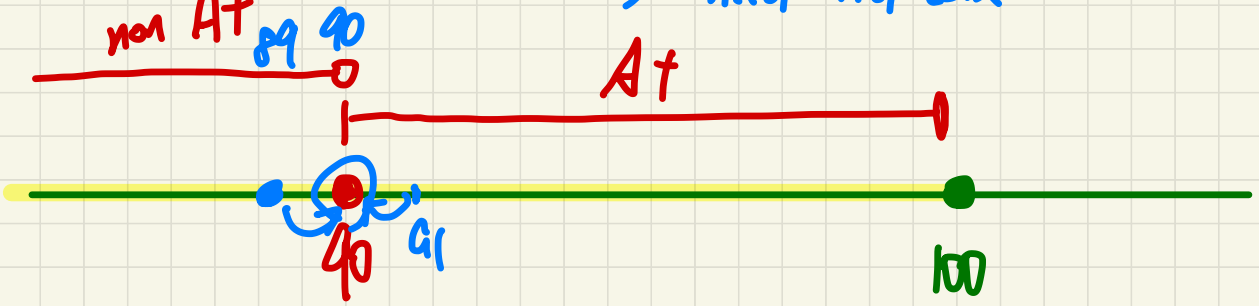
Exam Review III

Thursday December 13

@throws IllegalArgumentException when grade > 100

↳ need test case



@pre. assume input is never >100

Q. Which of the following tests would reveal an error in the imp.?

```
/**
 * @param x
 * @param y
 * @pre. 1 ≤ x ≤ 5 and -3 ≤ y ≤ 2
 */
int abs ( int x, int y ) {
    /* imp. */ return x - y;
}
```

x > y

abs(2, 1)

abs(4, 0)

Complete?

(A) abs(2, 1)

(B) abs(4, 0)

(C) abs(2, 2)

(D) abs(1, 2)

return 1 − 2 = −1

/** @pre. radius is not negative

@throws IAE when radius is negative.

*/

double area(double radius) {

/* .. */

}

< 0 ! >= 0

Which of the following tests would be considered as boundary cases?

(A) : -0.5

(B) 0.3

(C) : -0.8

(D) All.

(E) 0

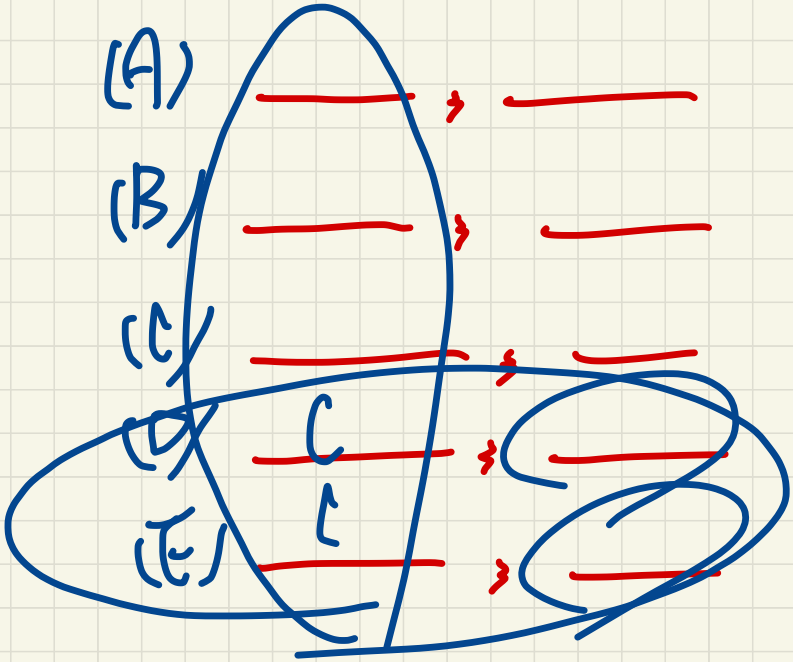
(E) None

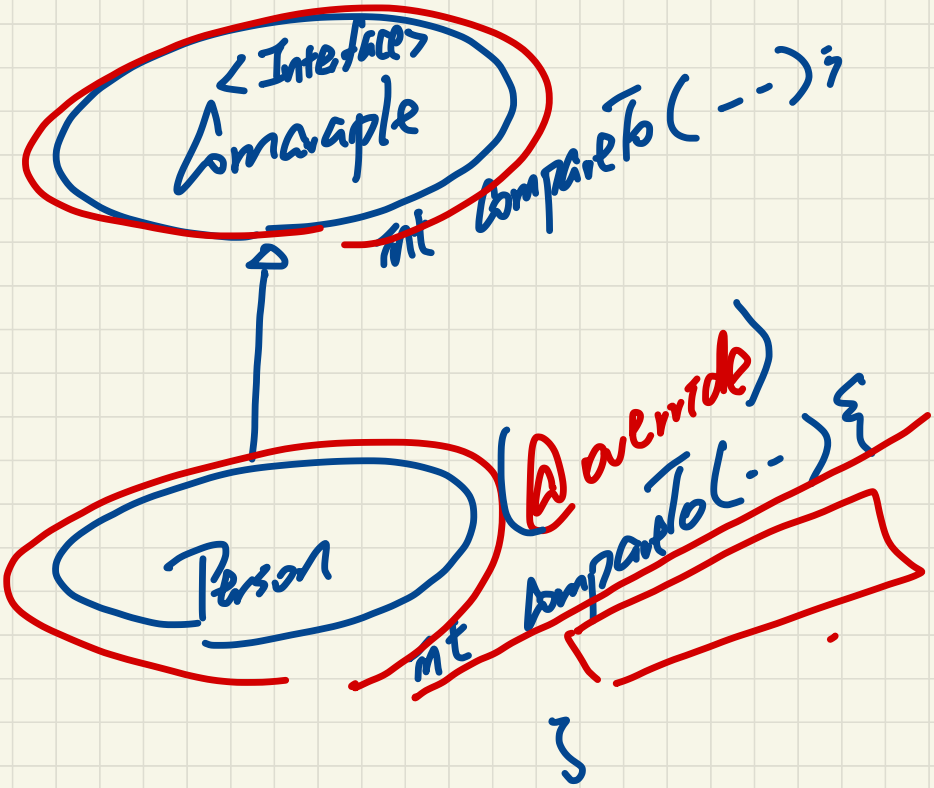
x 10.2

x.compareTo(y) must return 0 if [x is equal to y.]

x.equals(y)

(T)



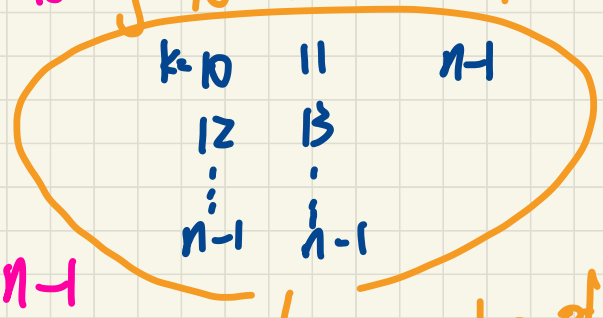

<Interface>
Comparable
int compareTo(--);
Person
@override
int compareTo(--){
}

$$\frac{r > s \qquad s > p}{r > p}$$

```
1  void prog(int[] a, int n)
2    for (int i = 0; i < n; i++) {
3      for (int j = i; j < n; j++) {
4        for (int k = j; k < n; k++) {
5          System.out.println(i * j + k);
6        }
7      }
8    }
```

$i = 0 \quad j = 0 \cdots n-1$

$i = 10 \quad j = 10 \; 11 \cdots n-1$

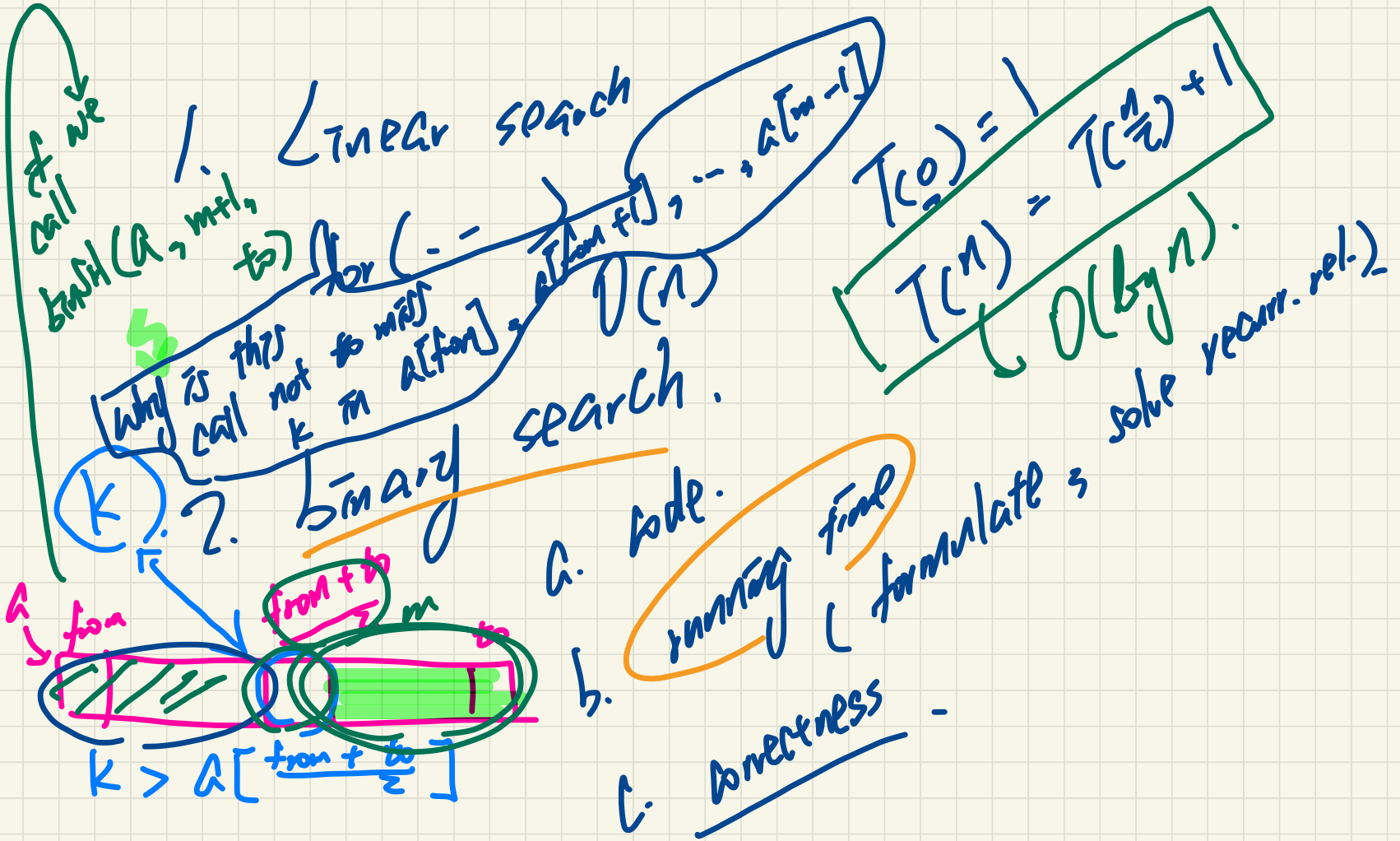


For each value of j, the # of iter. for k is $O(\frac{n}{2})$

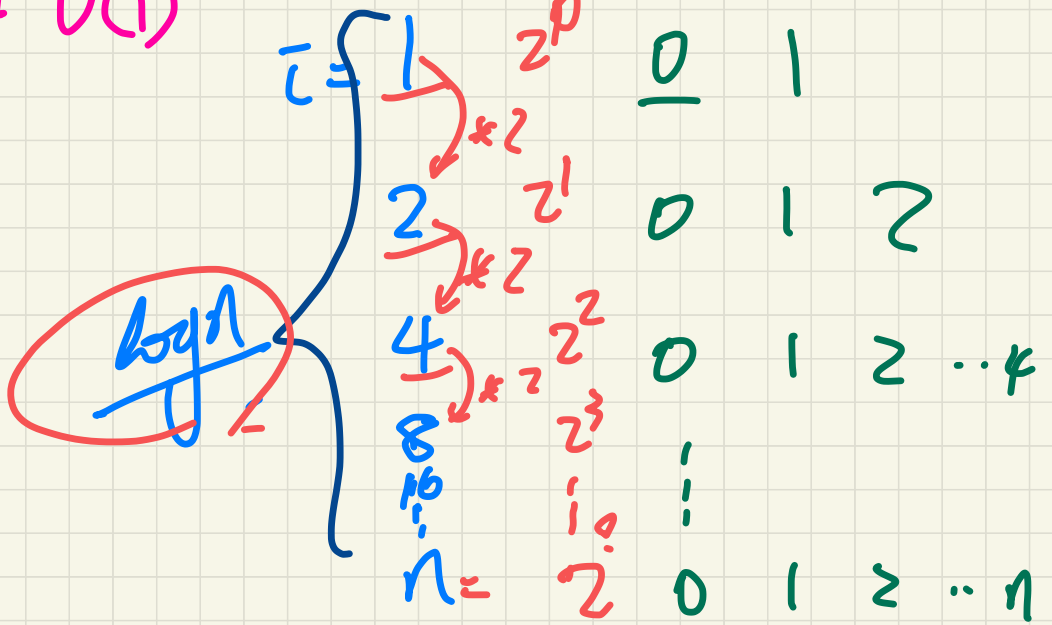
$\overbrace{n + n + \cdots + n}^{n \text{ terms}} = n^2$

$n + (n-1) + \cdots + 1 = \frac{(n+1) * n}{2}$

$O(n^2)$


if we call
bSrch(a, mid, to)
1. Linear search
for (- -)
a[from], ..., a[m-1]
O(n)
why is this call not to miss k in a[from]
T(0) = 1
T(n) = T(n/2) + 1
T(n) = O(log n).
(solve recurr. rel.)
2. binary search.
k
a.
from
from + to / 2
m
to
b. running time
c. correctness
k > a[from + to / 2]

```
for( int i=1; i<=n; i=2*i){
    for( int j=0; j<=i; j++){
        printf(...);   O(1)
    }
}
```

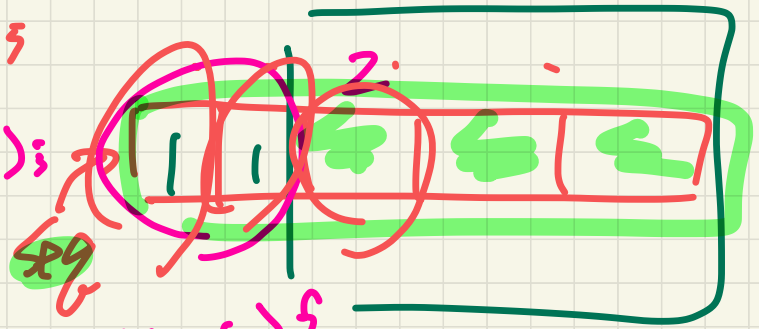


T(0) = 1
T(1) = 1
T(2) = 2
3 5 8 ...

@pre. n >= 0 → assume n is not negative,
no need to test n = -1, -2, -

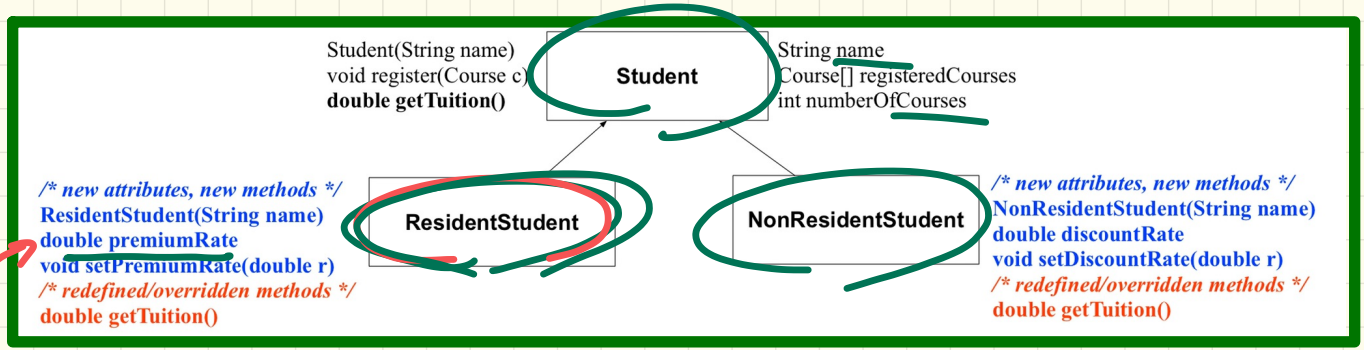
```
int[] fibo(int n) {
    int[] seq = new int[n];
    seq[0] = 1; seq[1] = 1;
    return fibH(seq, 2, seq.length-1);
}
```

T(n) = T(n-1) + 1

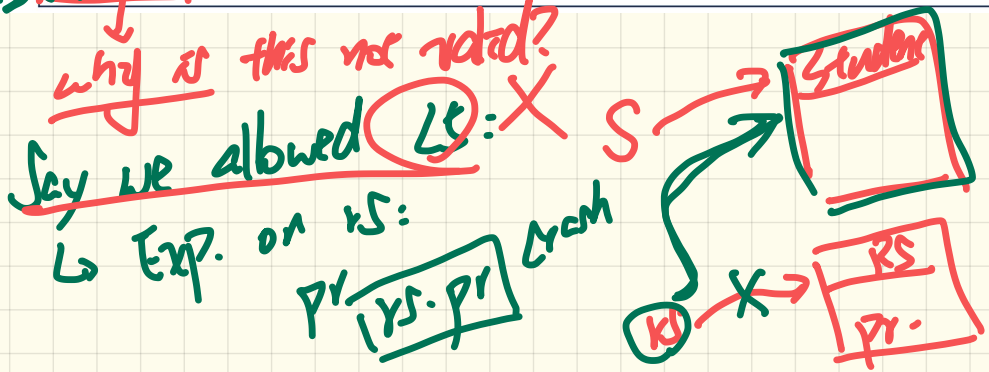


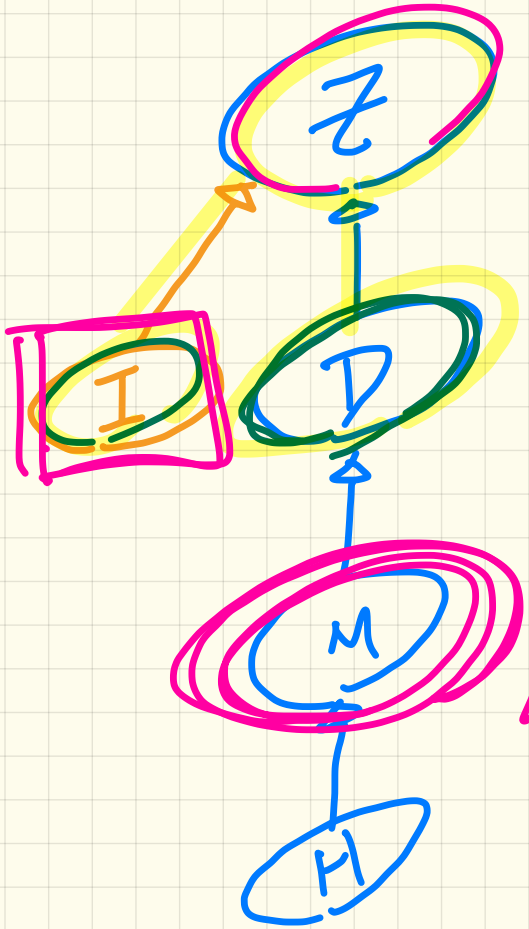

```
void fibH(int[] seq, int from, int to) {

}
```

Intuition: Polymorphism




```
1  Student s = new Student("Stella");
2  ResidentStudent rs = new ResidentStudent("Rachael");
3  rs.setPremiumRate(1.25);
4  s = rs; /* Is this valid? */
5  rs = s; /* Is this valid? */
```





D obj = new M();

I obj2 = obj; X

I obj2 = (I) obj; X

(I): neither upward nor downward

obj: S: D

compiles but CCE.

I obj2 = (I)((Z) obj)

I obj2 = (I)((Z) obj) → CCE.

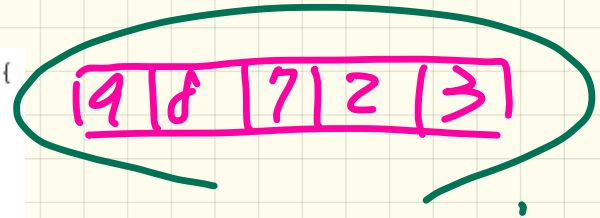
if(obj instanceof Z &&

(Z)obj instanceof I) {

obj2 = (I)((Z) obj);

}

```java
public static double avg(List<Double> x) {
    double avg = 0.0;
    int n = 0;
    for (int i = 0; i < x.size(); i++) {
        double xi = x.get(i);
        if (xi > 10.0) {
            n = n + 1;
            avg = avg + xi;
        }
    }
    return avg / n;
}
```



$\neg(\forall i \mid 0 \le i < x.\text{size} : x[i] < 10)$

Opt1. @pre. not all numbers are < 10

```java
public static double avg(List<Double> x) {
    double avg = 0.0;
    int n = 0;
    for (int i = 0; i < x.size(); i++) {
        double xi = x.get(i);
        if (xi > 10.0) {
            n = n + 1;
            avg = avg + xi;
        }
    }
    return avg / n;
}
```

Opt2. @throws IAE not all #'s are < 10

```java
public static double avg(List<Double> x) {
    double avg = 0.0;
    int n = 0;
    for (int i = 0; i < x.size(); i++) {
        double xi = x.get(i);
        if (xi > 10.0) {
            n = n + 1;
            avg = avg + xi;
        }
    }
    return avg / n;
}
```

```
if { throw IAE(...); } else { }
```

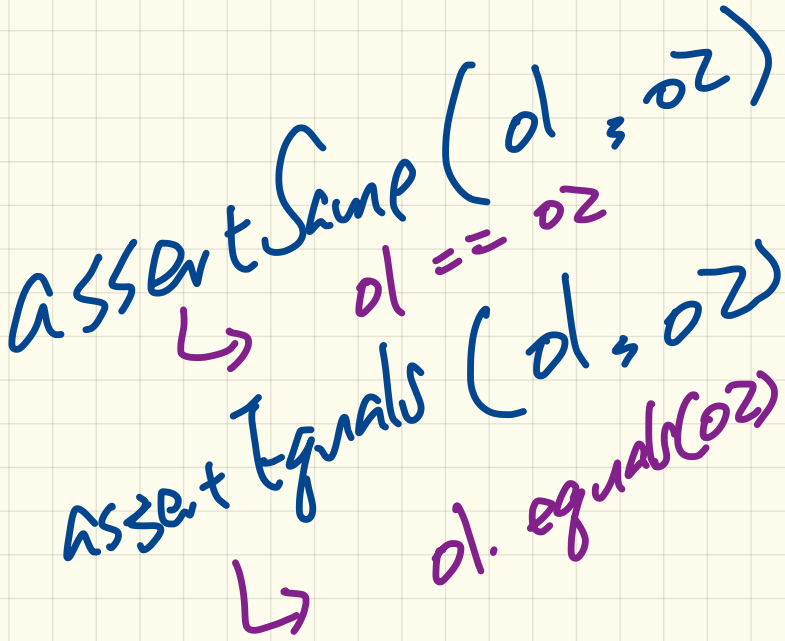
```java
public static double avg(List<Double> x) {
    double avg = 0.0;
    int n = 0;
    for (int i = 0; i < x.size(); i++) {
        double xi = x.get(i);
        if (xi > 10.0) {
            n = n + 1;
            avg = avg + xi;
        }
    }
    return avg / n;
}
```

if (n == 0) {
throw new IAE
} else {
return avg / n;
}

```java
public static double avg(List<Double> x) {
    double avg = 0.0;
    int n = 0;
    for (int i = 0; i < x.size(); i++) {
        double xi = x.get(i);
        if (xi > 10.0) {
            n = n + 1;
            avg = avg + xi;
        }
    }
    return avg / n;
}
```

if (n == 0) {
n = 1;
}
return avg / n;
0
1

0.


assertSame(o1, o2)
o1 == o2
assertEquals(o1, o2)
o1.equals(o2)