Test 2 Review

Friday November 22

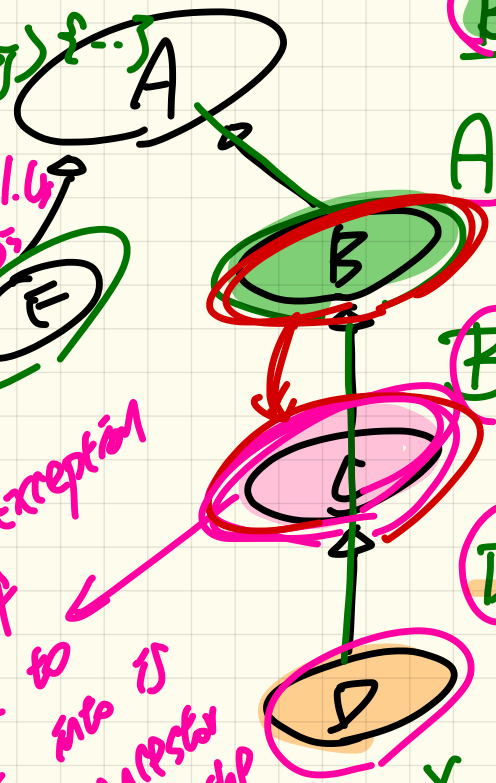
```
class Collector {
    C[] cs;
    void add(C obj) {...}
}
```

Collector c = new Coll.();
c.add((D) new B());

① valid down cast from B to D

② ClassCastExcep. ∵ DT B is not descen. of D.

ClassCastException if the type to cast is an ancestor of the DT.



B obj = new C();

A oa = (A) obj;

B ob = (A) obj;

D od = (D) obj;
down

X E oe = (E) obj;

- Inheritance
  - ↳ try eclipse demo
  - ↳ practice test
  - ↳ class Collector {
    - ↗ void add((C) oc) {...}
  - }

∴ B cannot fulfill C. Collector c = ——
X c.add(new B(2));

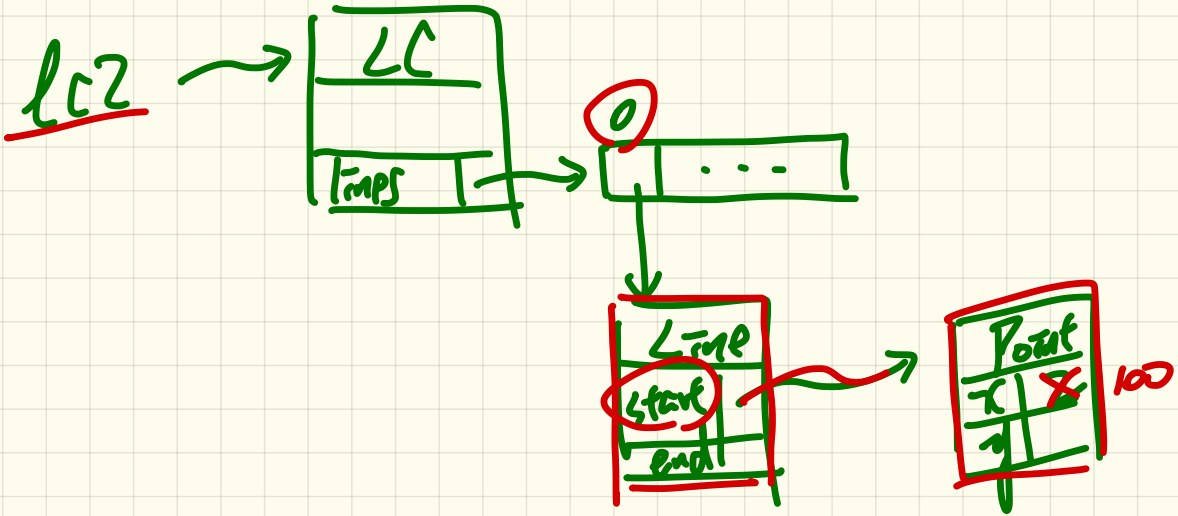
XD RECURSION.

- hashCode
- equals / call by value / collection ((C))

agg:

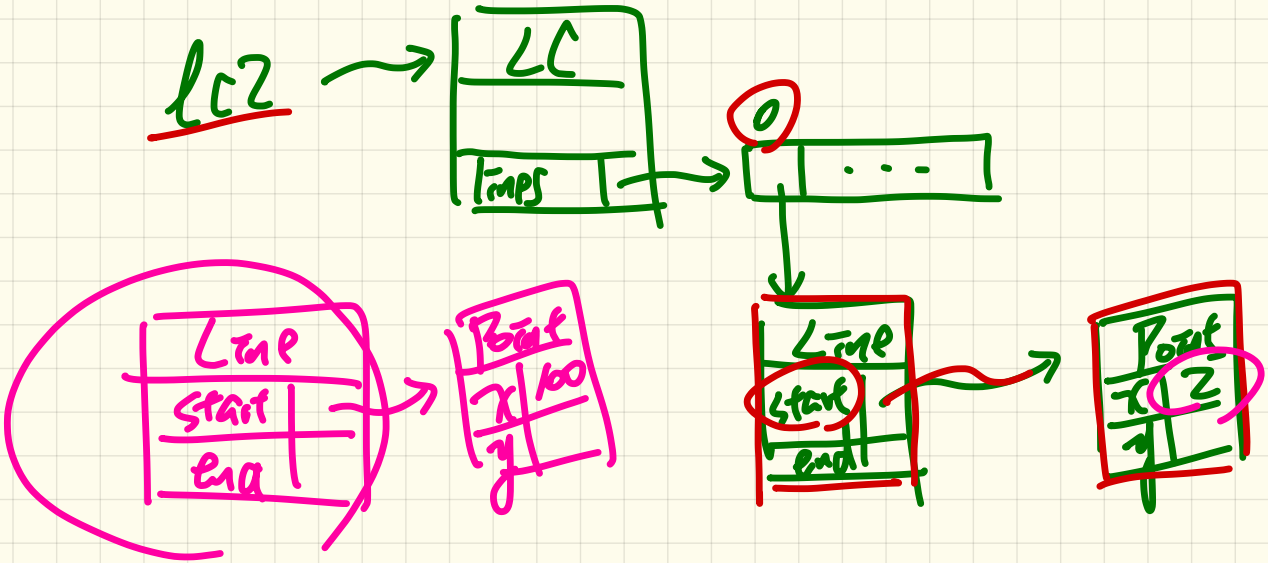
lc2.getLineAt(0).getStart().getX()

lc2.getLineAt(0).getStart().setX(100)



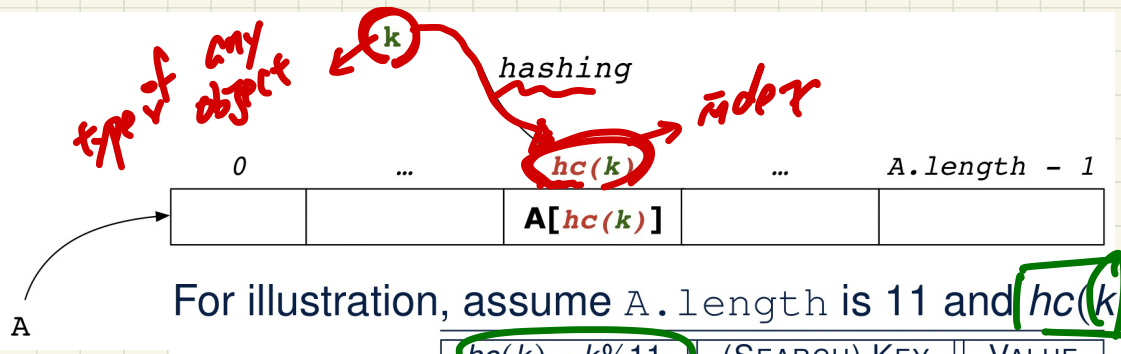
Comp: - lc2.getLineAt(0).getStart().getX()

lc2.getLineAt(0).getStart().setX(100)



Implementing a Hash Table via Hashing

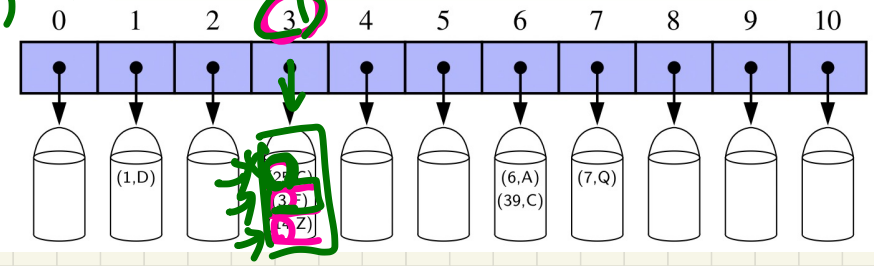
k1.equals(k2) ⇒ hc(k1) = hc(k2)



- Converting k to hc(k)
- Indexing into A[hc(k)]

For illustration, assume `A.length` is 11 and hc(k) = k%11

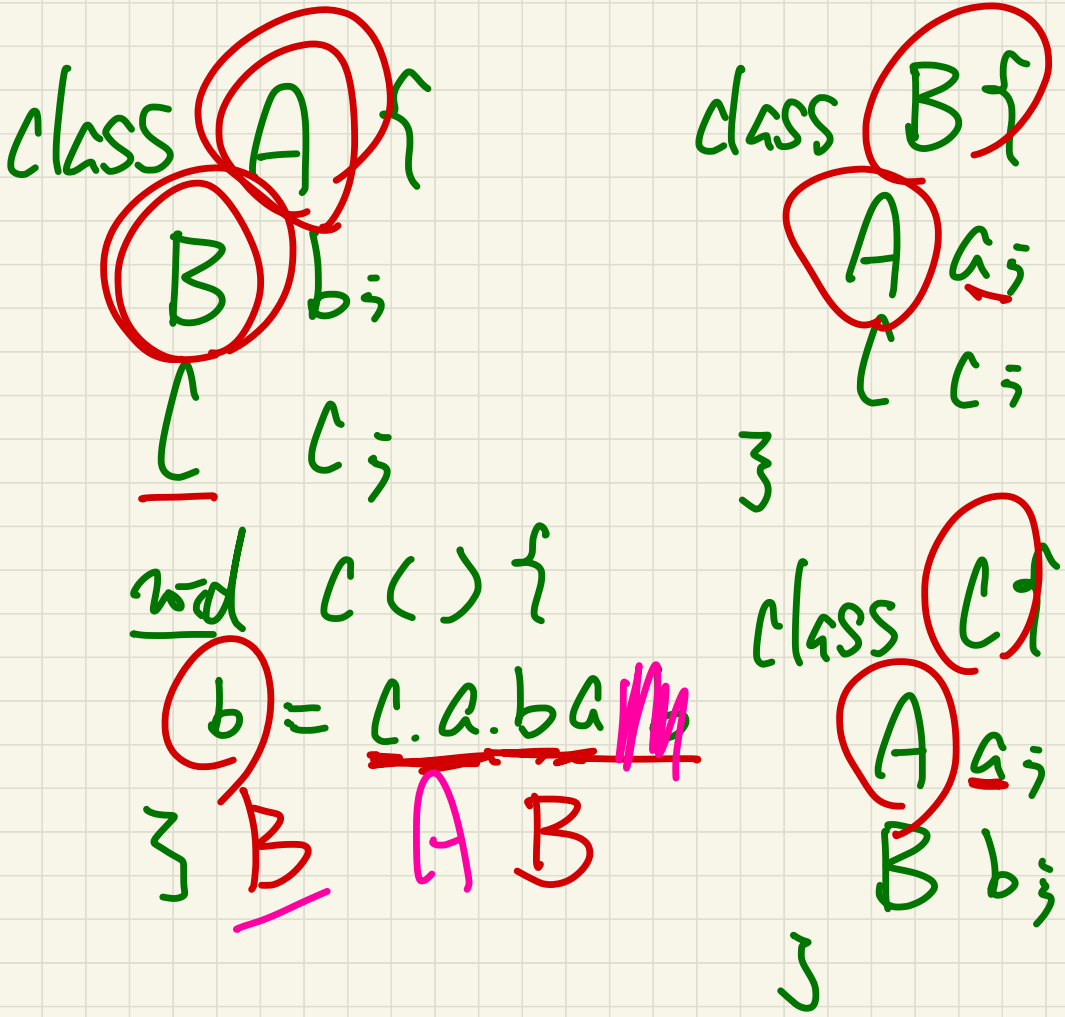
hc(k) = k%11	(SEARCH) KEY	VALUE
0	1	D
1	25	C
2	3	F
3	14	Z
4	6	A
5	39	C
6	7	Q



bucket array

search(3) hc(3) = 3

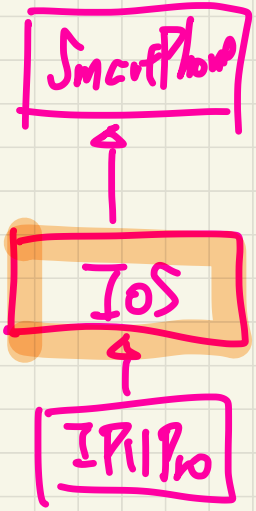
collision: distinct keys map to same hashCode


class A {
B b;
C c;
void c() {
b = c.a.b a
} B
A B
class B {
A a;
C c;
}
class C {
A a;
B b;
}

obj instanceof C

↳ TRUE if ① the dynamic type of obj can fulfill the expectations on C.

② the DT of obj is a descendant of C.




```
SmartPhone p =
    new IOS();
```

p instanceof SmartPhone (T)

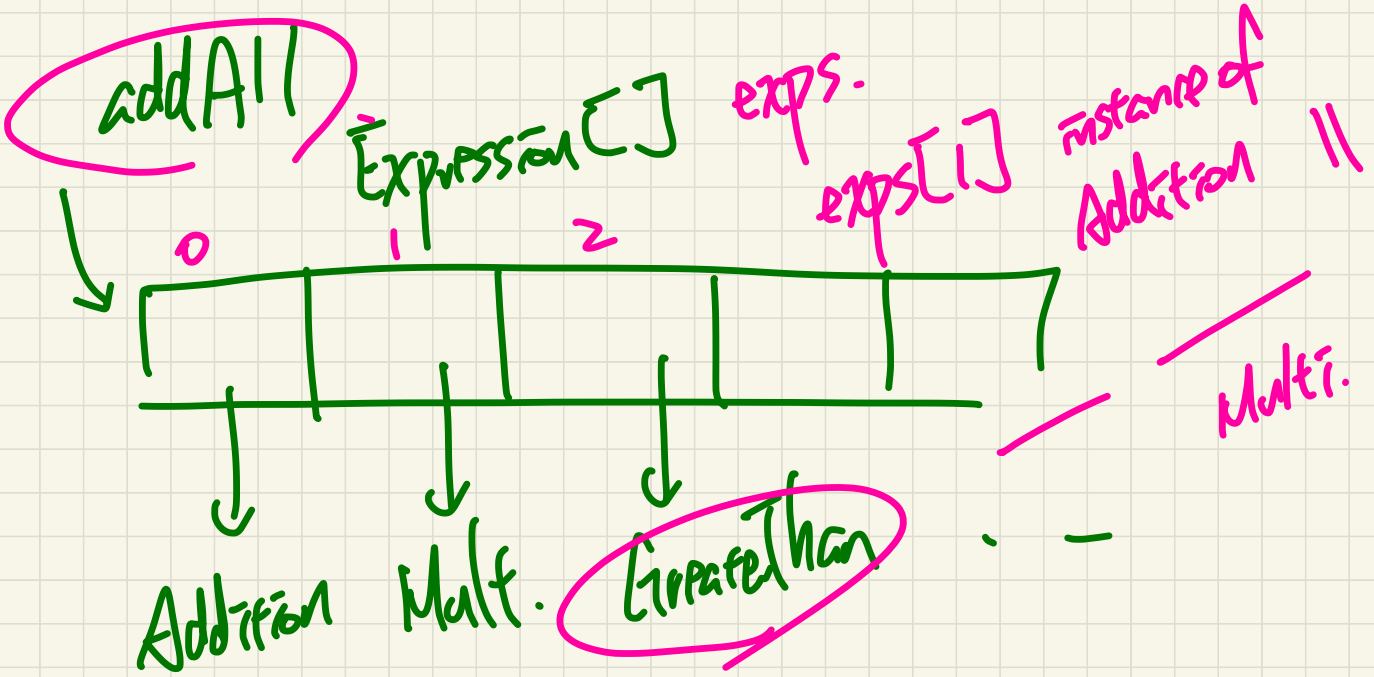
p instanceof IOS (T)

p instanceof IPhone (F)

if (obj instanceof C) {

}

- Compiles if either upward or downward cast
- CCE if the DT of obj is not a descendant class of C.


addAll
Expression[] exps.
exps[1] instanceof Addition ||
0
1
2
Addition
Mult.
GreaterThan
Multi.

(2+3) + (3*4) + (3>4)

Haskell

↳ functional programming.

Implement Map

keys & values

Naive Solution