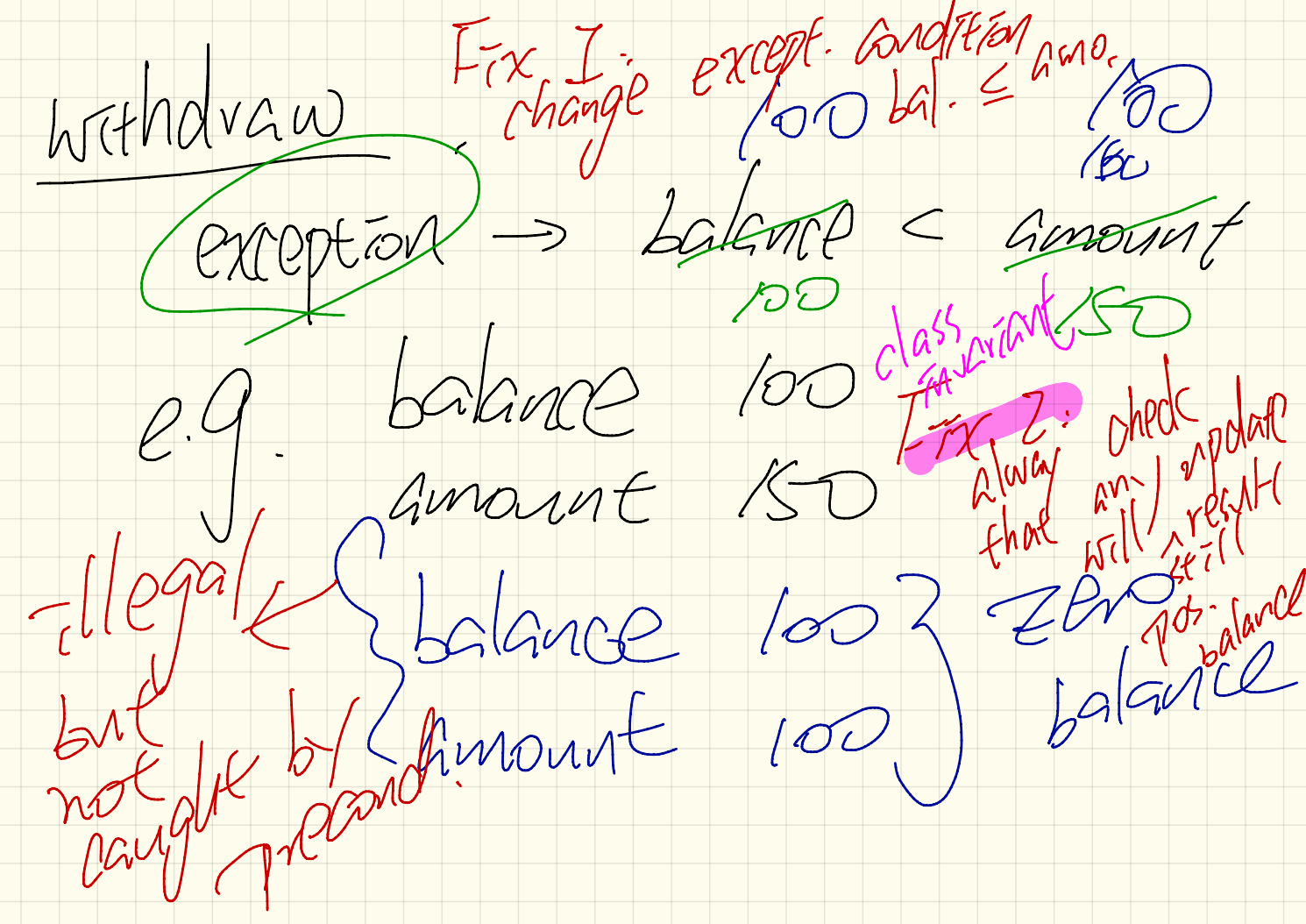
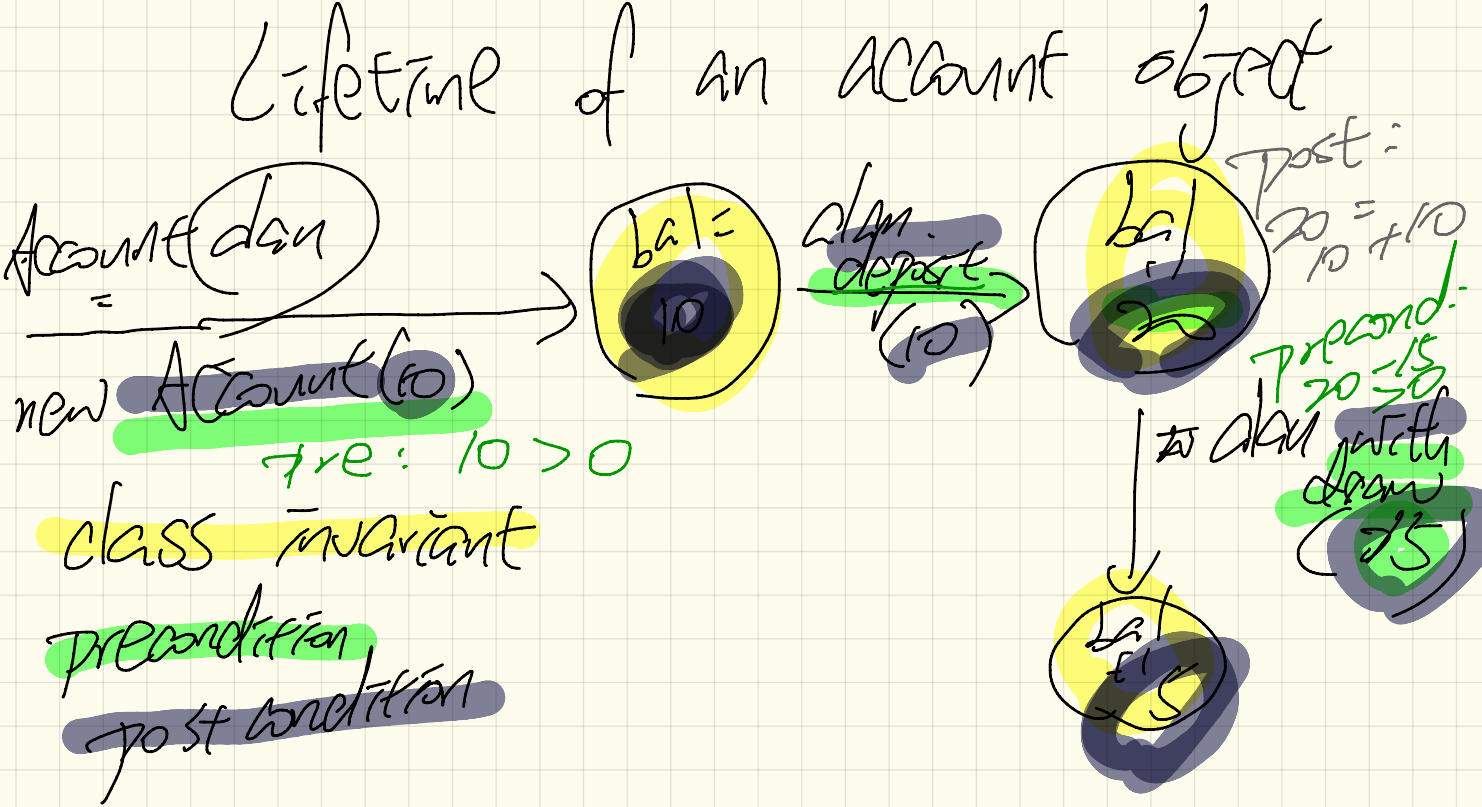
Wednesday Sept. 13

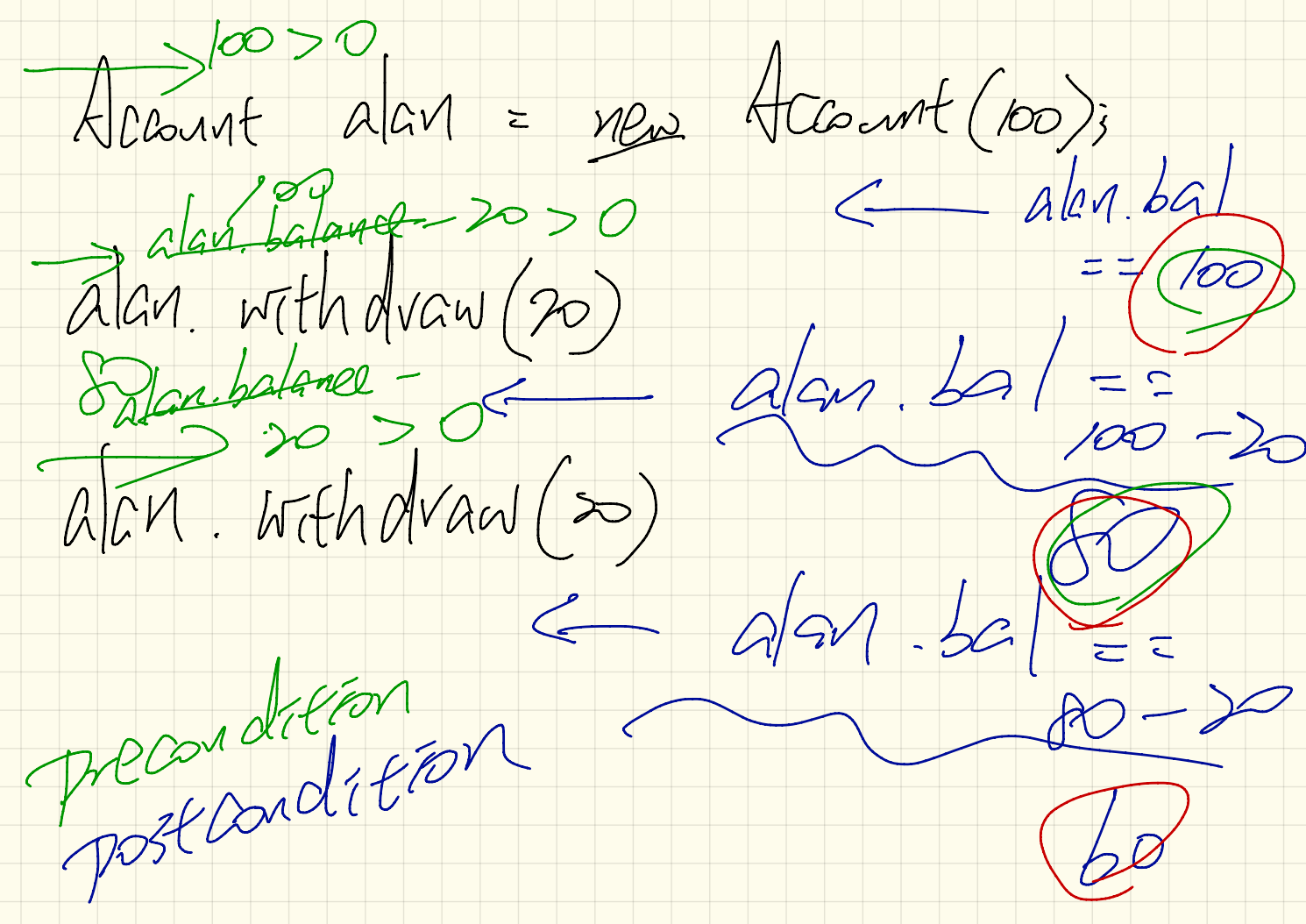
Lecture 2


withdraw
exception → balance < amount
Fix 1: change except. condition
100 bal. ≤ amo.
150
e.g. balance 100
amount 150
class invariant
Fix 2: check always that amount will update result
illegal
but not caught by precondition
balance 100
amount 100
zero
pos. balance
balance

3 kinds of contract

Lifetime of an account object




100 > 0
Account alan = new Account(100);
alan.bal == 100
alan.balance -- 20 > 0
alan.withdraw(20)
alan.bal == 100 - 20
80
alan.balance - 20 > 0
alan.withdraw(20)
alan.bal == 80 - 20
60
precondition
postcondition

precond of m1 is satisfied

$\wedge$ conjunction

obj.m1(...)

← postcond of m1 is satisfied

precond of m2 is satisfied

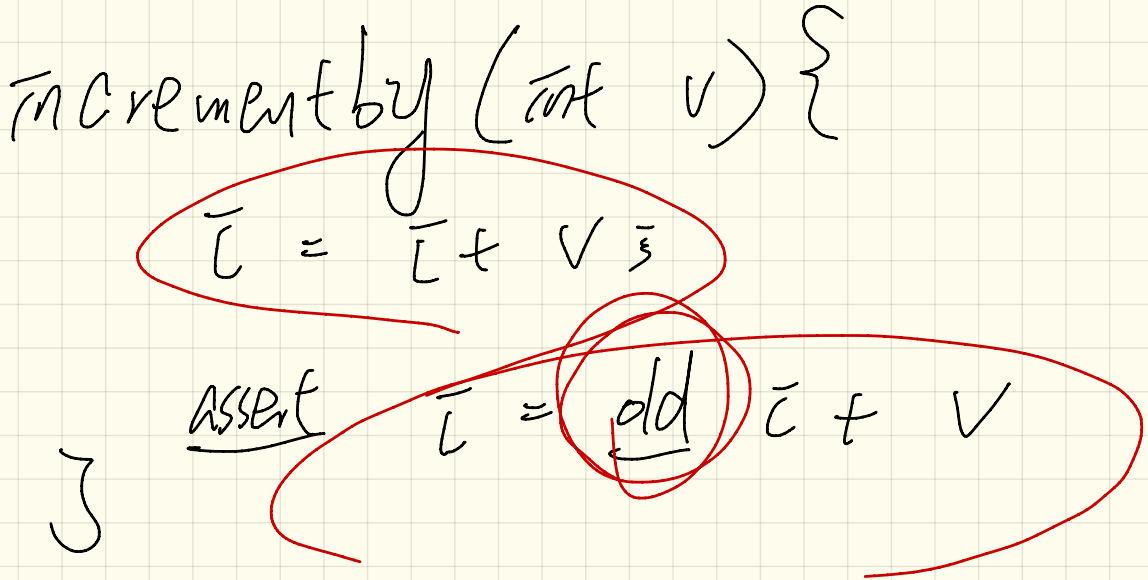
obj.m2(...)

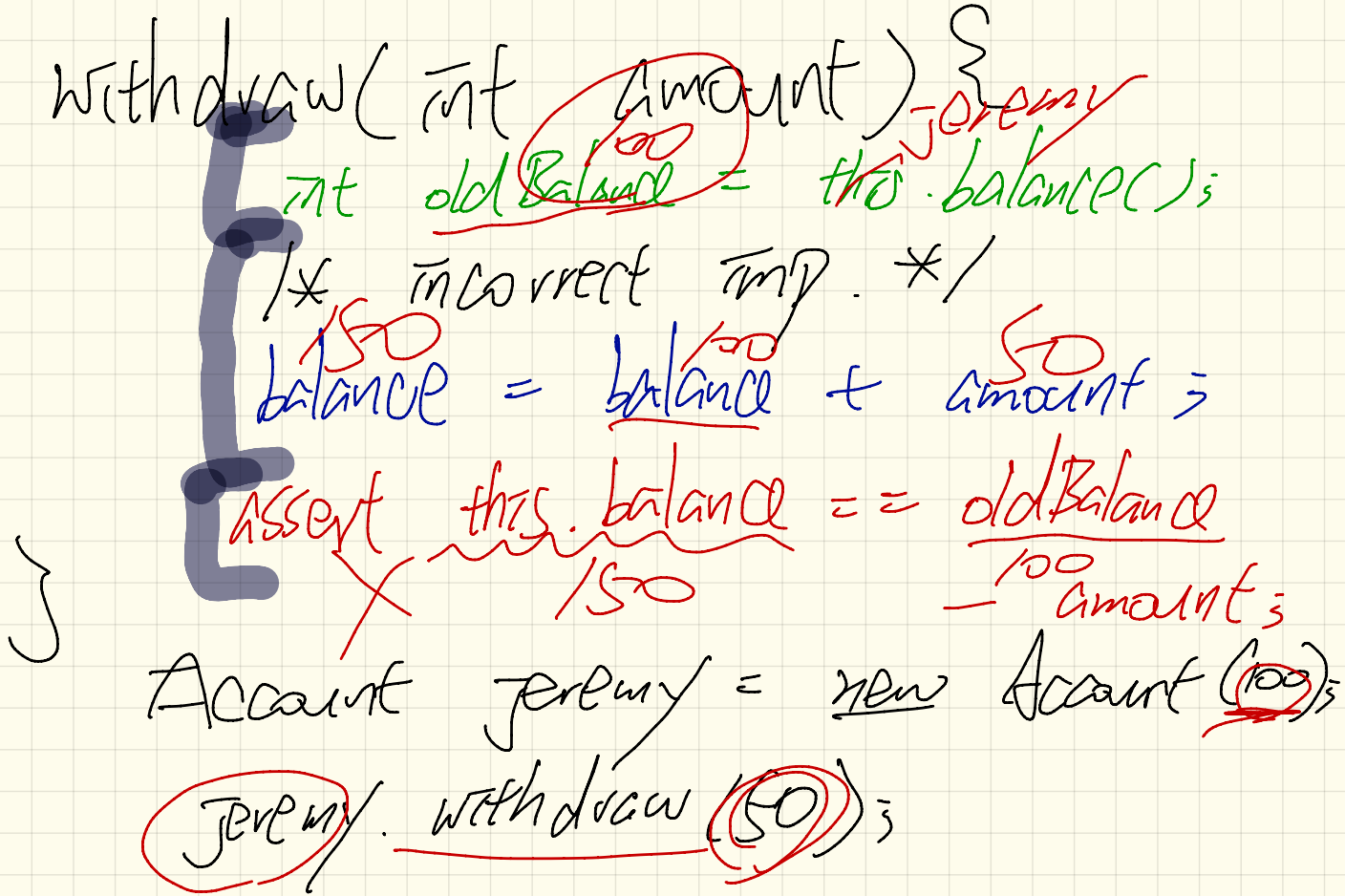
$\wedge$ INV is satisfied

← postcond of m2 is satisfied

$\wedge$ INV is satisfied

Think of INV is the common postcond. of every method


incrementby(int v){
c = c + v;
assert c = old c + v
}


withdraw(int amount) {
int oldBalance = this.balance();
/* incorrect imp. */
balance = balance + amount;
assert this.balance == oldBalance - amount;
}
Account jeremy = new Account(100);
jeremy.withdraw(50);
jeremy
100
150
50
150
100
50

void Sort (int[] x) {

implementation

quick
merge

selection
insertion

bubble sort
heap sort

post condition

$\forall i \mid 0 \le i < x.length - 1 : x[i] \le x[i+1]$

}

1. Correctness → what to achieve (contracts)

2. Efficiency → how to achieve