Date:

## Object Oriented matrics:

WMC : weighted methods per class (How many functions)

CBO : Coupling Between Objects (too much classes in one class) ⇑

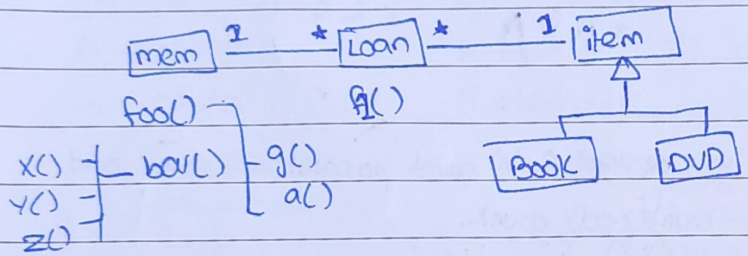
DIT : Depth of Inheritance Tree (if inherited, then check depth) LCom : Lack of Cohesion of method.

NOC : Number of children (Total Number of Children) (immediate children, don't count grand children) Too much is bad.

RFC : Response for a class (when you call a function, to how many function does it calls further) ⇒ Sum of RFC of total member function is the RFC of the class.

metrics are needed to measure quality of programs/designs.

Example: Compute the metrics




Coupling: Kitni classes se connected hai

TOOLS:
→ CCCC
→ SonarQube
→ Understand

| | MEM | LOAN | ITEM | BOOK | DVD |
|---|---|---|---|---|---|
| WMC | 2 | 1 | 0 | 0 | 0 |
| CBO | 1 | 2 | 3 | 1 | 1 |
| DIT | 0 | 0 | 0 | 1 | 1 |
| NOC | 0 | 0 | 2 | 0 | 0 |
| RFC | 5 | 0 | 0 | 0 | 0 |
| LCOM | | | | | |

⇒ we need Balanced Values

Rust code Analysis

LCom:

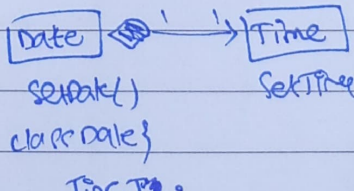
```
class Date{
    int d,m,y;
    void setDate(-){ }..
    void setTime(-)
}
```

LCom = 1 ⇑ (int h, min, s;)

LCom ⇓ 2

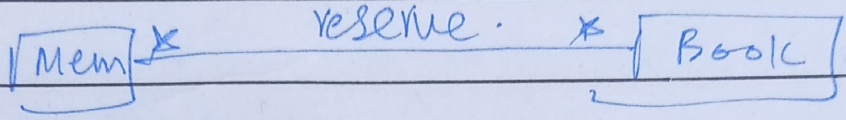
two functions that is used to set all the data members.

⇓ solution.



Date _____ Page _____

* O(1) search time in HASH Table
but comes at cost of increased RAM



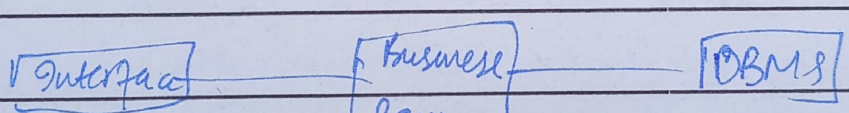

Member class

```
List<Book*> ls;
```

Book class

```
Queue<Mem*> Qq;
```

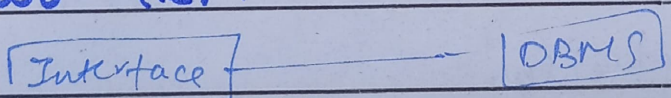
## Three tier architecture:




Application classes
Logic

much better than two tier

## Two-tier:




+ code.

→ Not modular, difficult to change.

* Thin client → website user.
* Thick client → user of application

Date: Day:

* In two tier, when trying to do some change in interface, we will basically need to rebuild whole app since codes + interface is in same tier

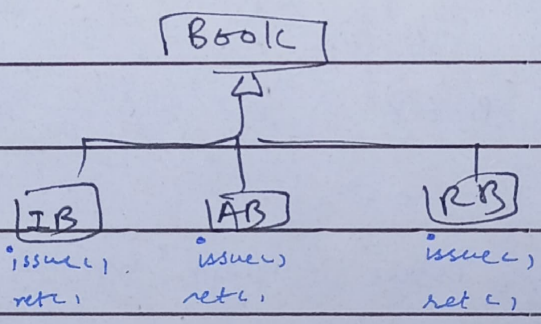
→ We can remove if/else by using polymorphism

```
class Book {
int state;
bool issue (Mem* m) {
    if (state == AVA)
    {
A       ——— ———
        state = ISS;
        return true
    }
    else
B   return false;
}
```

const int AVA = 3;
const int ISS = 2;

Date _____ Day _____

```
bool ret() {
    if (state == ISS)
    {
        state = AVA;
        return true;
    }
    else
        return false;
}
```




6 For each book, these three classes would be there..

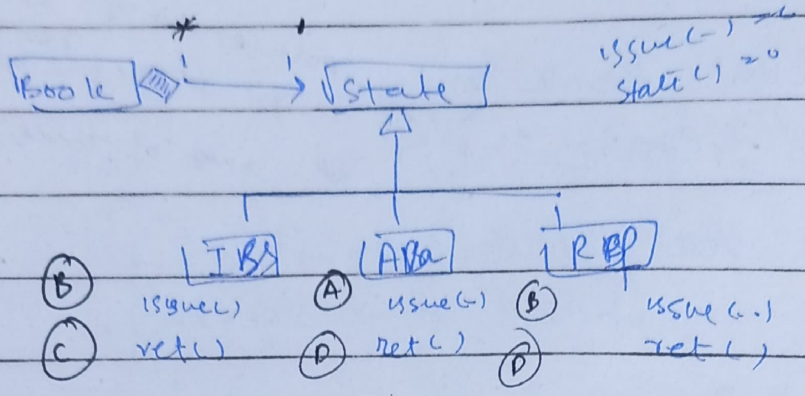
→ Data of each book would be inherited in child classes.

Now, when book is changed from available to issue, its data would be copied to issue book class and object of available book class will need to be delleted.

Many to one because of singelton

Date: _____ Day: _____

No array of book in state bcz of uni-directional




- Now, for each book, there is only one ptr of state against it.

- No copying, deletion in this case only status will be set.

→ Make Iss, Ava, Ref classes as Singelton. So, only one object can be created of them.

```
int main ( ) {
  Book b1 ( ) ;
  b1.setState(new Ava( ) ) ;
  // // ( new Iss( ) ) ;
  // // ( Ava::getIns( ) ) ;
  // ( Iss::getIns( ) ) ;
```

with help of singelton

setting state of book b1

Date _____ Day _____

```
class Book {
    State *st;
    bool issue(Mem* m) {
        return st->issue(m);
    }
}

bool ret()
    { return st->ret(); }
}
```

pointer to object of state type stored in it will be of one of three classes

## Object Oriented Metrics:

Metrics are used to measure quality of program/design

**WMC:** weighted methods per class.

no. of fns

**CBO:** Coupling b/w objects.

no. of connections from class → lines

**DIT:**

Depth of inheritance tree

Date: _____ Day: _____

**NOC:**

No. of children. — only children not grand child

**RFC:**

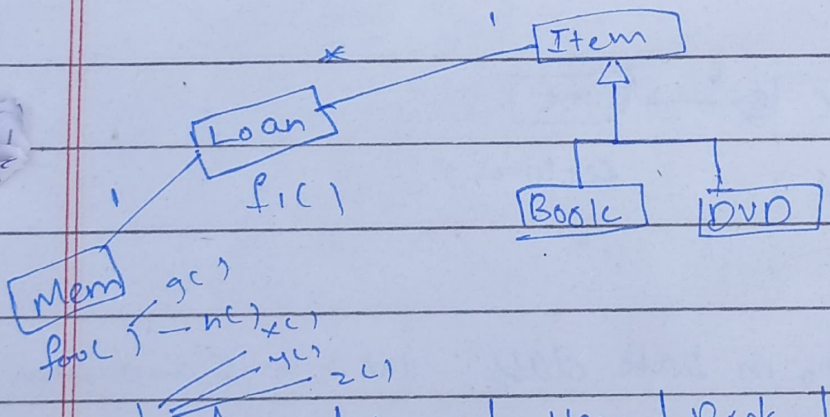
Response for a class

no. of
fun. which is in class → calls how many ↓ funs further

**LCOM:**

Lack of cohesion of methods

need to see code to tell, can't tell from design



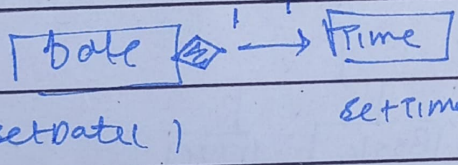

| | Mem | Loan | Item | Book | DVD |
|---|---|---|---|---|---|
| WMC | 2 | 1 | 0 | 0 | 0 |
| CBO | 1 | 2 | 3 | 1 | 1 |
| DIT | 0 | 0 | 0 | 1 | 1 |
| NOC | 0 | 0 | 2 | 0 | 0 |
| RFC | 5 | 0 | 0 | 0 | 0 |
| LCOM | | | | | |

— inherited from Item.

Date _____ Page _____

```
class Date {
    int d, m, y;
    void setDate(--) {---}
    int h, m, s;
    void setTime(--) {---}
}
```

Should be low LCOM




Now, in Date class int h, m, s and fun will not be there. There would be ptr/object to time class such as t1.

Date: _____ Day: _____

SONAR QUBE

CCCC

I examines Java code etc

generate reports about code