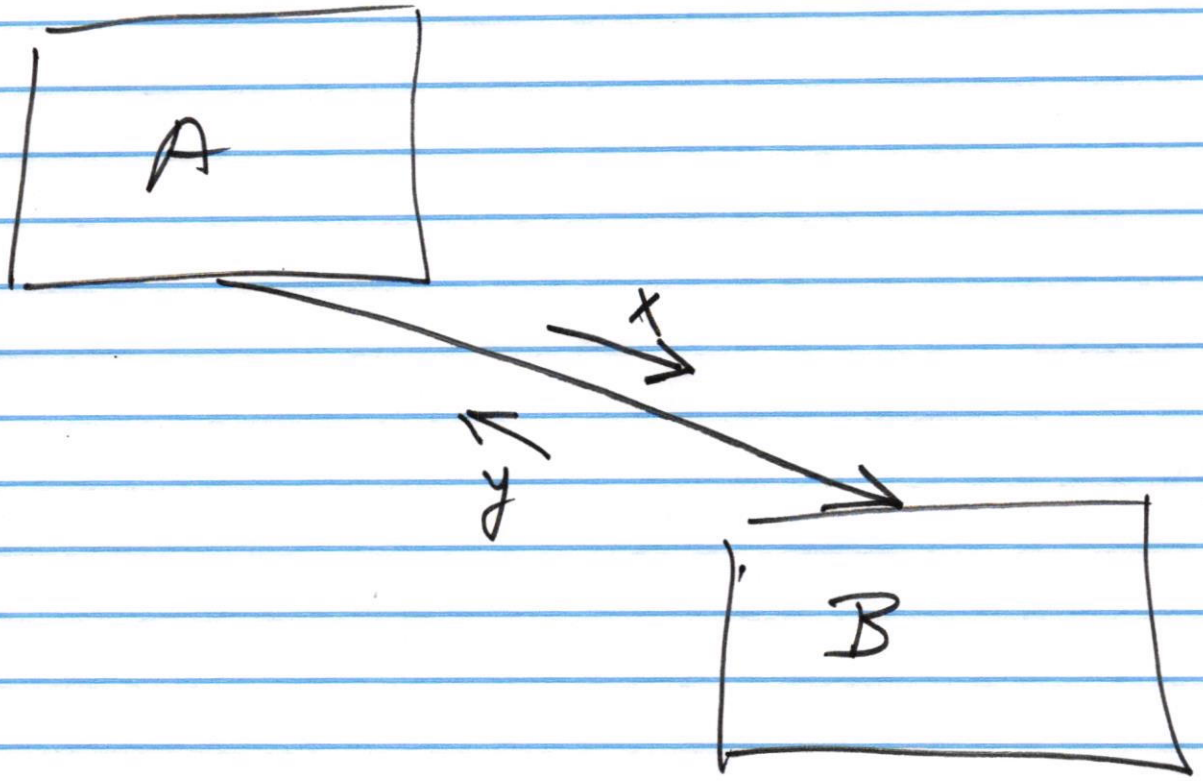
↓

structure chart



module

call relationship.



A calls B

x is an input data to B

y is an output data from B

Design evaluation criteria

* coupling.
* cohesion
* information hiding.

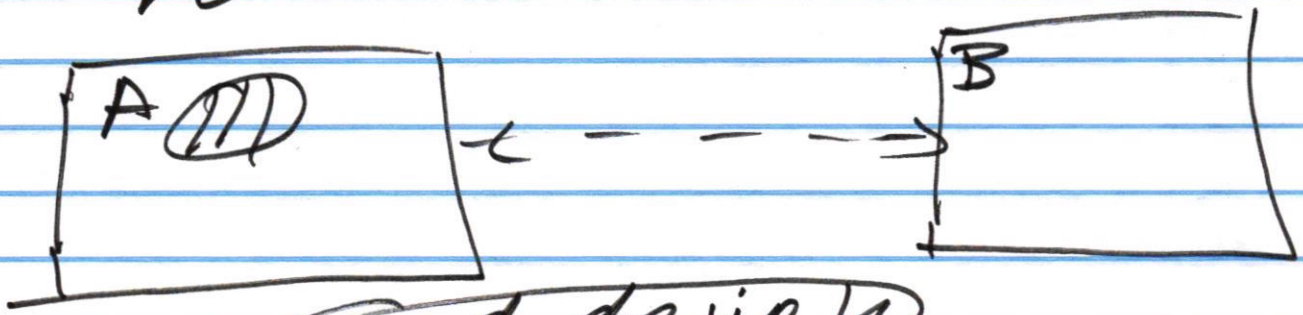
Maintainability.

the major factor for the evaluation

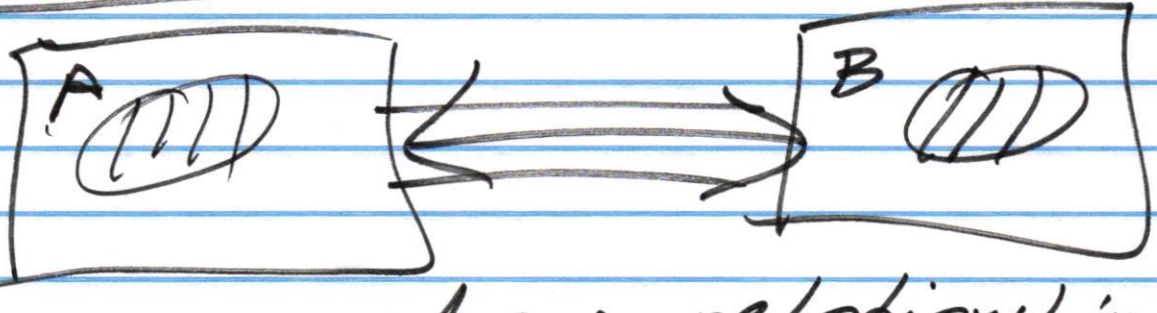
coupling.

a degree of relationship between two components

weak relationship



good design



strong relationship.

bad design

Types of coupling.

worst * common coupling.

*

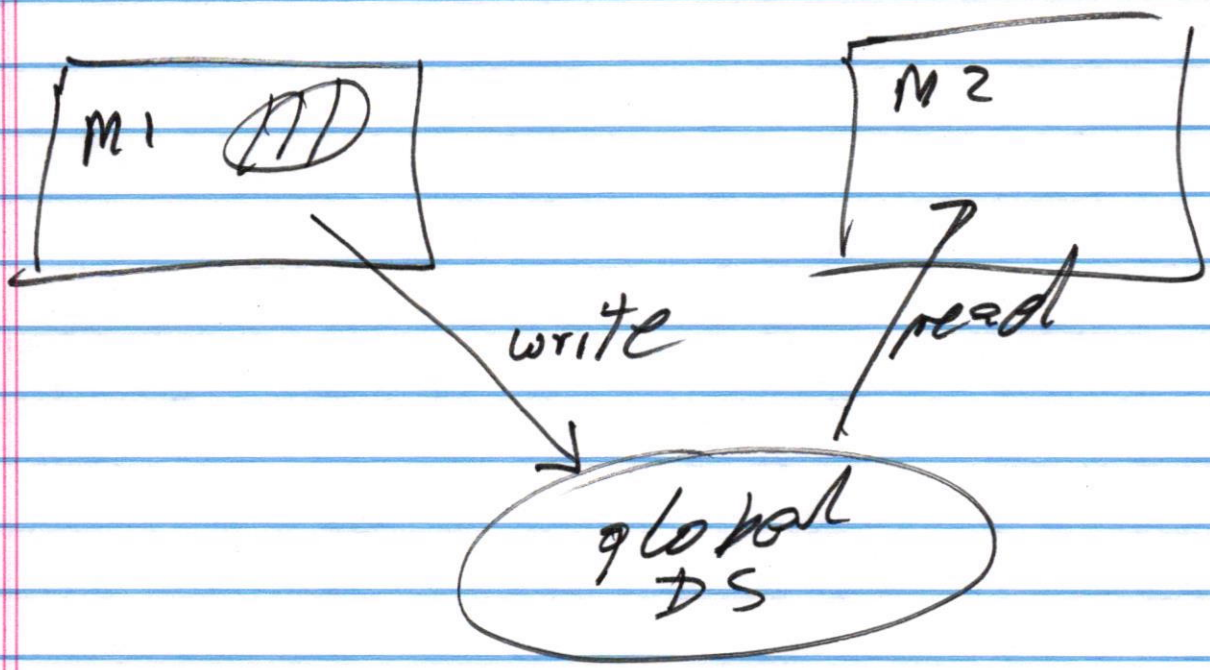
*

*

best * data —11—

* no —11—

Common coupling.



write - read relationship.

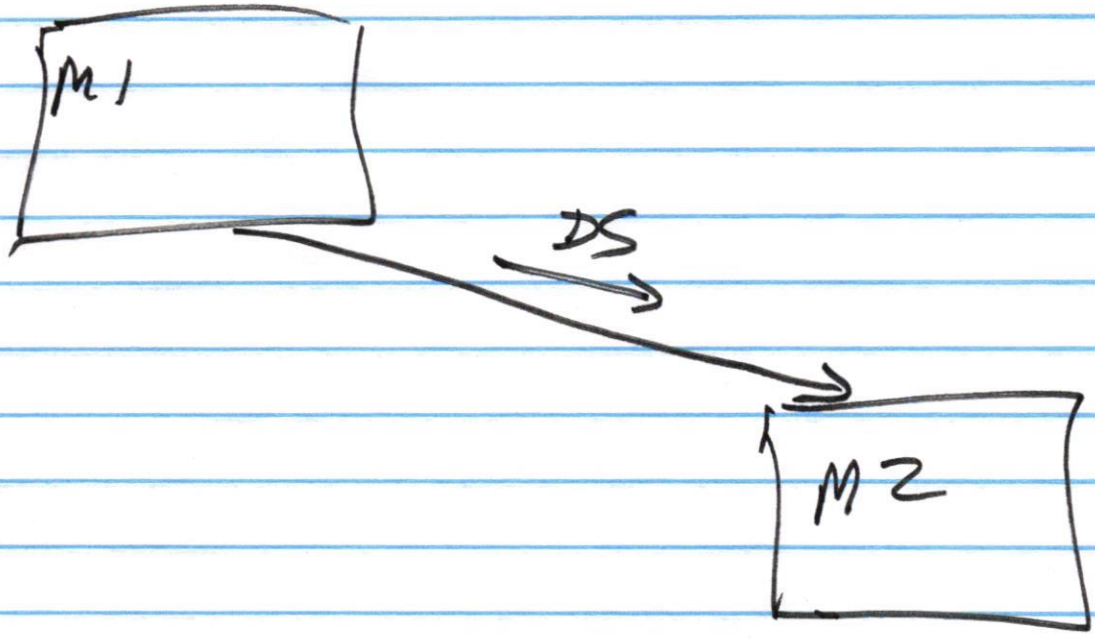
---

read - read relationship

write - write - " -

no common coupling

Data coupling.



all elements of DS are used by M2

Grader system

R1: Read correct answers

R2: Read student answers

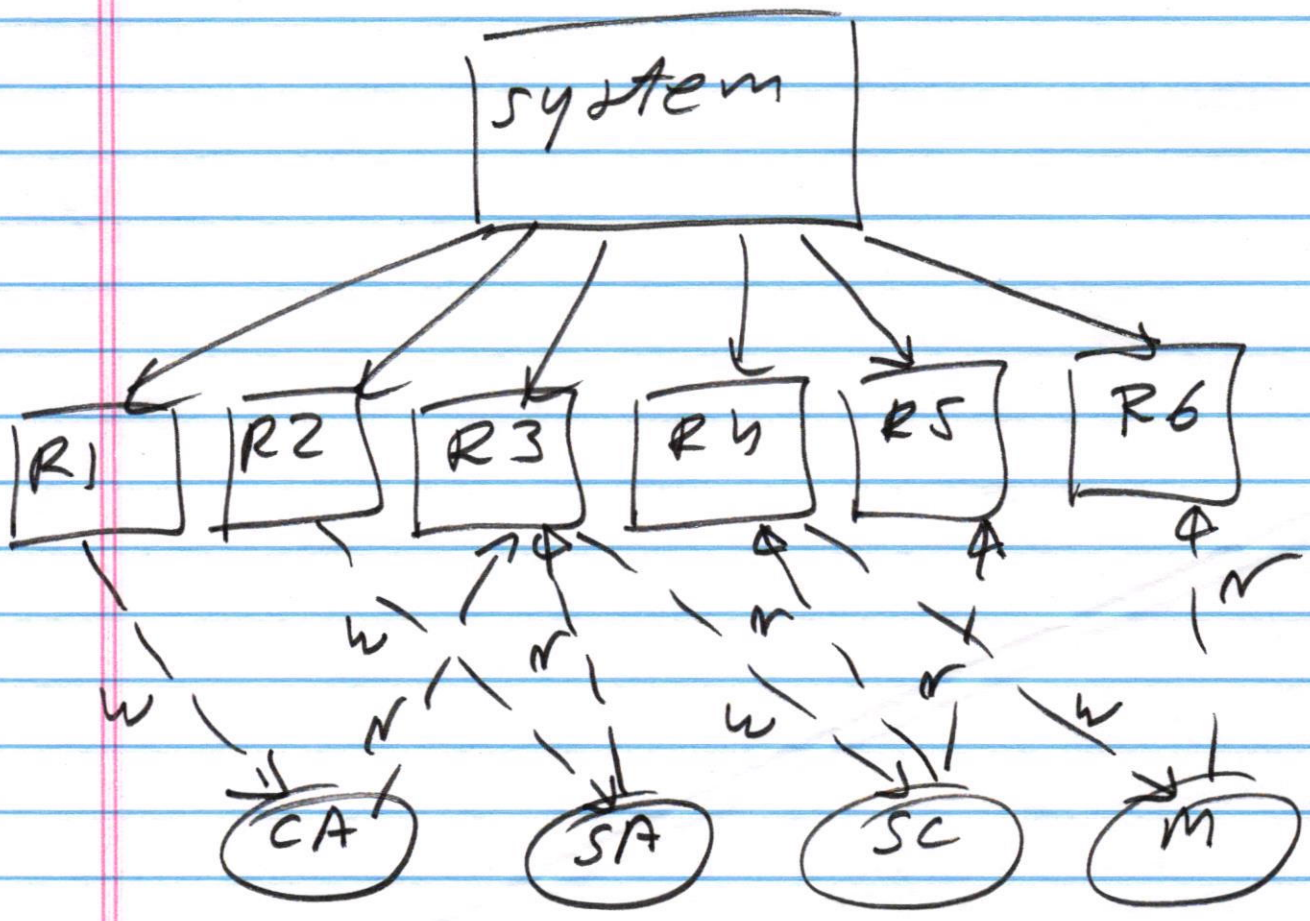
R3: Compute scores

R4: Compute statistics: mean

R5: Report test scores

R6: Report statistics: mean

Design #2



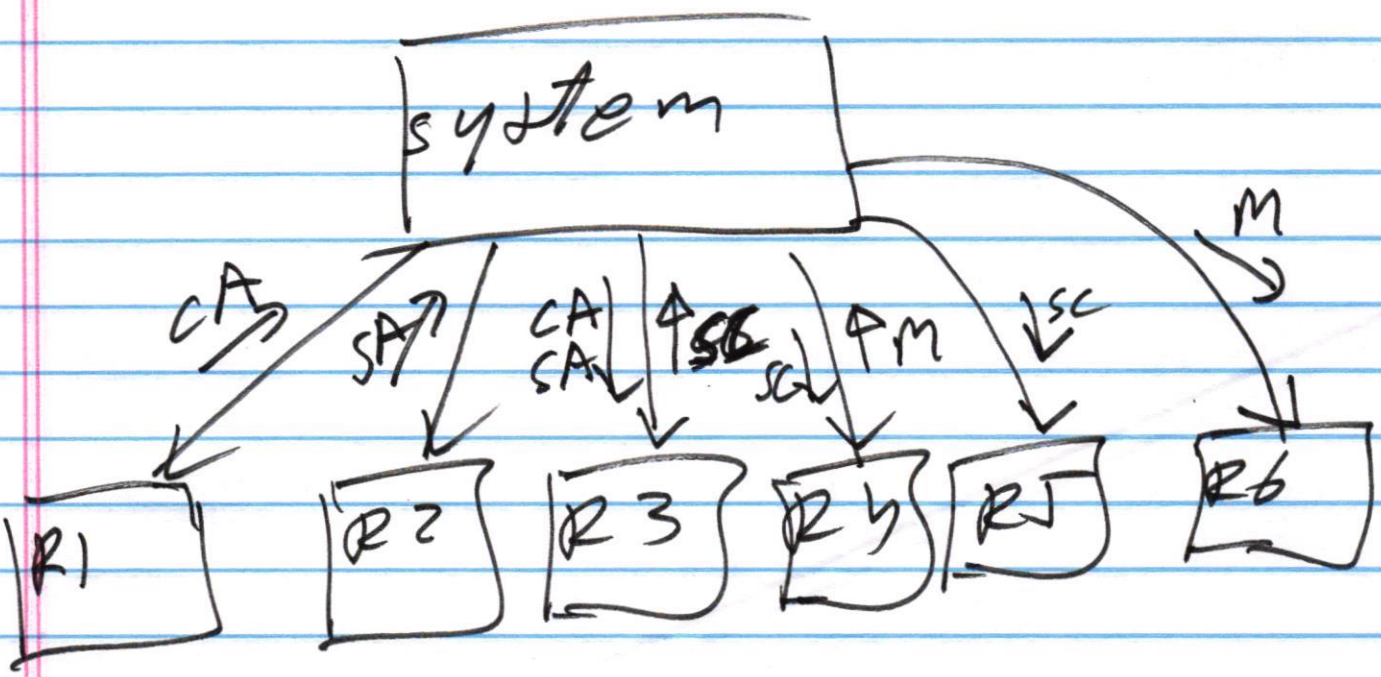
CA: correct answers
SA: student " "
SC: scores
M: mean

common couplings

R1 → R3
R2 → R3
R3 → R4
R3 → R5
R4 → R6

maintenance: bad

performance: good



maintenance: good

performance: bad

maintability vs performance

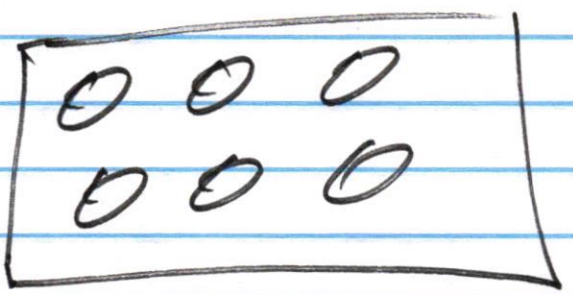
trade-offs between maintainability and performance

cohesion

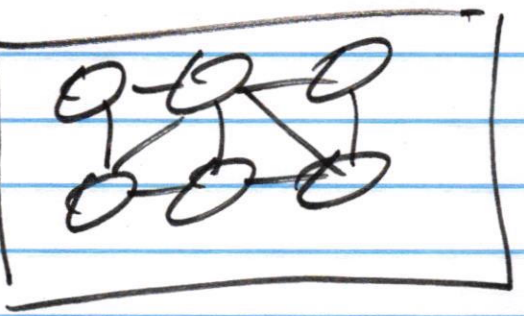
a degree of relationship between internal elements of a component

element: responsibility.

no relationship
weak —"—
bad design



strong relationship
good design



cohesion

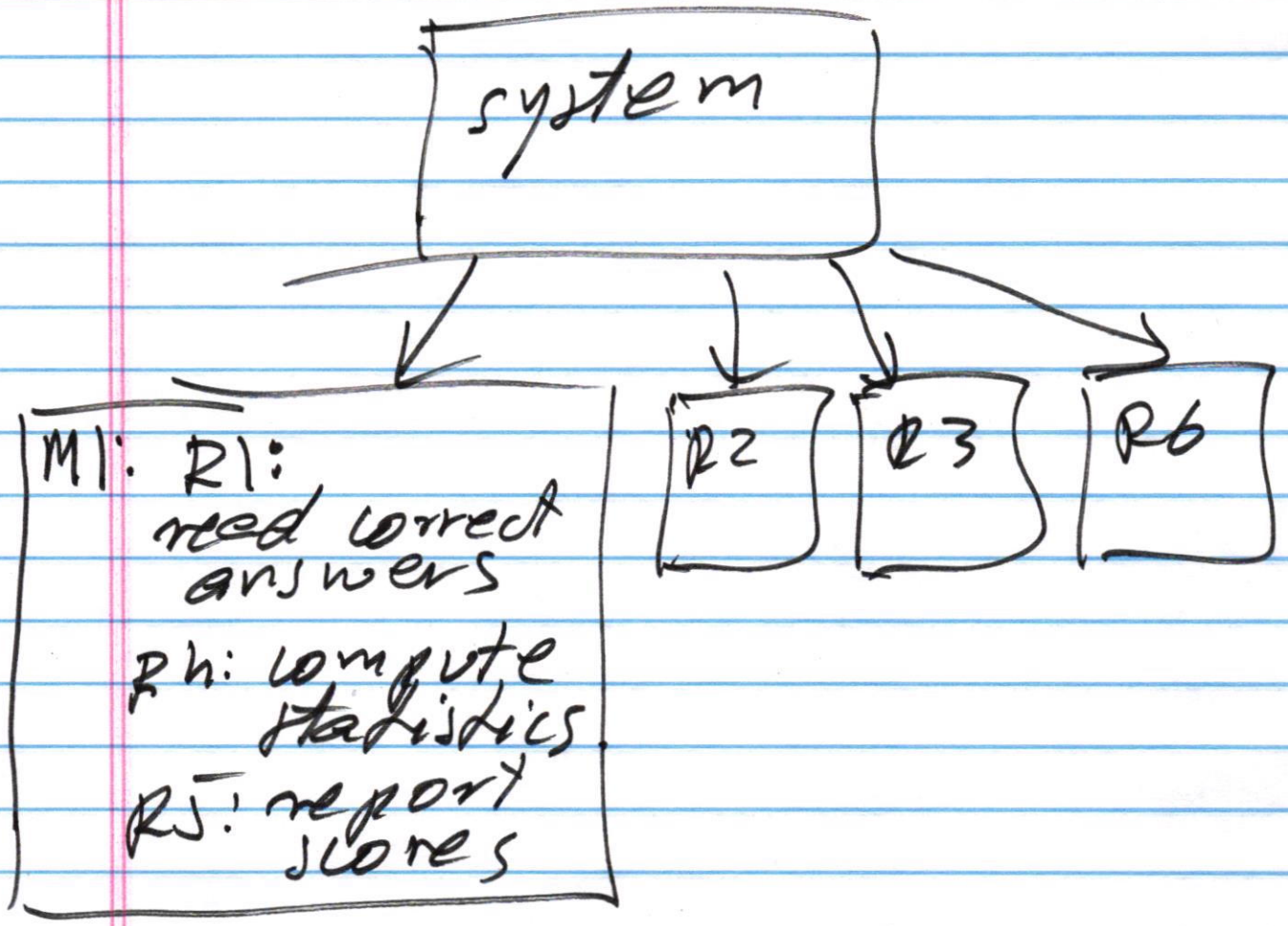
worst ✱ coincidental

✱
✱
✱
.
.
.

best ✱ ✱ functional
informational

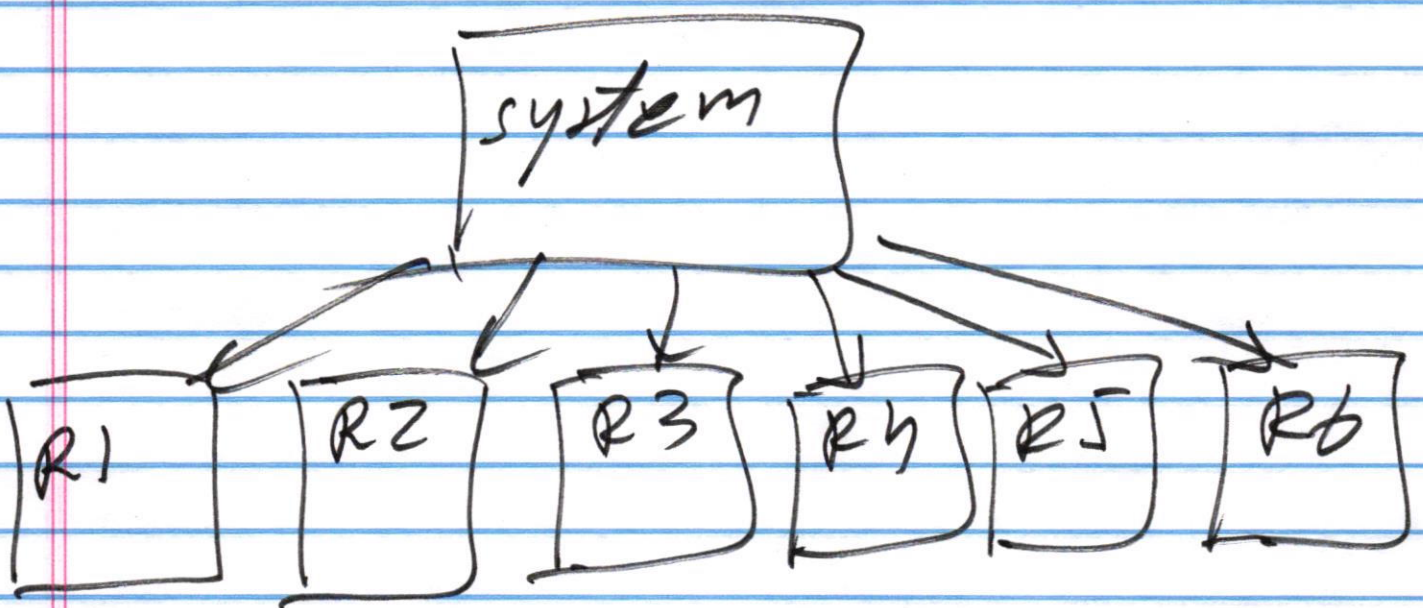
coincidental cohesion

No meaningful relationship between responsibilities.



Functional cohesion

a component performs one well defined responsibility.



functional cohesion

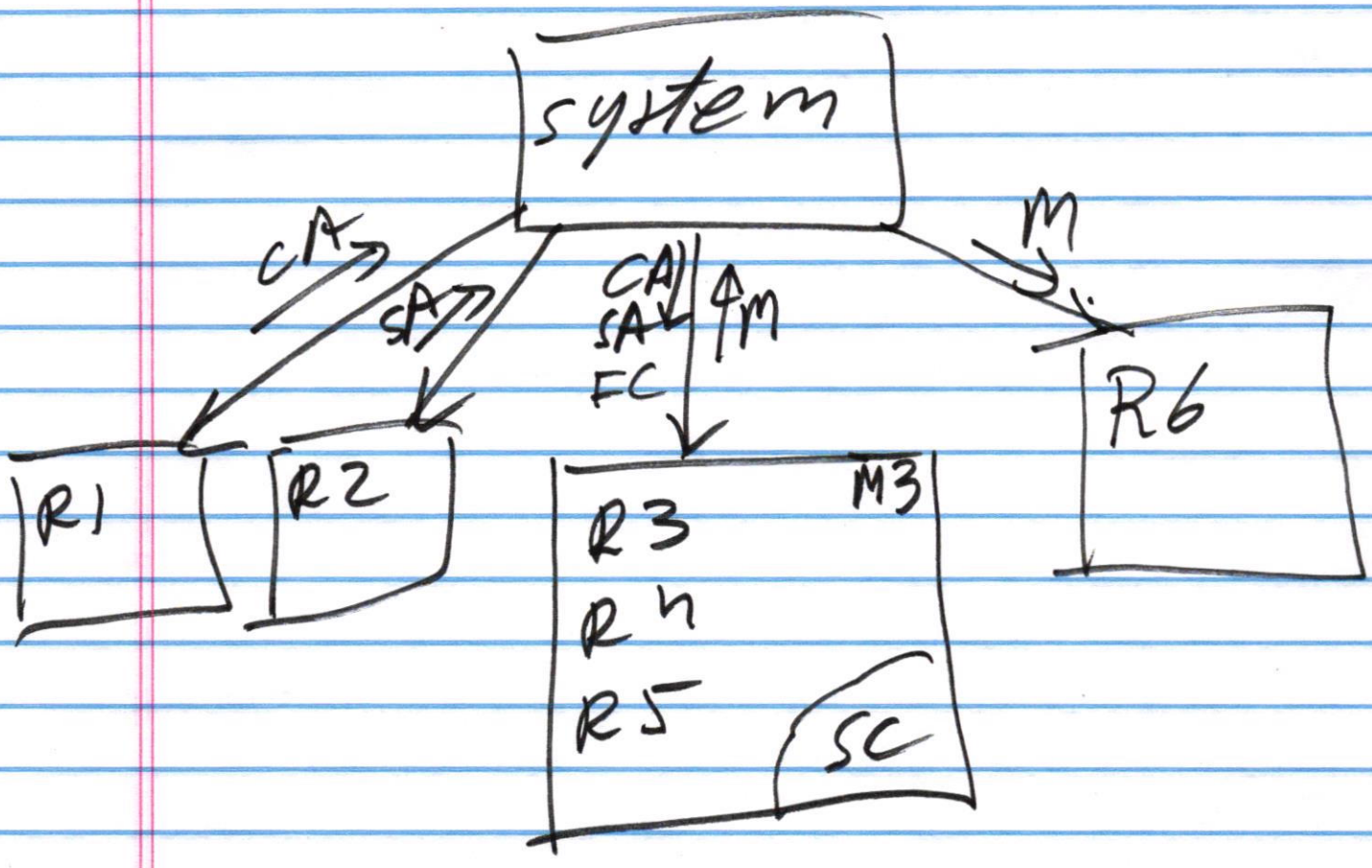
Informational cohesion

* component with multiple responsibilities.
* all responsibilities are related

example

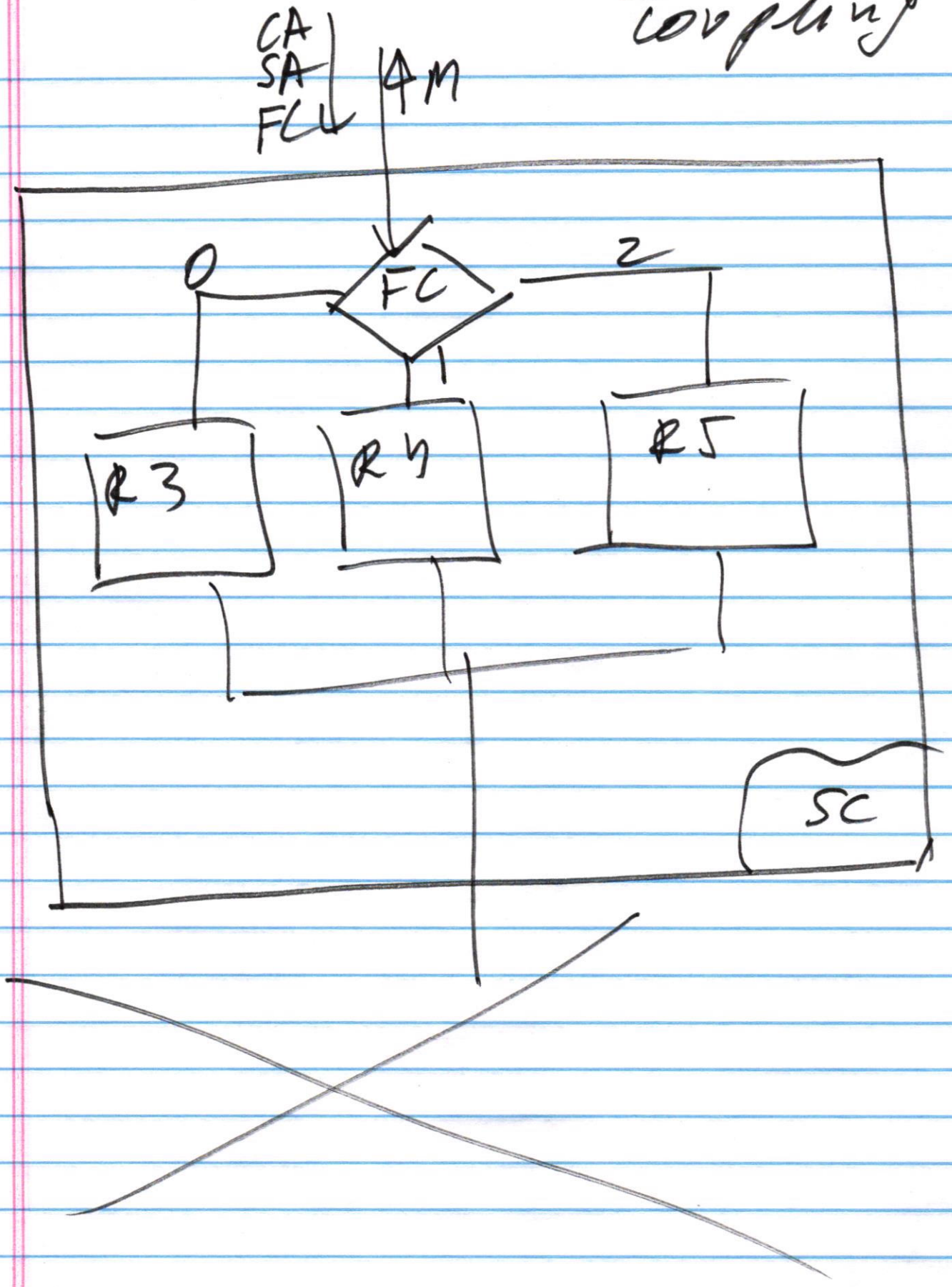
* all responsibilities acces the same data structure.
* hide the data structure inside of a module.
* each responsibility must have a seperate entry point and exit point.

CA, SA, SC, M

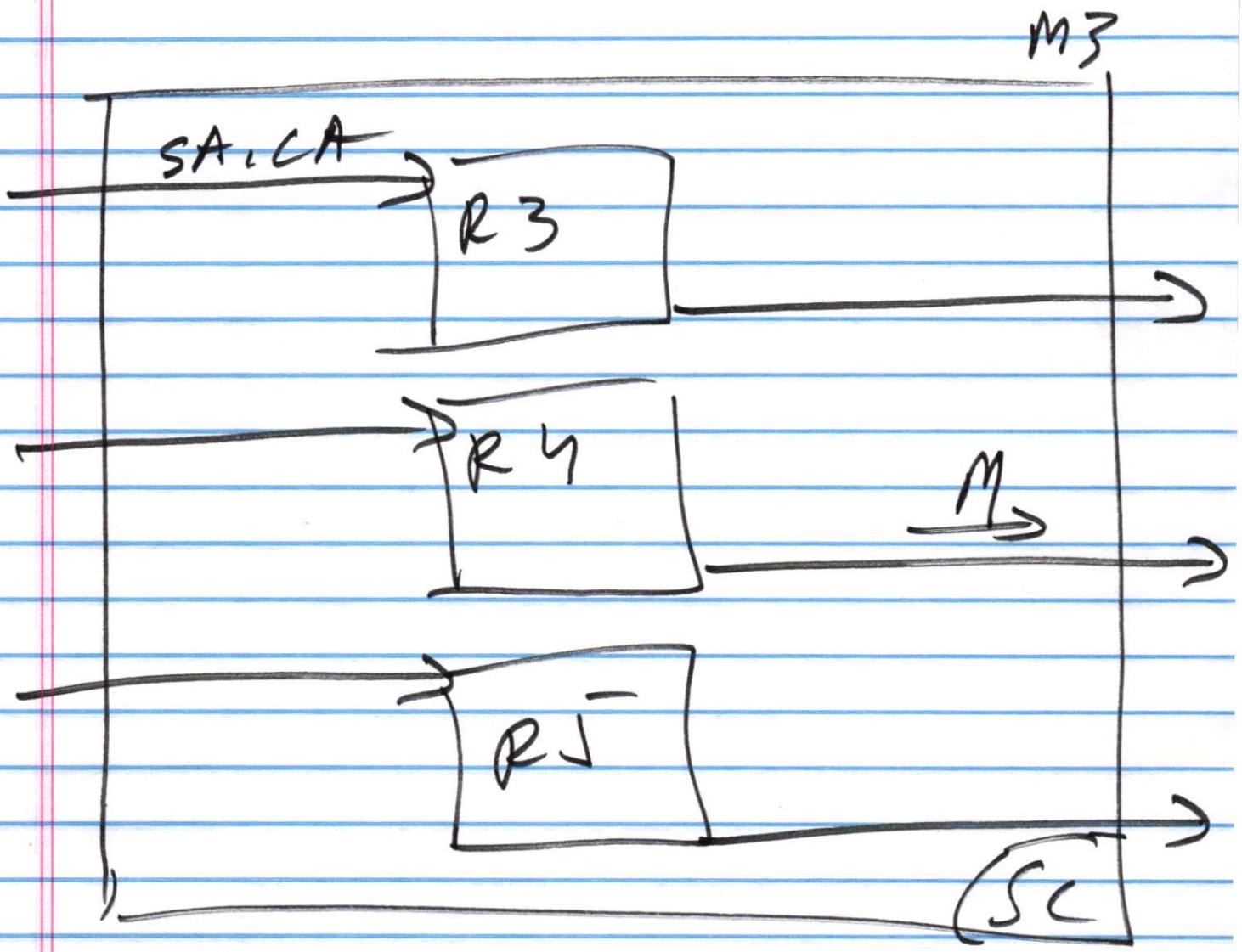


FC: function code

control coupling.



multiple entry/exit points.

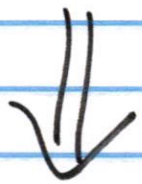


good design

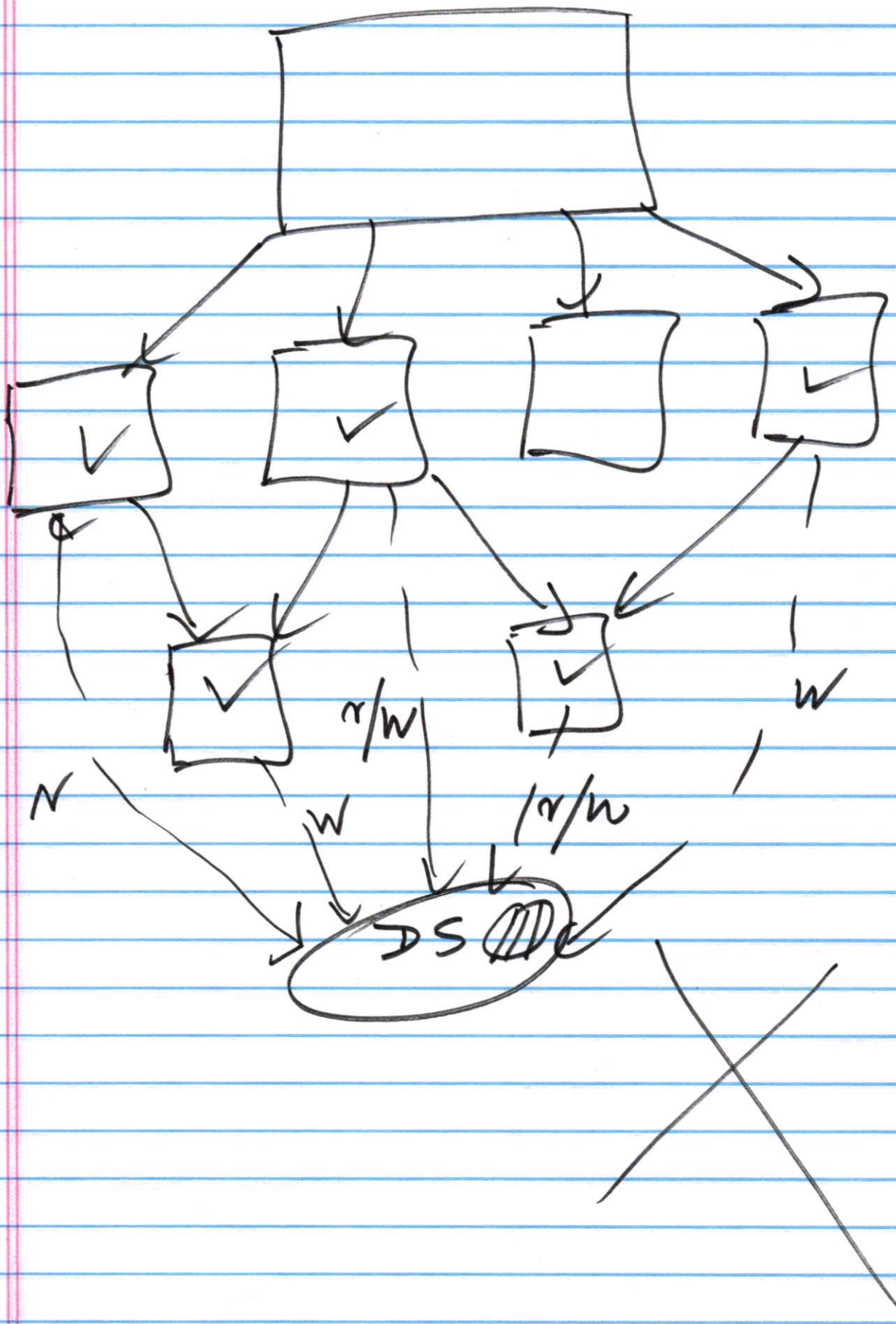
* maximize cohesion
* minimize coupling.

Information hiding.

making changes to data structures.



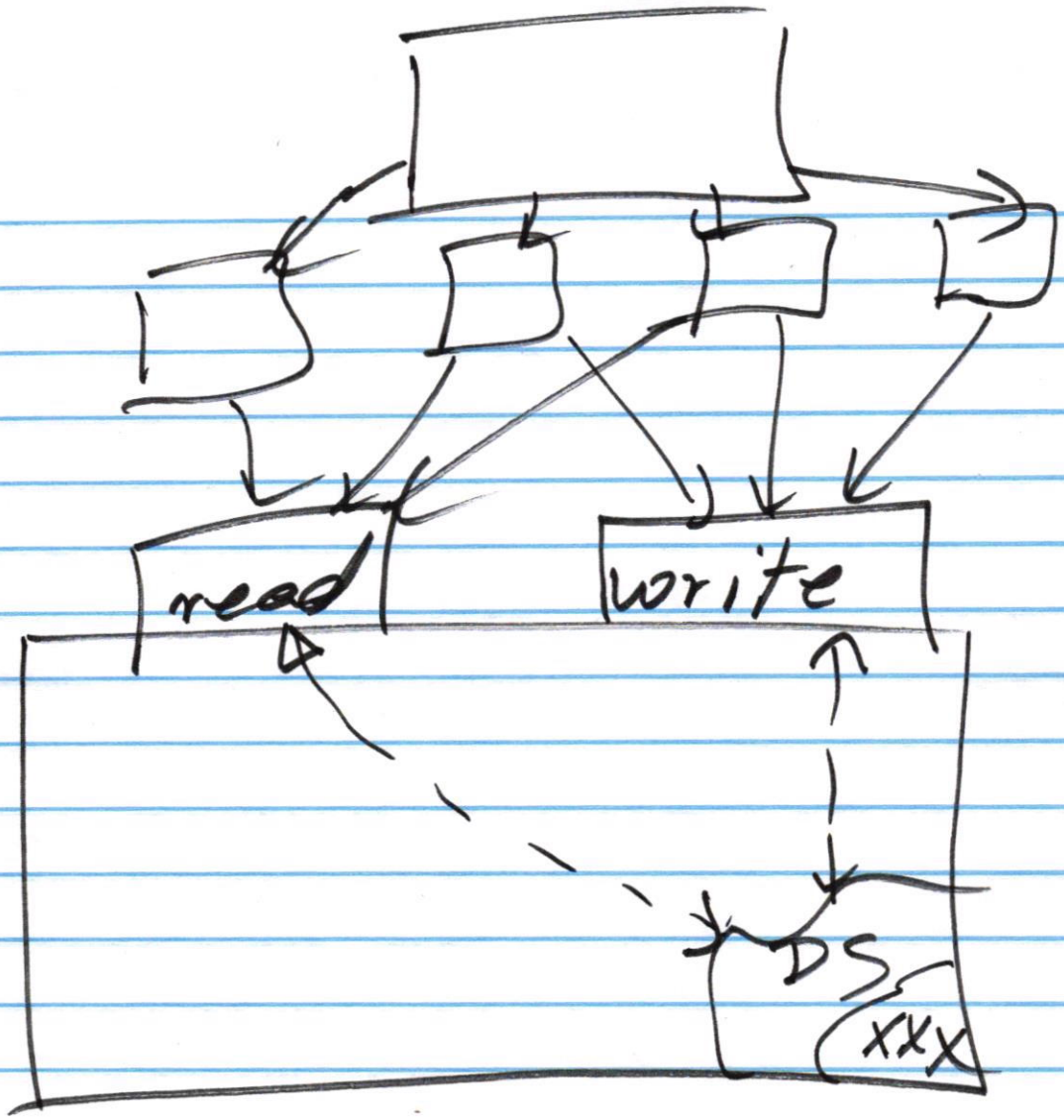
minimize modifications when data structures are modified.


r/w
r
w
r/w
w
DS

* group responsibilities that "must" access directly data structure.

* put them into one component

* provide access operations to indirect access to the data structure


read
write
DS
xxx

* modular design
* object-oriented design
  * information hiding.
  * informational cohesion
  * user defined data type
  * inheritance
  * polymorphism related to inheritance

user-defined data type

primitive data types

```
int    x, y, z;
char   a, b;
string  S1, S2
```

user-defined data type.

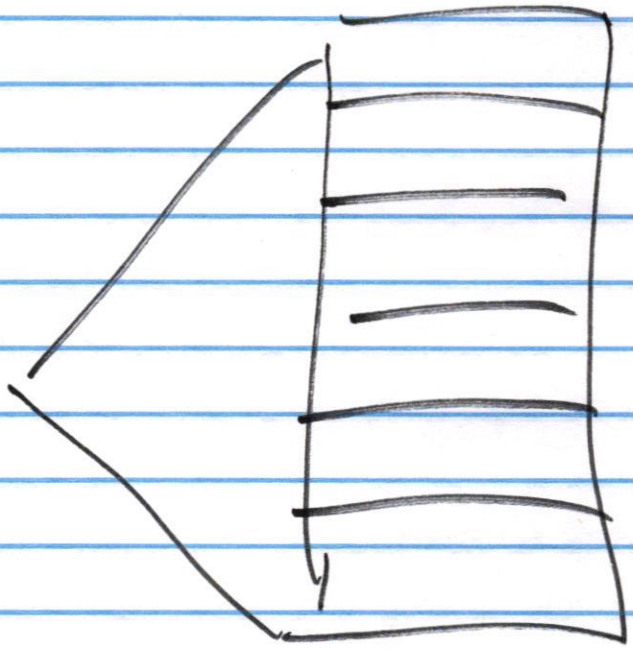
data type: stack,
graph
tree
.

```
Stack    S1, S2;
S1.push(x)
y = S1.pop()
```

class: user defined
data type.

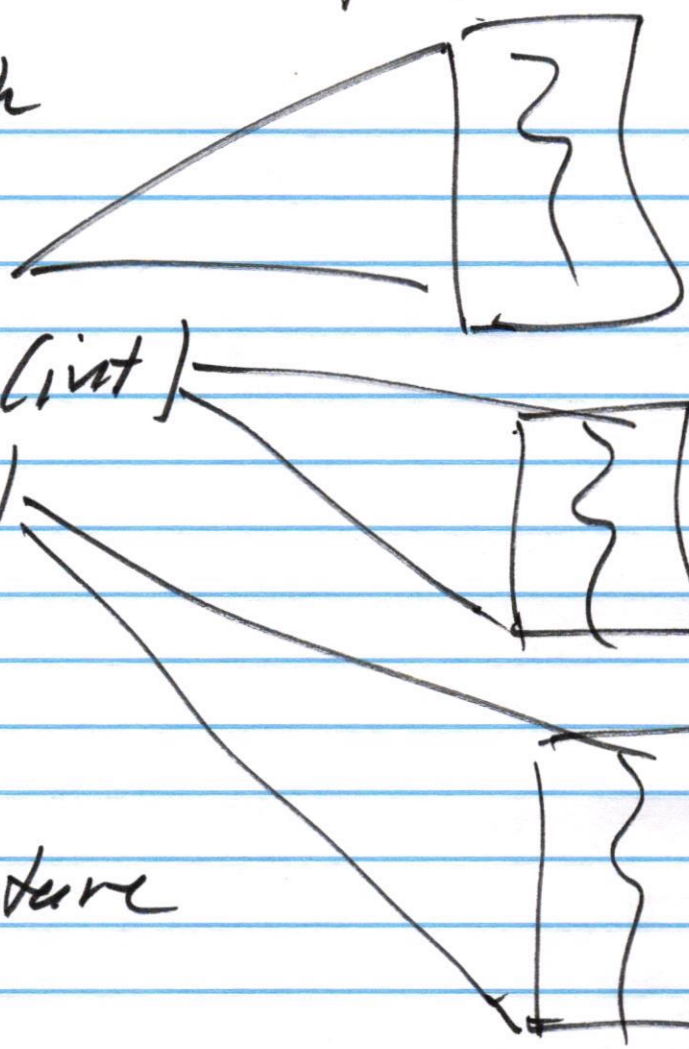
```
class name
{public:
    operations

 private
   data
};
```



implementation of operations

implementations of operations



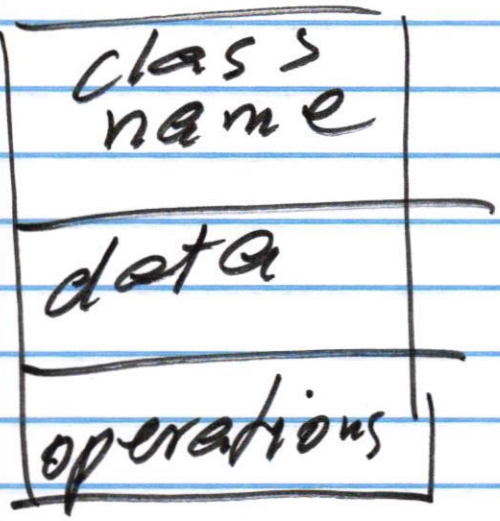
```
class stack
{ public:
    stack()
    void push(int)
    int pop()
  private:
    int top
    data structure
};
```

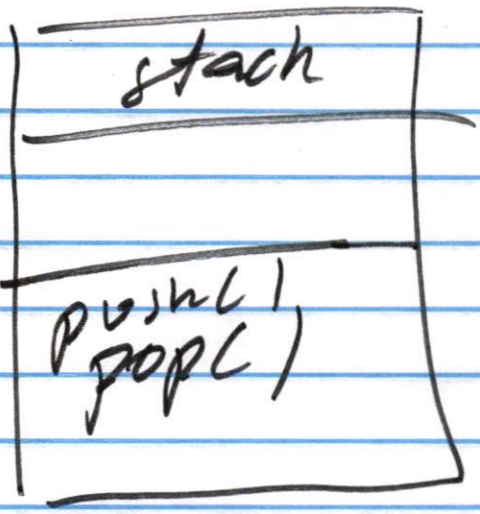
---

stack S1, S2, S3

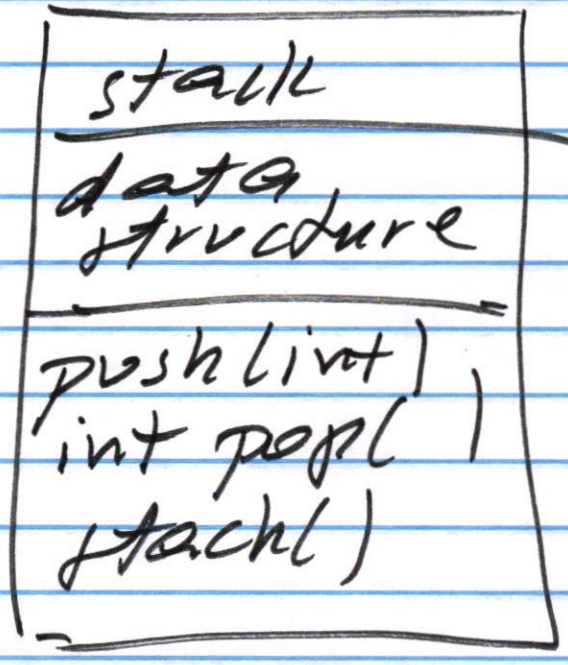
S2.push(x)
S1.push(y)
z = S2.pop()

class



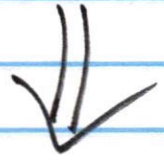


Detailed design



Object-oriented design

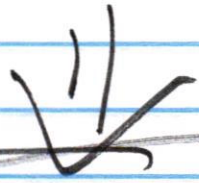
system structure



a set of classes

+

relationships between classes



class diagram

graphical notation.

UML: Unified Modeling Language

relationships between classes

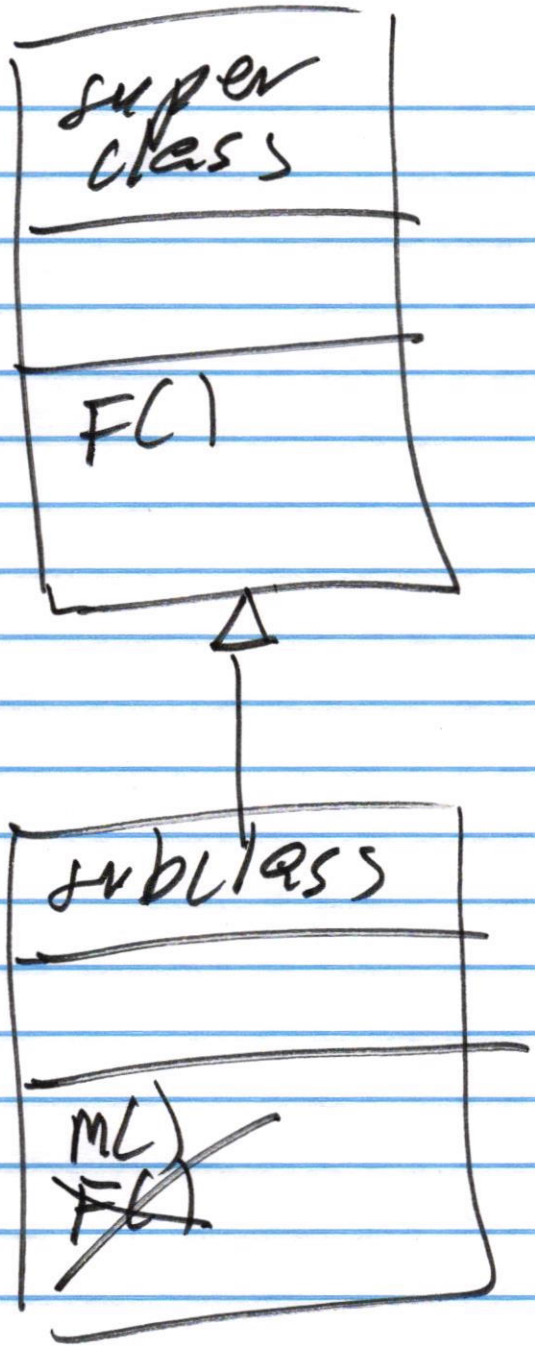
1. inheritance
2. aggregation
3. association

Inheritance

Motivation: reuse of code

1. superclass (base class)
2. subclass (derived class)

subclass "inherits"
* operations/methods
* data structures
from superclass !!



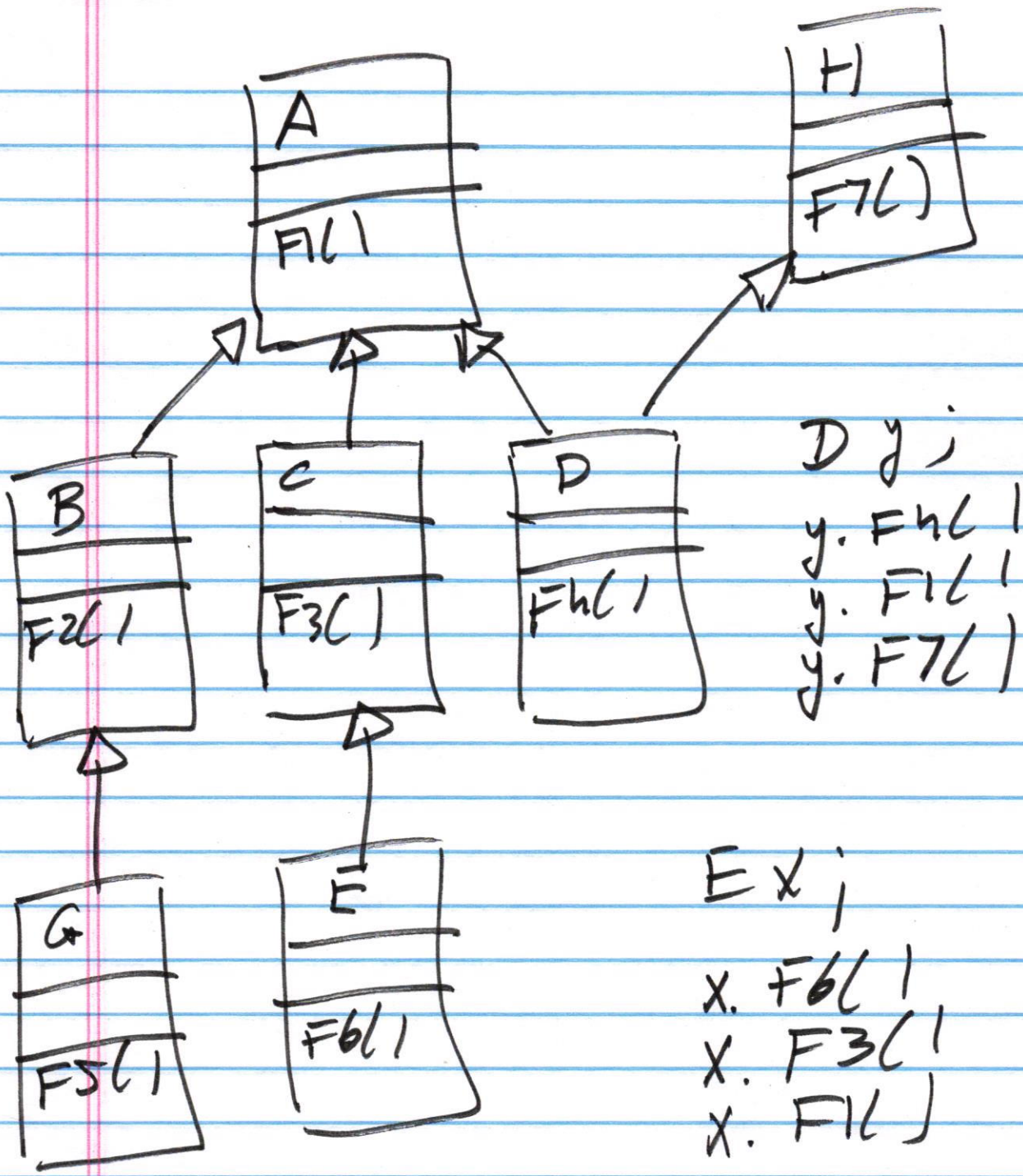
superclass A

subclass B

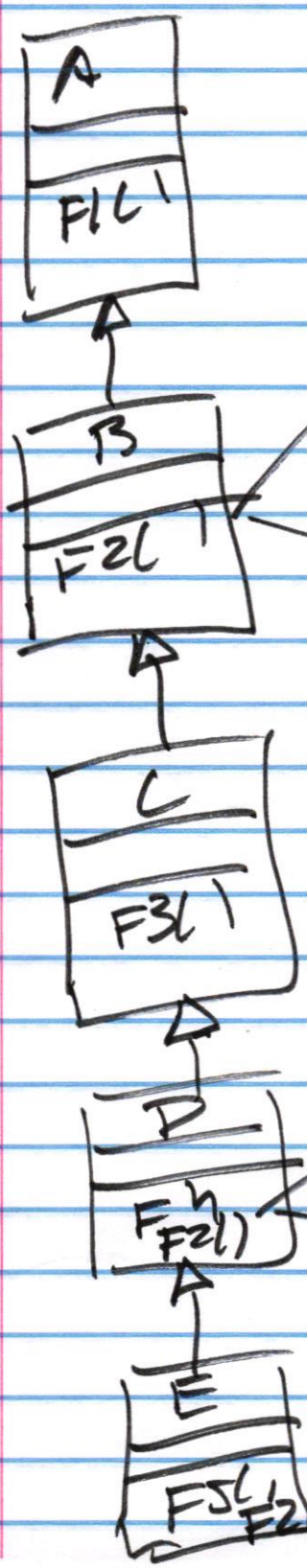
A.F()

B.M()

B.F()


A
F1()
H
F7()
B
F2()
C
F3()
D
Fh()
G
F5()
E
F6()
D y;
y. Fh()
y. F1()
y. F7()
E x;
x. F6()
x. F3()
x. F1()

coupling related to inheritence



inheritance is a very strong coupling.

modify F2()

suppose:

C and E are OK.

but

D is not OK.

old version of F2()

new version of F2()