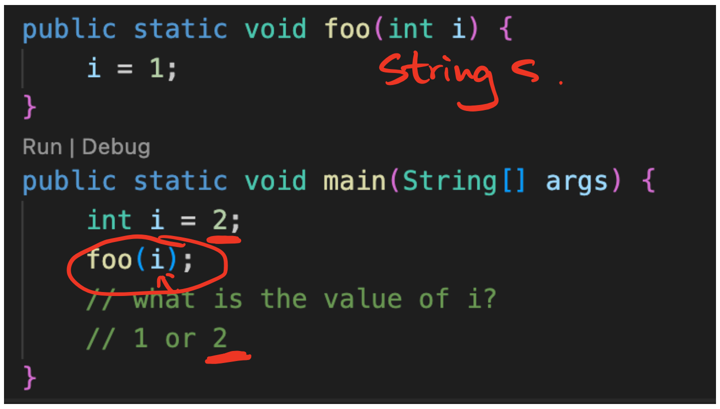
Data Structure Recitation 2

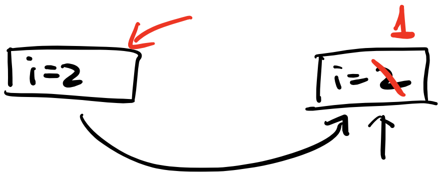
- ☐ 1. References and pass-by-value
- ☐ 2. Review for singly-linked list.
- ☐ 3. Equivalence testing for linked list (§ 3.5.2)

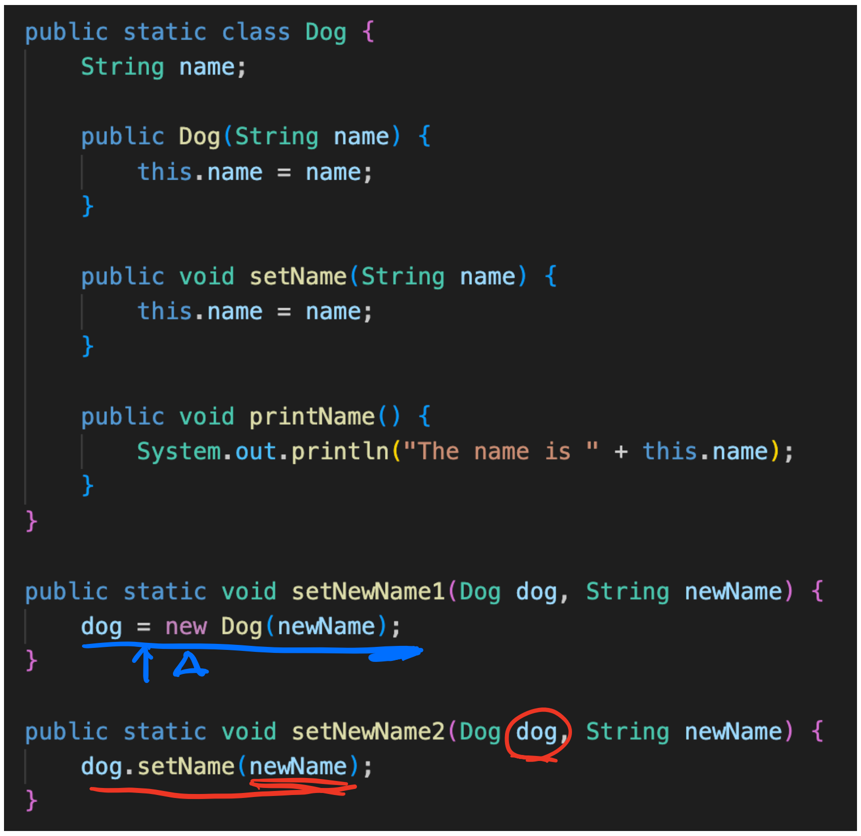
1. References and pass-by-value.



```java
public static void foo(int i) {
    i = 1;
}
Run | Debug
public static void main(String[] args) {
    int i = 2;
    foo(i);
    // what is the value of i?
    // 1 or 2
}
```

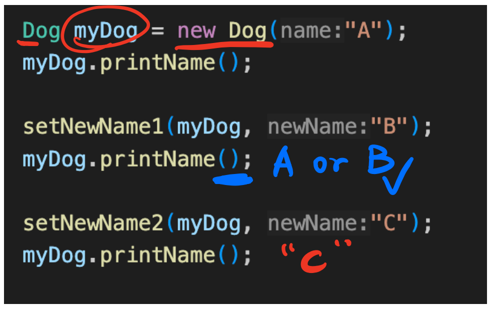
int i=2;





```java
public static class Dog {
    String name;

    public Dog(String name) {
        this.name = name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void printName() {
        System.out.println("The name is " + this.name);
    }
}

public static void setNewName1(Dog dog, String newName) {
    dog = new Dog(newName);
}

public static void setNewName2(Dog dog, String newName) {
    dog.setName(newName);
}
```

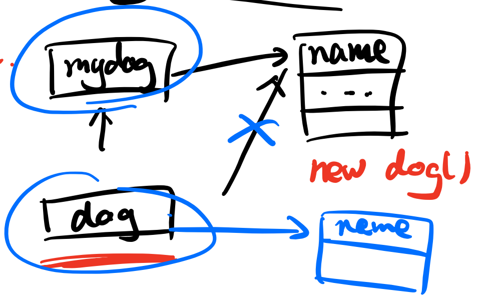
int
char
boolean
or bool



```java
Dog myDog = new Dog(name:"A");
myDog.printName();

setNewName1(myDog, newName:"B");
myDog.printName();

setNewName2(myDog, newName:"C");
myDog.printName();
```

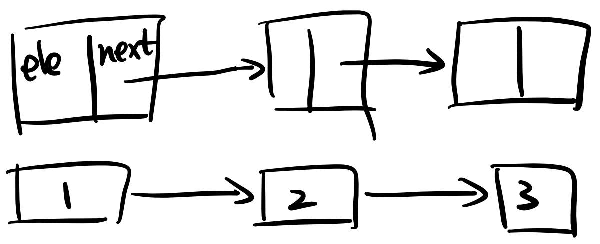
Dog ≠ int
↑ objects ↑ primitive

Dog mydog;



2. Review for singly linked list

Node.
 └ int ele
 └ Node next



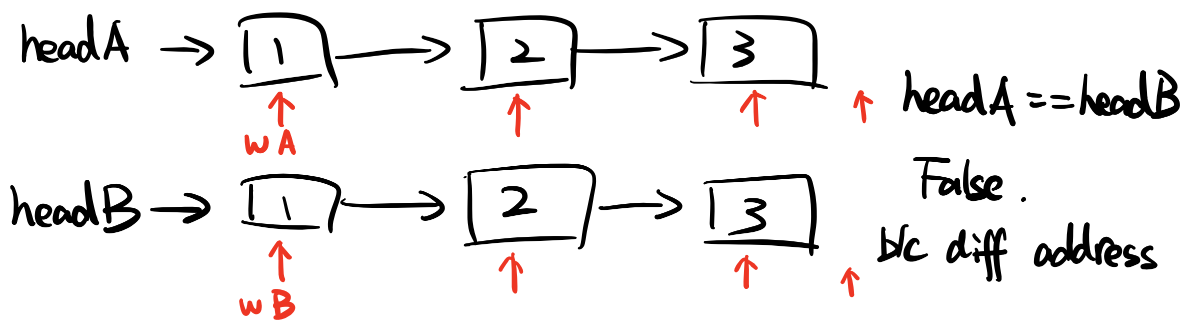
general guidelines.

1: draw a graph

2: figure out which edges to change

3: think of corner cases.

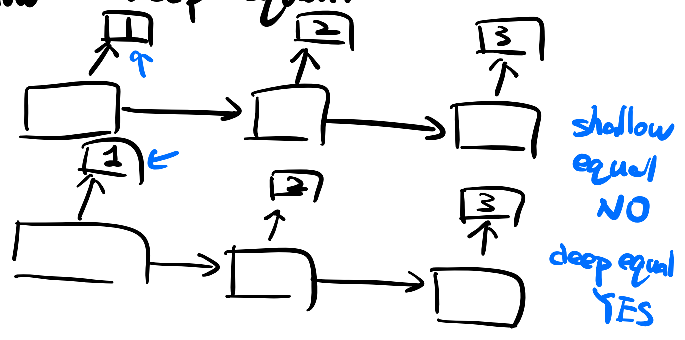
3. Equivalence testing for linked list (§ 3.5.2)



```
public boolean equals(Object o) {
  if (o == null) return false;
  if (getClass( ) != o.getClass( )) return false;
  SinglyLinkedList other = (SinglyLinkedList) o;       // use nonparameterized type
  if (size != other.size) return false;
  Node walkA = head;                                   // traverse the primary list
  Node walkB = other.head;                             // traverse the secondary list
  while (walkA != null) {
    if (!walkA.getElement( ).equals(walkB.getElement( ))) return false; //mismatch
    walkA = walkA.getNext( );
    walkB = walkB.getNext( );
  }
  return true;    // if we reach this, everything matched successfully
}
```

Code Fragment 3.19: Implementation of the SinglyLinkedList.equals method.

shallow & deep equal.


1
2
3
1
2
3
shallow equal
NO
deep equal
YES