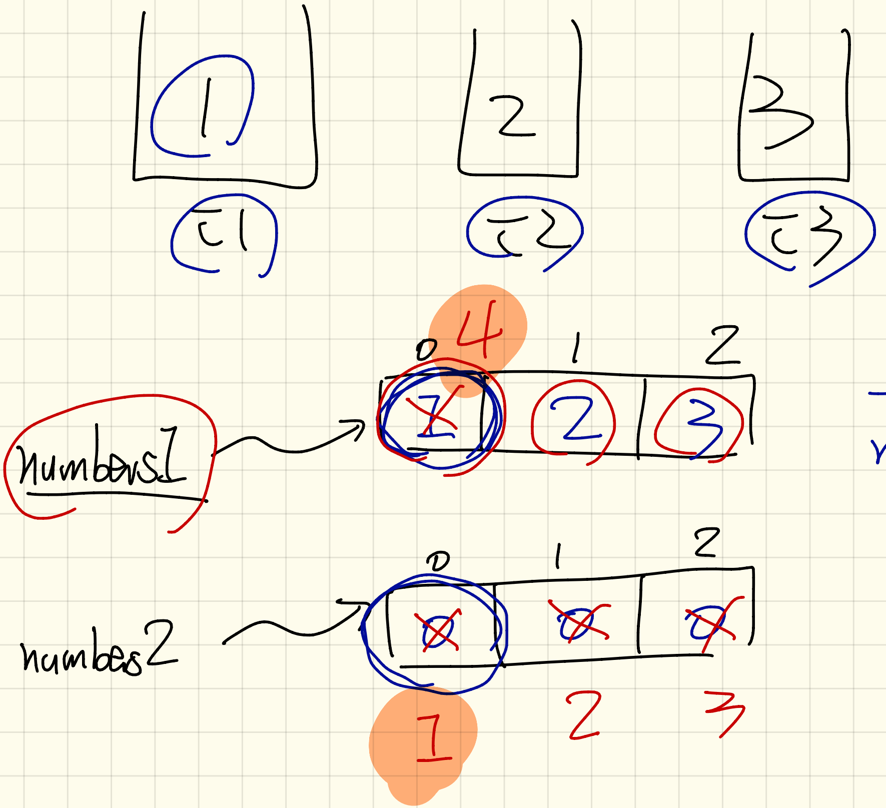
Monday April 2

Lecture 12

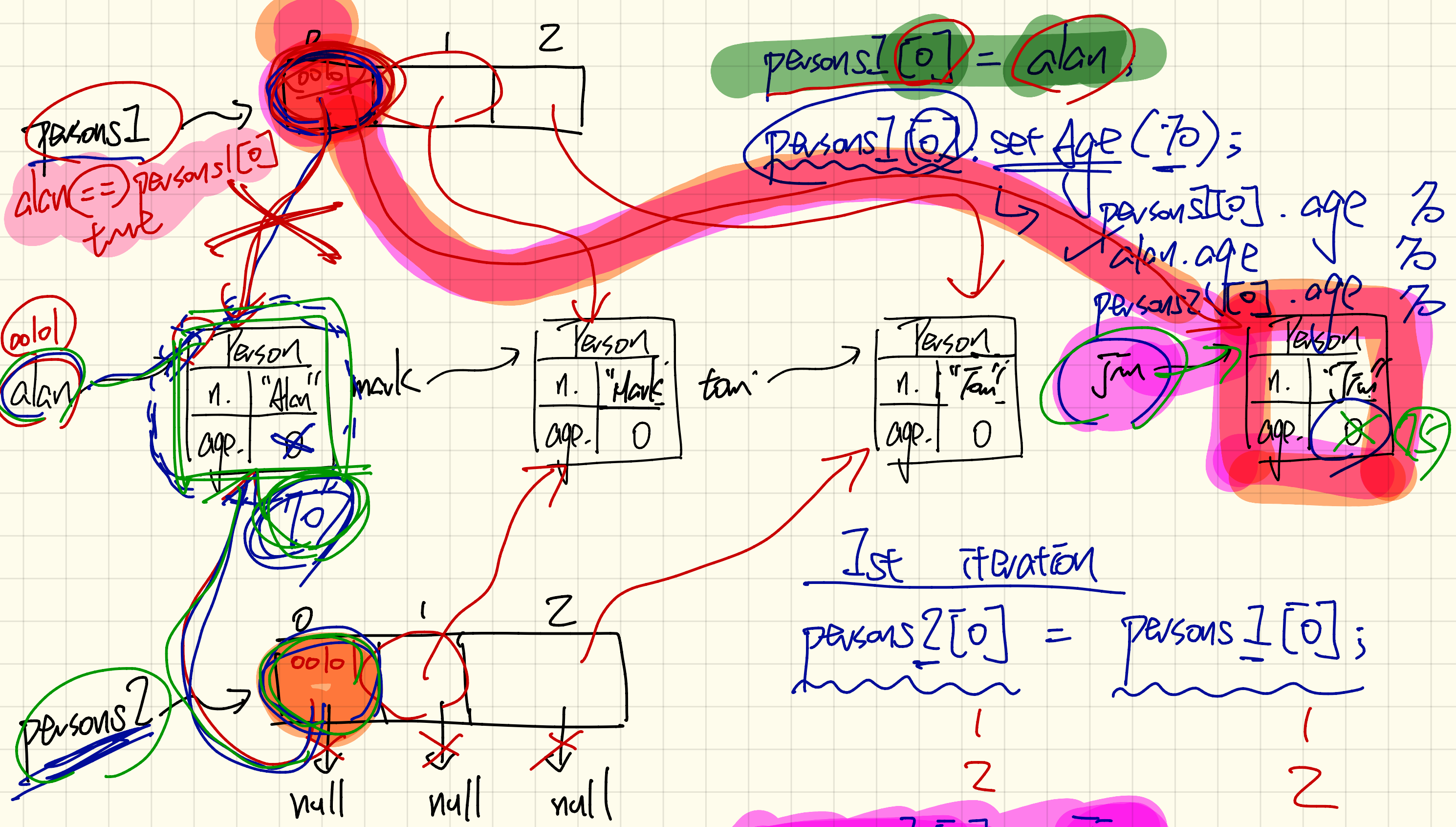


1st iteration

numbers2[0] = numbers1[0];

1 1

2 2



persons1[0] = alan;

persons1[0].setAge(70);

↳ persons1[0].age 70

✓ alan.age 70

persons2[0].age 70

1st iteration

persons2[0] = persons1[0];

1 1

2 2

persons1[0] = jim;

↳ persons1[0] == jim

persons1[0].setAge(75);

alan == persons1[0] true

persons1[0] == persons2[0] true

Static (modifier)

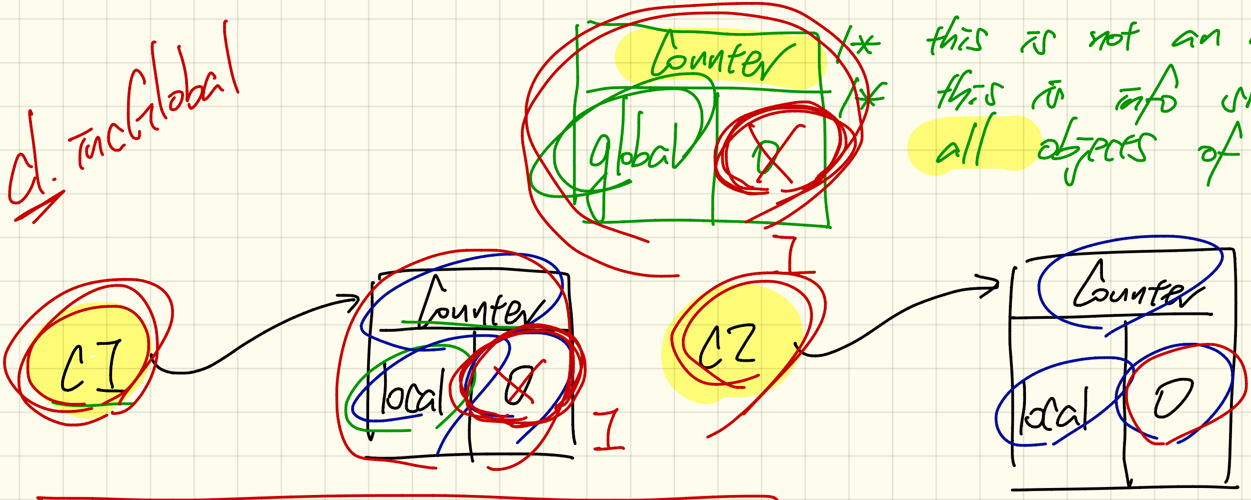
local variable
↳ to a specific context object

______ int i ; non-static variable

static int j ; static variable

global variable
↳ to all context objects.

c1.incGlobal



/* this is not an object */
/* this is info shared by all objects of Counter */

c1.global	✓	c1.local	✓
c2.global	✓	c2.local	✓
Counter.global	✓	Counter.local	✗

Parameters vs. Arguments

Developer/Supplier

```
class MyClass {
  int i;
  int getValue() {
  }
  void setValue(int j) {
    i = j;
  }
}
```

signature

input/parameter (variable)

User/Client

```
class MyClassUser {
  ______ main(...) {
    MyClass o1 = new MyClass();
    o1.setValue(3);
  }
}
```

argument (value).

```
Math.abs(-2.5);
```

static method

```
Math m = new Math();
m.abs(-2.5);
```

overloading.

```
abs (double a)
abs (float a)
abs (int a)
abs (long a)
```

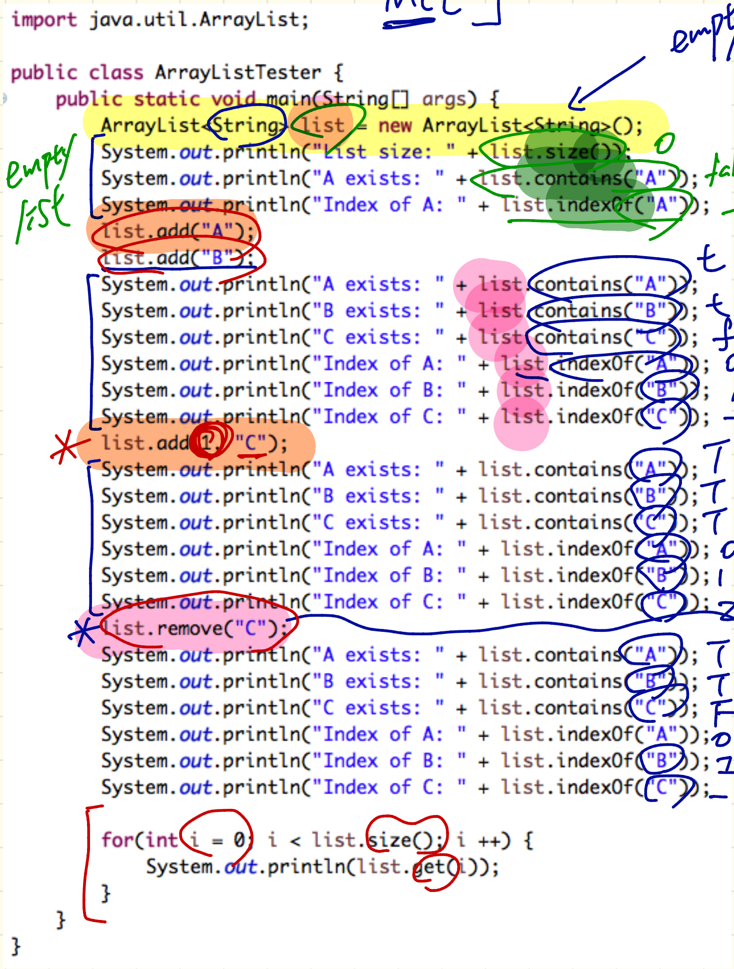
When multiple methods have the same name, then:

1. they have different numbers of paramters

2. same # of paramatas, but different types.

```
class SMS {
    void addStudent(Student s) { -- }
                      1 parameter
    void addStudent(String name, int marks){--}
                        2 parameters.
    void addStudent(String s);
}
```

Tester of ArrayList



```
import java.util.ArrayList;

public class ArrayListTester {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>();
        System.out.println("List size: " + list.size());
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("Index of A: " + list.indexOf("A"));
        list.add("A");
        list.add("B");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));
        list.add(1, "C");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));
        list.remove("C");
        System.out.println("A exists: " + list.contains("A"));
        System.out.println("B exists: " + list.contains("B"));
        System.out.println("C exists: " + list.contains("C"));
        System.out.println("Index of A: " + list.indexOf("A"));
        System.out.println("Index of B: " + list.indexOf("B"));
        System.out.println("Index of C: " + list.indexOf("C"));

        for(int i = 0; i < list.size(); i ++) {
            System.out.println(list.get(i));
        }
    }
}
```

int[ ]

empty list

empty list

0

false

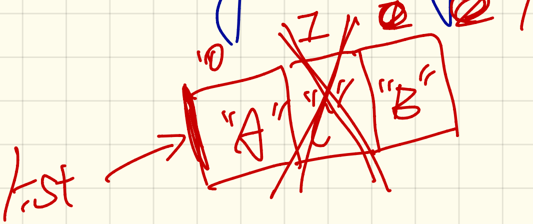
-1

t
t
f
0
1
-1

T
T
T
0
1
2

T
T
F
0
1
-1

ArrayList<String> list =
new ArrayList<String>( );



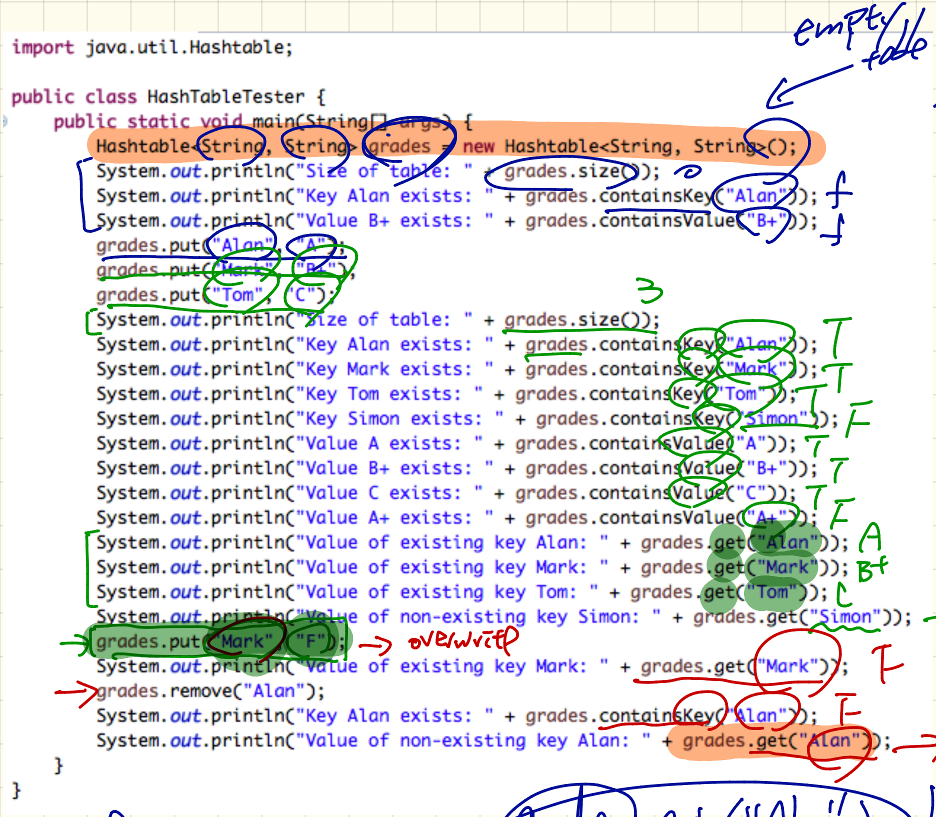
list.remove("B");

Person

ArrayList< >

class name

Tester of HashTable



```
import java.util.Hashtable;

public class HashTableTester {
    public static void main(String[] args) {
        Hashtable<String, String> grades = new Hashtable<String, String>();
        System.out.println("Size of table: " + grades.size());
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Value B+ exists: " + grades.containsValue("B+"));
        grades.put("Alan", "A");
        grades.put("Mark", "B+");
        grades.put("Tom", "C");
        System.out.println("Size of table: " + grades.size());
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Key Mark exists: " + grades.containsKey("Mark"));
        System.out.println("Key Tom exists: " + grades.containsKey("Tom"));
        System.out.println("Key Simon exists: " + grades.containsKey("Simon"));
        System.out.println("Value A exists: " + grades.containsValue("A"));
        System.out.println("Value B+ exists: " + grades.containsValue("B+"));
        System.out.println("Value C exists: " + grades.containsValue("C"));
        System.out.println("Value A+ exists: " + grades.containsValue("A+"));
        System.out.println("Value of existing key Alan: " + grades.get("Alan"));
        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
        System.out.println("Value of existing key Tom: " + grades.get("Tom"));
        System.out.println("Value of non-existing key Simon: " + grades.get("Simon"));
        grades.put("Mark", "F");
        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
        grades.remove("Alan");
        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
        System.out.println("Value of non-existing key Alan: " + grades.get("Alan"));
    }
}
```

empty table

0

f

f

3

T

T

T

F

T

T

T

F

A

B+

C

null

overwrite

F

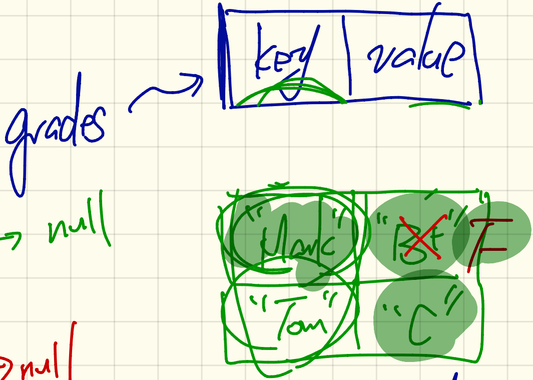
F

null

HT<String, String> grades

type for keys

type for values.



String.

grades.get("Alan").charAt(2);

null

"One key corresponds to one value"