



References in Java

CIS 110 (3/24/2021)

**For non-primitive types,
variable assignment works by
passing a reference.**



What's a reference?

harry	
-------	--

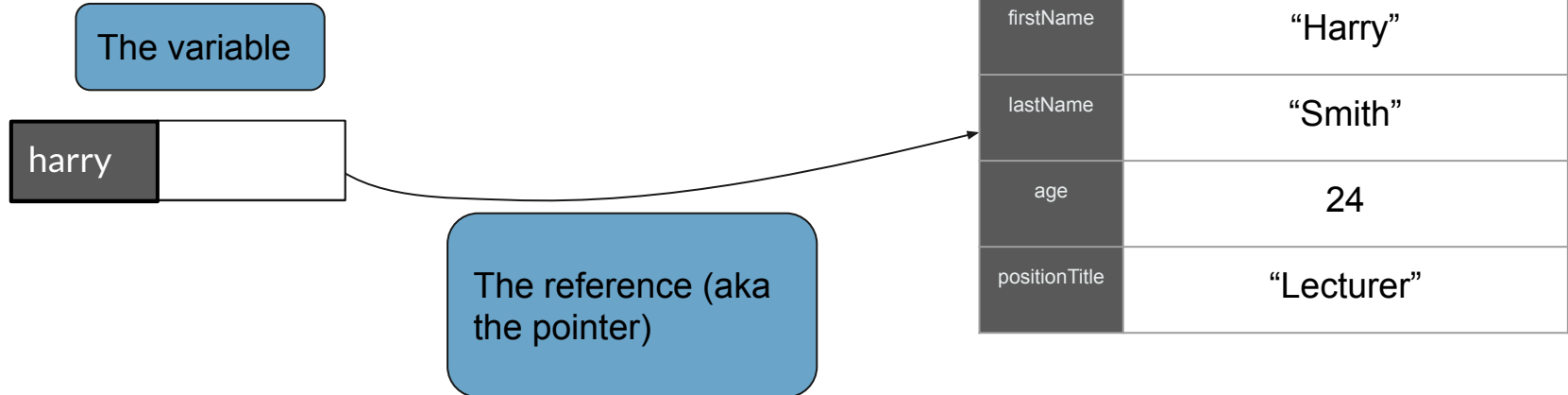
This is!

firstName	"Harry"
lastName	"Smith"
age	24
positionTitle	"Lecturer"

What's a reference?

```
Instructor harry = new Instructor("Harry", "Smith", 24, "Lecturer");
```

The pointee (aka the data, stored somewhere in the computer's memory)





Why does the pointee look like this?

```
1 public class Instructor {  
2     private String firstName;  
3     private String lastName;  
4     private int age;  
5     private String positionTitle;  
6  
7     ...  
8 }
```

The pointee is a “box” containing the data that any object of its Class must have based on the fields (name, age, etc.)

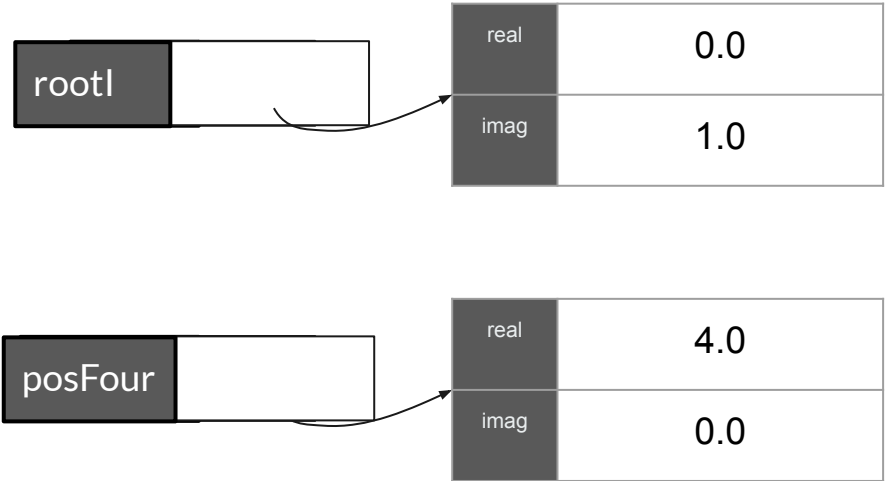
The pointee (aka the data, stored somewhere in the computer’s memory)

firstName	“Harry”
lastName	“Smith”
age	24
positionTitle	“Lecturer”

Worked Example of Variable Assignment

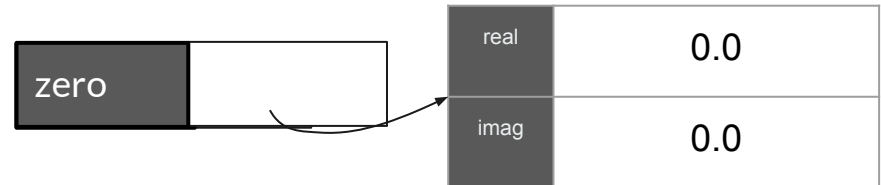
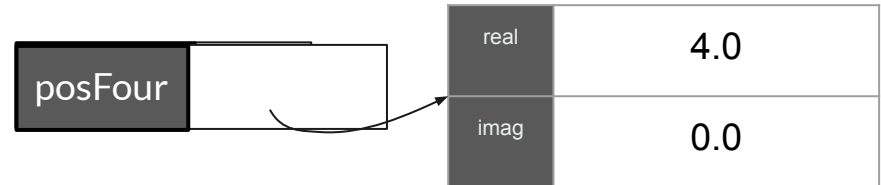
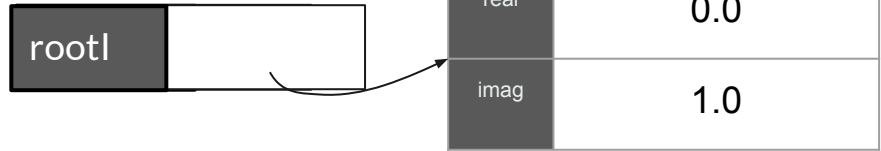
```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);
```

```
1 public class Complex {  
2     private double real;  
3     private double imag;  
4  
5     public Complex(double real, double imag) {  
6         this.real = real;  
7         this.imag = imag;  
8     }  
9  
10    public void copyValues(Complex other) {  
11        this.real = other.real;  
12        this.imag = other.imag;  
13    }  
14    ...  
15 }
```



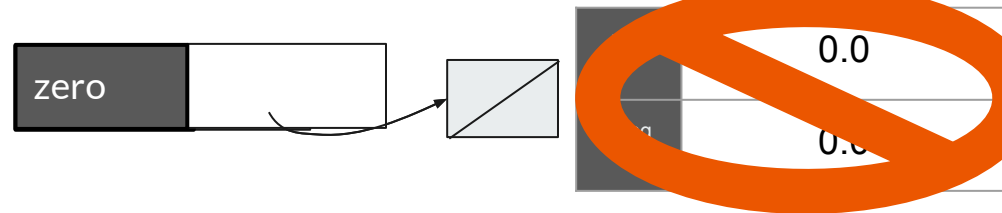
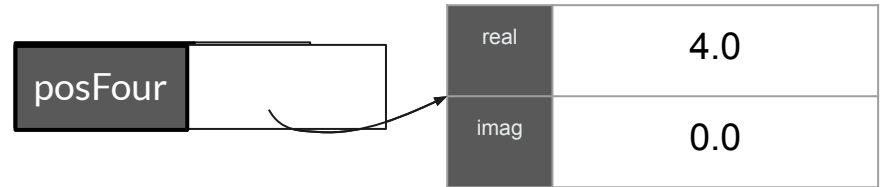
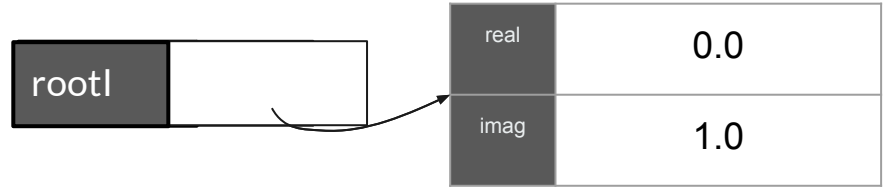
Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);  
Complex zero = new Complex(0.0, 0.0);
```



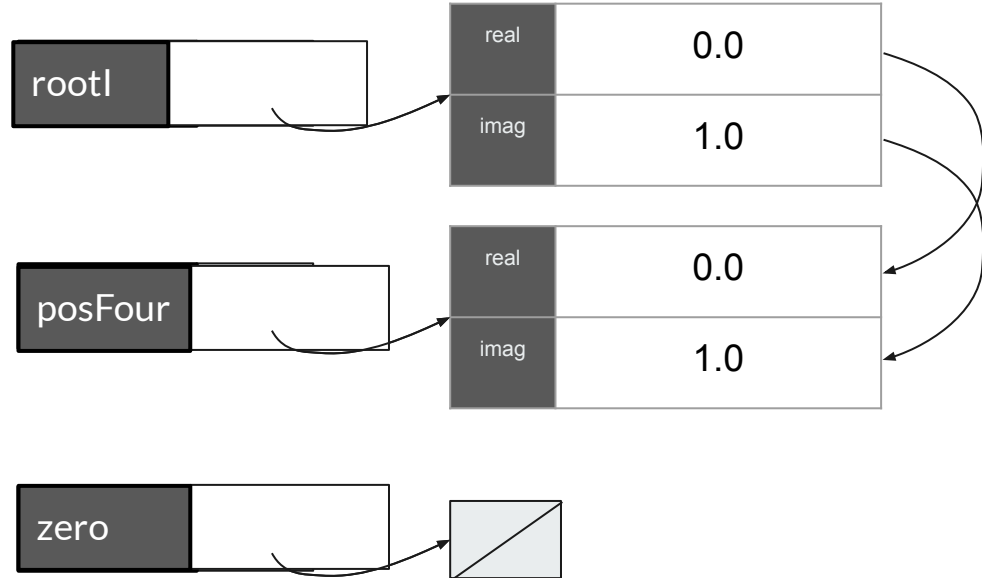
Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);  
  
Complex zero = new Complex(0.0, 0.0);  
zero = null;
```



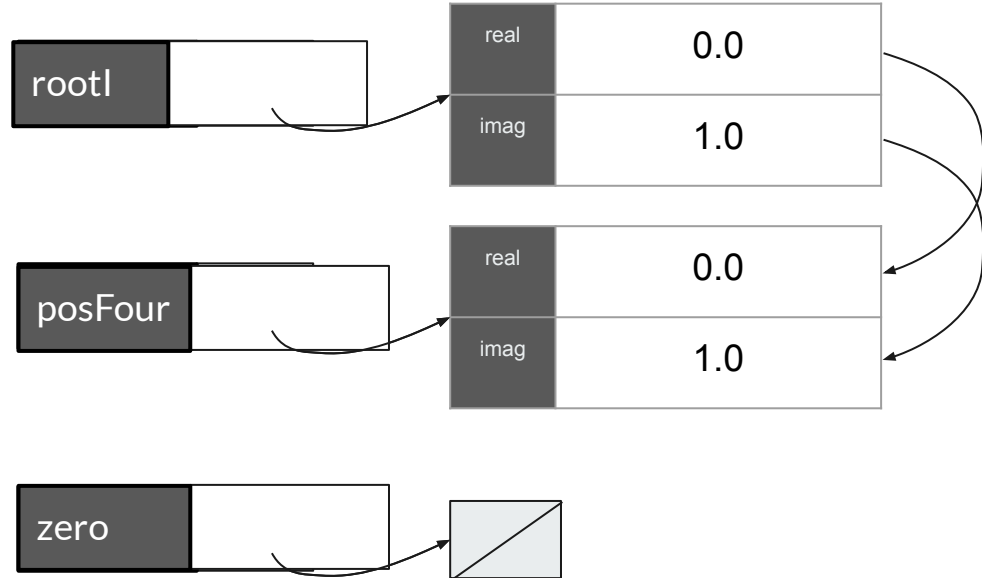
Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);  
  
Complex zero = new Complex(0.0, 0.0);  
zero = null;  
  
posFour.copyValues(rootI);
```



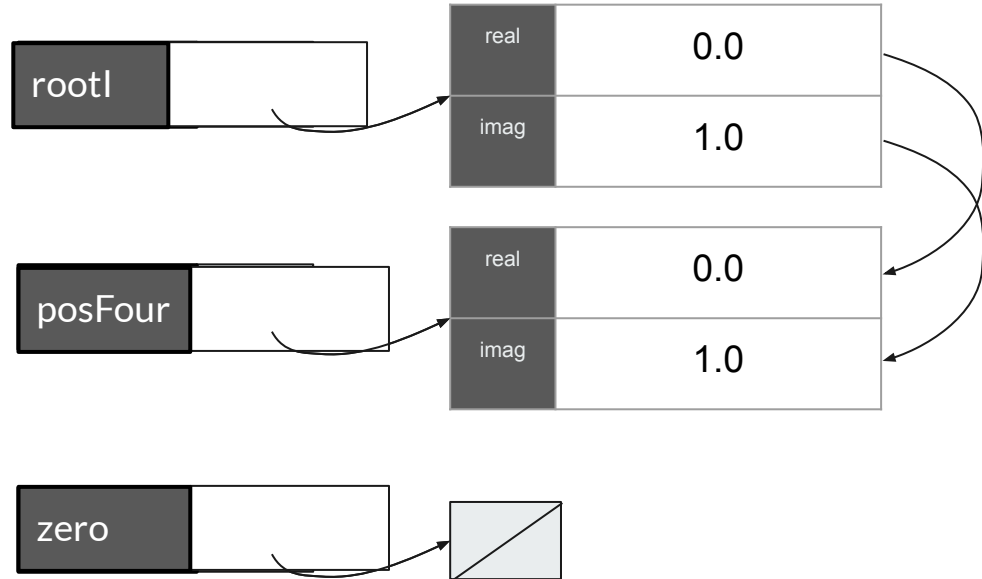
Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);  
  
Complex zero = new Complex(0.0, 0.0);  
zero = null;  
  
posFour.copyValues(rootI);
```



Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);  
  
Complex zero = new Complex(0.0, 0.0);  
zero = null;  
  
posFour.copyValues(rootI);  
  
posFour == rootI; //How does this evaluate?
```



When poll is active, respond at pollev.com/cis110sp21

Text **CIS110SP21** to **22333** once to join

How does `posFour == root1` evaluate here?

true

false



Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);  
Complex posFour = new Complex(4.0, 0.0);
```

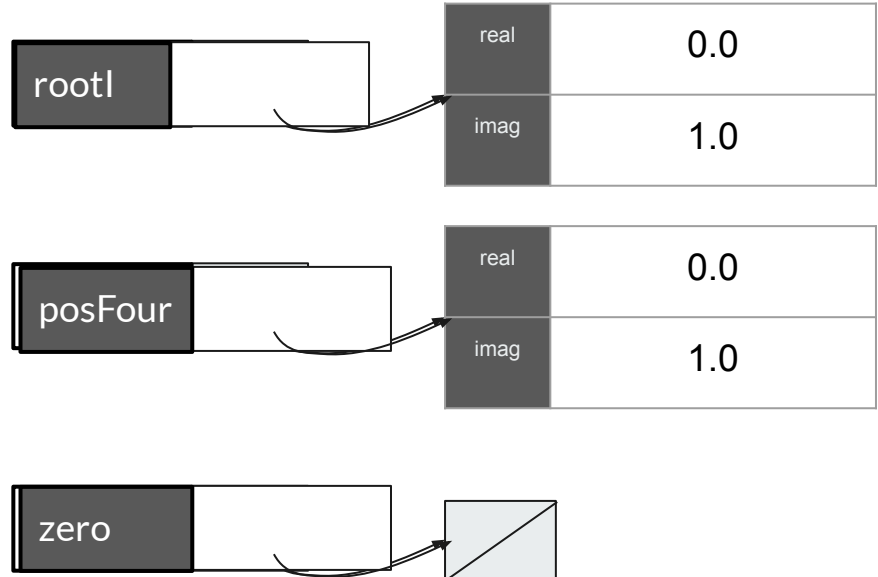
```
Complex zero = new Complex(0.0, 0.0);  
zero = null;
```

```
posFour.copyValues(rootI);
```

```
posFour == rootI; // false
```

```
posFour.equals(rootI); // true or false?
```

```
public boolean equals(Complex other) {  
    return this.real == other.real &&  
           this.imag == other.imag;  
}
```



When poll is active, respond at pollev.com/cis110sp21

Text **CIS110SP21** to **22333** once to join

How does `posFour.equals(rootI)` evaluate?

true

false



Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);
Complex posFour = new Complex(4.0, 0.0);

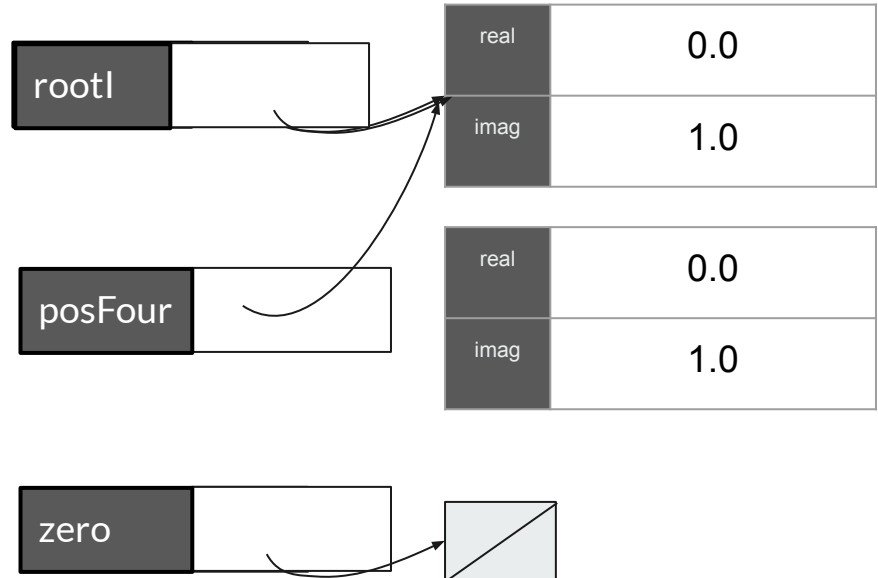
Complex zero = new Complex(0.0, 0.0);
zero = null;

posFour.copyValues(rootI);

posFour == rootI; // false
posFour.equals(rootI); // true

posFour = rootI;

// how do these evaluate?
posFour == rootI;
posFour.equals(rootI);
```



When poll is active, respond at pollev.com/cis110sp21

Text **CIS110SP21** to **22333** once to join

**How do these evaluate: `posFour == rootI;`
`posFour.equals(rootI);`**

false, false

false, true

true, false

true, true



Worked Example of Variable Assignment

```
Complex rootI = new Complex(0.0, 1.0);
Complex posFour = new Complex(4.0, 0.0);

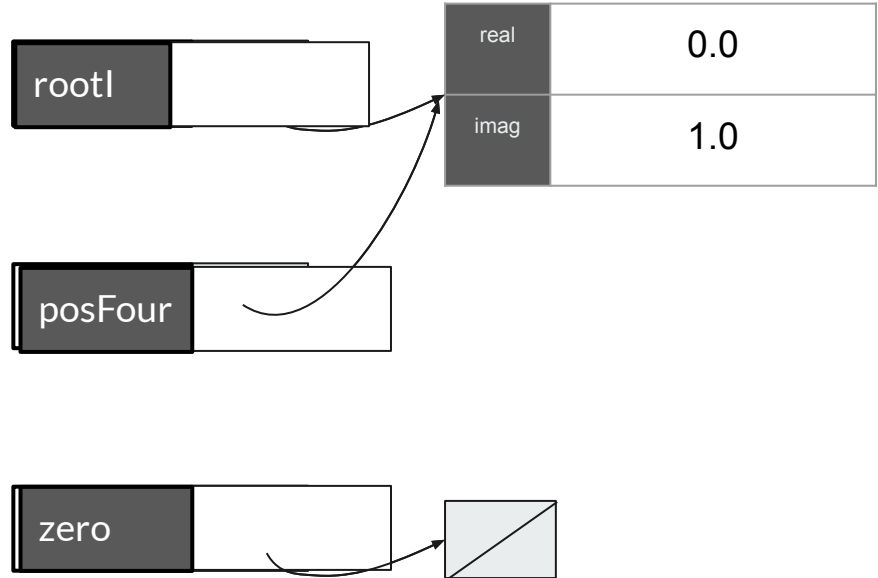
Complex zero = new Complex(0.0, 0.0);
zero = null;

posFour.copyValues(rootI);

posFour == rootI; // false
posFour.equals(rootI); // true

posFour = rootI;

// how do these evaluate?
posFour == rootI; // true
posFour.equals(rootI); // true
```





Dereferencing

Dereferencing uses the . operator

```
Instructor harry = new Instructor("Harry", "Smith", 24, "Lecturer");  
System.out.println(harry.lastName + ", " + harry.firstName)
```

For harry.lastName...

harry is the variable

harry

harry. tells Java to follow the reference stored inside of the variable

firstName	"Harry"
lastName	"Smith"
age	24
positionTitle	"Lecturer"

Dereferencing uses the . operator

```
Instructor harry = new Instructor("Harry", "Smith", 24, "Lecturer");  
System.out.println(harry.lastName + ", " + harry.firstName)
```

For harry.lastName...

harry.lastName is the value of the field "lastName" of the object that we've arrived at following the reference.

harry is the variable

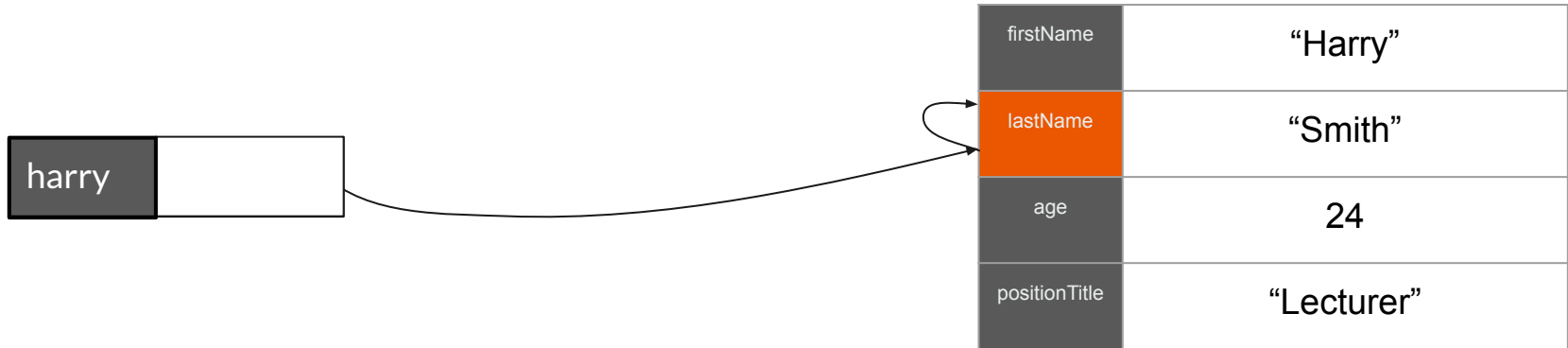
harry

harry. tells Java to follow the reference stored inside of the variable

firstName	"Harry"
lastName	"Smith"
age	24
positionTitle	"Lecturer"

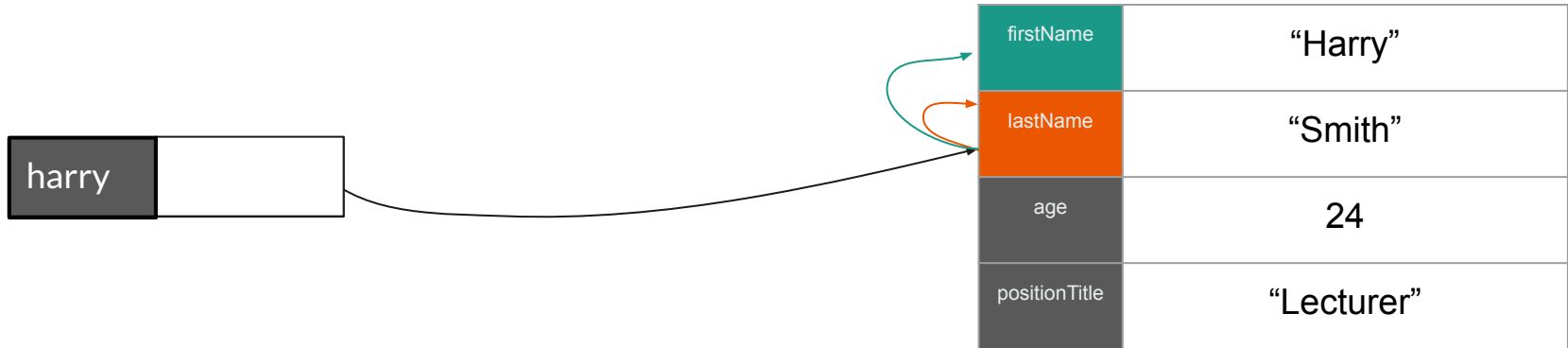
Dereferencing uses the . operator

```
Instructor harry = new Instructor("Harry", "Smith", 24, "Lecturer");  
System.out.println(harry.lastName + ", " + harry.firstName)
```



Dereferencing uses the . operator

```
Instructor harry = new Instructor("Harry", "Smith", 24, "Lecturer");  
System.out.println(harry.lastName + ", " + harry.firstName)
```



harry

Dereferencing can be repeated

```
1 public class Instructor {  
2     private String firstName;  
3     private String lastName;  
4     private Instructor chair;  
5     private String positionTitle;  
6  
7     ...  
8 }
```

Replace the age field with the Instructor's department chair...

firstName	"Harry"
lastName	"Smith"
chair	
positionTitle	"Lecturer"

firstName	"Zachary"
lastName	"Ives"
chair	
positionTitle	"Professor"

Dereferencing can be repeated

```
harry.chair.lastName
```

harry is the variable

harry

harry. follows the reference

firstName	"Harry"
lastName	"Smith"
chair	
positionTitle	"Lecturer"

firstName	"Zachary"
lastName	"Ives"
chair	
positionTitle	"Professor"

Dereferencing can be repeated

harry.chair.lastName

harry is the variable



harry. follows the reference

firstName	"Harry"
lastName	"Smith"
chair	harry.chair stores a reference
positionTitle	"Lecturer"

harry.chair. follows this ref.

A diagram showing a reference to a chair object. An arrow points from the 'chair' field of the first table to the 'chair' field of this second table.

firstName	"Zachary"
lastName	"Ives"
chair	
positionTitle	"Professor"

Dereferencing can be repeated

harry.chair.lastName

harry is the variable



harry. follows the reference

firstName	"Harry"
lastName	"Smith"
chair	harry.chair stores a reference
positionTitle	"Lecturer"

harry.chair. follows this ref.

A diagram showing a table with a 'chair' row. An arrow points from the 'chair' cell to a second table, which represents the object referenced by the 'chair' property of the first table.

firstName	"Zachary"
lastName	"Ives"
chair	
positionTitle	"Professor"

What is the value of `harry.chair.chair`? (This is tricky!)



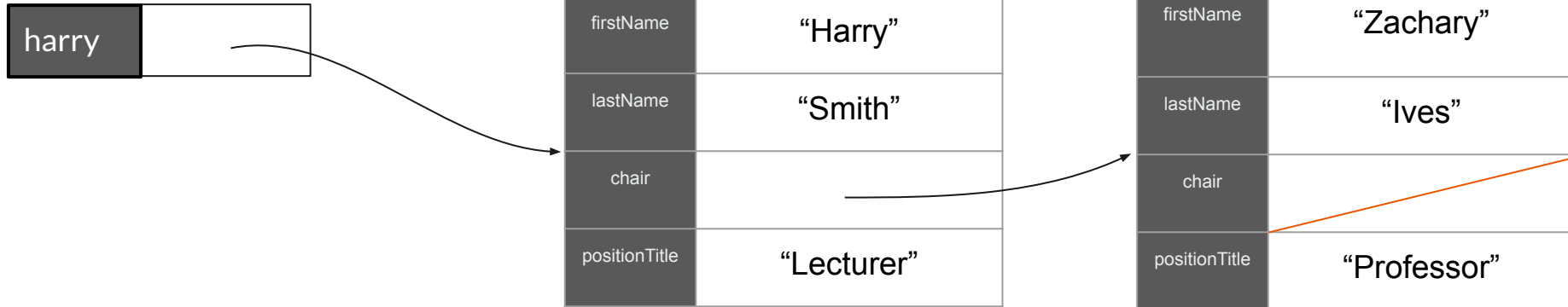
What is the value of `harry.chair.chair.firstName`? (Even trickier.)



Dereferencing a null pointer gives a `NullPointerException`

```
harry.chair.chair.lastName
```

`harry.chair.chair` is null, so `harry.chair.chair.lastName` asks Java to follow a reference that doesn't exist!



And finally... *STATIC!*

Static fields and methods belong to the entire Class. All objects of that Class share its same Static values.

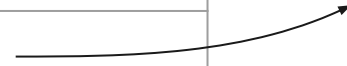
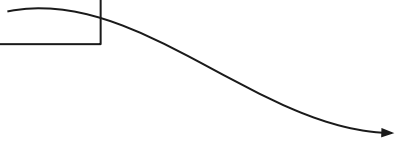
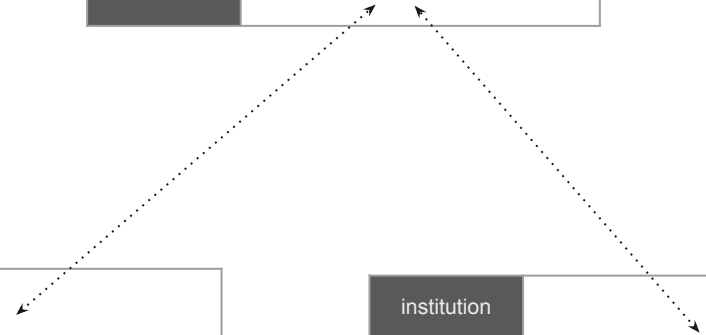
```
1 public class Instructor {
2   private static String institution = "Penn";
3   private String firstName;
4   private String lastName;
5   private Instructor chair;
6   private String positionTitle;
7
8   ...
9 }
```

Instructor	
institution	"Penn"

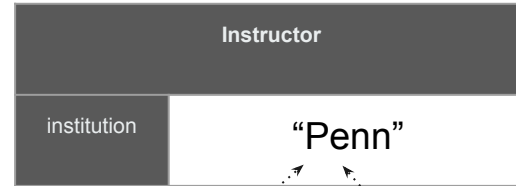
harry	
-------	--

institution	
firstName	"Harry"
lastName	"Smith"
chair	
positionTitle	"Lecturer"

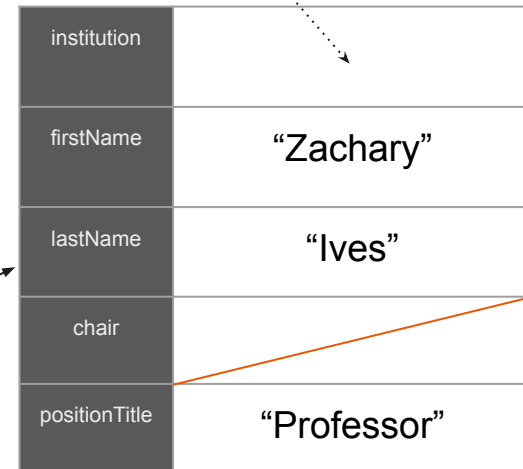
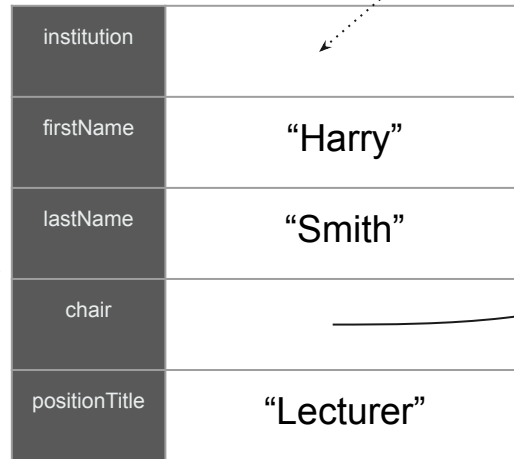
institution	
firstName	"Zachary"
lastName	"Ives"
chair	
positionTitle	"Professor"




```
1 public class Instructor {
2   private static String institution = "Penn";
3   private String firstName;
4   private String lastName;
5   private Instructor chair;
6   private String positionTitle;
7
8   ...
9 }
```



These dashed arrows are not technically references, but this mental model should be useful.



**Only static fields can be
referenced inside of static
methods.**

