Monday Oct. 1

Lecture 8

Employees:

name	id	salary
alan	2	4500.34
mark	3	3450.67
tom	1	3450.67

List 1: tom alan mark

List 2: alan tom mark

highest salary

Alternatively

"smallest" object in the list

mark 3 alan 2 tom 1

Sorting based on id's:

List 1 → tom alan mark

comp

smaller if id smaller

larger comes first

→ Sorting based on salaries and id's:

smaller comes first

List 2 → alan tom mark

Comparable Employee: Version I

```
class CEmployee1 implements Comparable<CEmployee1> {
  ... /* attributes, constructor, mutator similar to Employee */
  @Override
  public int compareTo(CEmployee1 e) { return this.id - e.id; }
}
```

```
@Test
public void testComparableEmployees_1() {
  /*
   * CEmployee1 implements the Comparable interface.
   * Method compareTo compares id's only.
   */
  CEmployee1 alan = new CEmployee1(2);
  CEmployee1 mark = new CEmployee1(3);
  CEmployee1 tom = new CEmployee1(1);
  alan.setSalary(4500.34);
  mark.setSalary(3450.67);
  tom.setSalary(3450.67);
  CEmployee1[] es = {alan, mark, tom};
  /* When comparing employees,
   * their salaries are irrelevant.
   */
  Arrays.sort(es);
  CEmployee1[] expected = {tom, alan, mark};
  assertArrayEquals(expected, es);
}
```

Comparable Employee: Version 2.1

```
class CEmployee2  implements Comparable<CEmployee2> {
  ... /* attributes, constructor, mutator similar to Employee */
  @Override
  public int compareTo(CEmployee2 other) {
    int salaryDiff = Double.compare(this.salary, other.salary);
    int idDiff = this.id - other.id;
    if(salaryDiff != 0) { return -salaryDiff; }
    else { return idDiff; } } }
```

Double.compare(alan.salary, mark.salary);

Double

without this, alan will appear later than mark in the list.

```
@Test
public void testComparableEmployees_2() {
  /*
   * CEmployee2 implements the Comparable interface.
   * Method compareTo first compares salaries, then
   * compares id's for employees with equal salaries.
   */
  CEmployee2 alan = new CEmployee2(2);
  CEmployee2 mark = new CEmployee2(3);
  CEmployee2 tom = new CEmployee2(1);
  alan.setSalary(4500.34);
  mark.setSalary(3450.67);
  tom.setSalary(3450.67);
  CEmployee2[] es = {alan, mark, tom};
  Arrays.sort(es);
  CEmployee2[] expected = {alan, tom, mark};
  assertArrayEquals(expected, es);
}
```

alan < mark

Comparable Employee: Version 2.2

```
class CEmployee2 implements Comparable<CEmployee2> {
  ... /* attributes, constructor, mutator similar to Employee */
  @Override
  public int compareTo(CEmployee2 other) {
    if(this.salary > other.salary) {
      return -1;
    }
    else if (this.salary < other.salary) {
      return 1;
    }
    else { /* equal salaries */
      return this.id - other.id;
    }
  }
```

alan

mark.id − tom.id

alan.compareTo(mark); → −1
"alan < mark"

alan.compareTo(tom); → −1
"alan < tom"

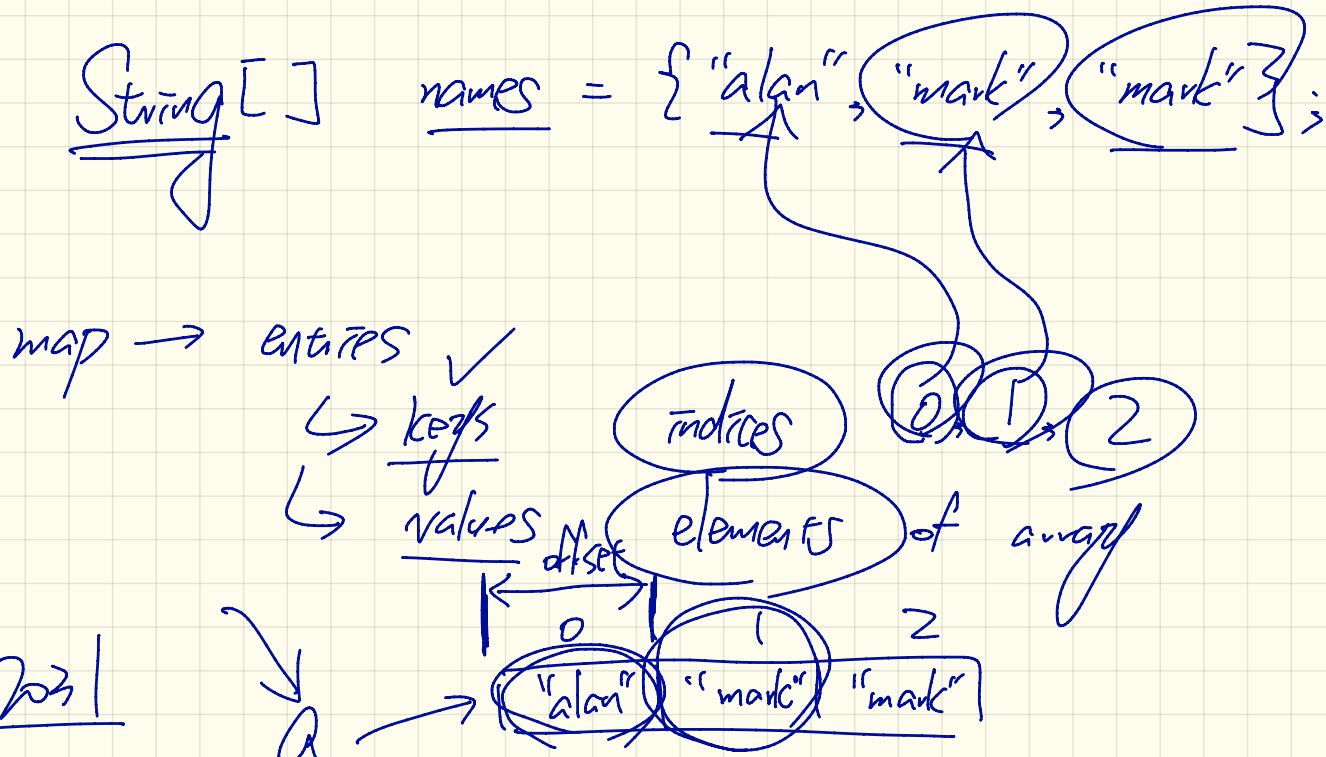
mark.compareTo(tom); → 2
this, other
"mark > tom"

r > s
s > p
⇒ r > p ✗

larger salary
↳ occur earlier in the sorted list
↳ considered as "smaller"

```
@Test
public void testComparableEmployees_2() {
  /*
   * CEmployee2 implements the Comparable interface.
   * Method compareTo first compares salaries, then
   * compares id's for employees with equal salaries.
   */
  CEmployee2 alan = new CEmployee2(2);
  CEmployee2 mark = new CEmployee2(3);
  CEmployee2 tom = new CEmployee2(1);
  alan.setSalary(4500.3);
  mark.setSalary(3450.6);
  tom.setSalary(3450.6);
  CEmployee2[] es = {alan, mark, tom};
  Arrays.sort(es);
  CEmployee2[] expected = {alan, tom, mark};
  assertArrayEquals(expected, es);
}
```

String[] names = {"alan", "mark", "mark"};



Implementing a Map using an Array

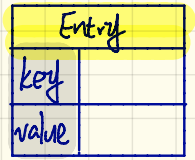
Entry	
(Search) Key	Value
1	D
25	C
3	F
14	Z
6	A
39	C
7	Q

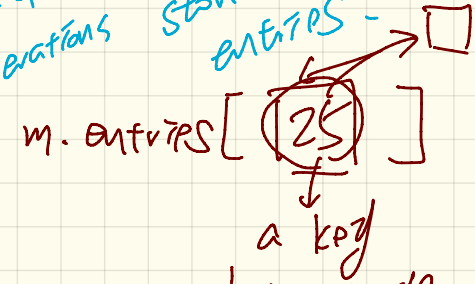
Worst Case:

$y = x$

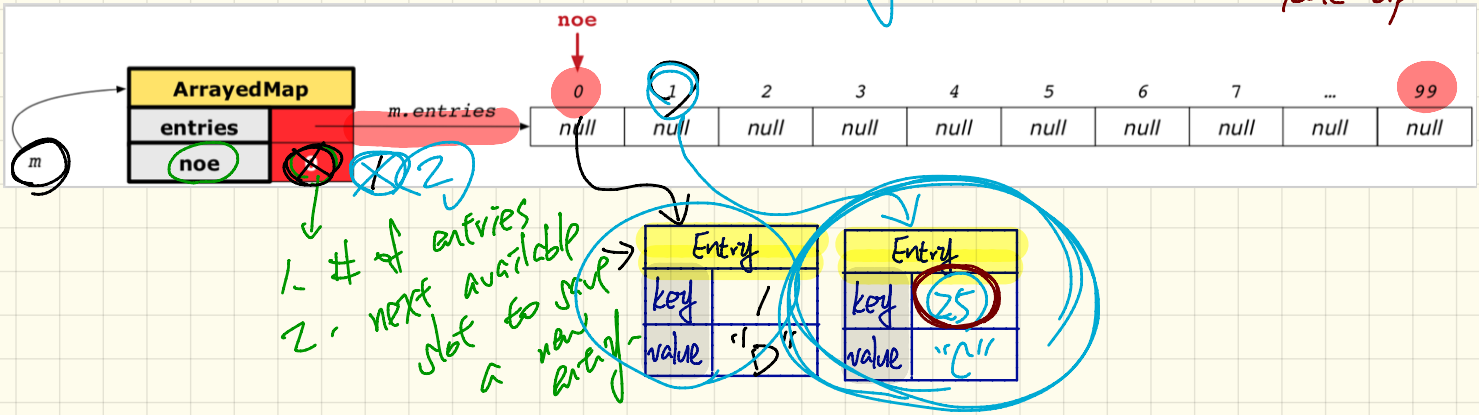
y = # of iterations

x = # of stored entries





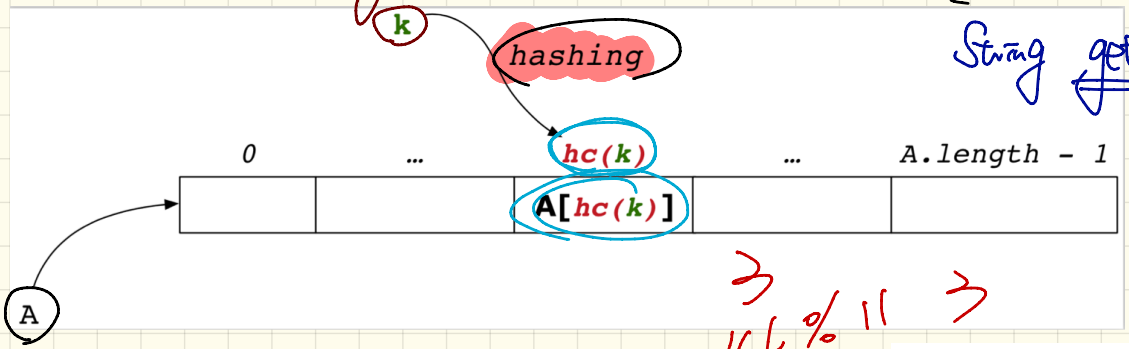
m.entries[0] = m.get(25)



Hashing → m.get(1)
→ m.get(3)

key is 3

A[0] → efficient.

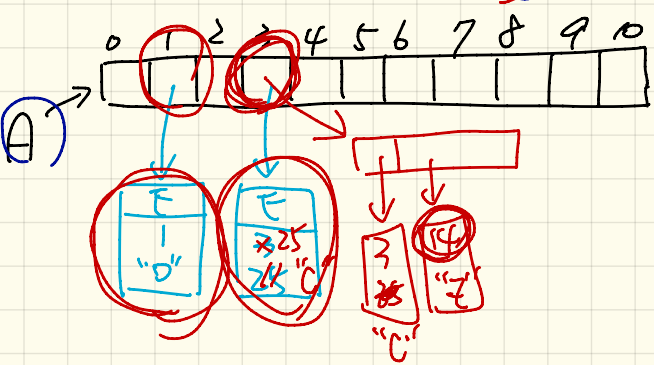


String get(int key){
return A[key % 11];
}

1 % 11 (1)
3 % 11 (3)

3
14 % 11 3

Say A.length is 11 and
hc(k) = k % 11



hc(k)	ENTRY (SEARCH) KEY	VALUE
1	1	D
3	25	C
	3	F
	14	Z
	6	A
	39	C
	7	Q