

Notes

TA office hours – the TA's are great! Please check with them if you need help.

- Are the times of their office hours not good?

A reminder to follow the specifications of the labs exactly

- I will give partial credit, but please test and review carefully vs the spec
- A hint: print out labs, cross out each sentence as you implement, highlight as you test

Last time...

We discussed Java classes and objects

We experimented with class `String`

We saw how to access command-line arguments to Java programs

CS112 –
Java
Programming

Spring 2024

Copyright 2023 Paul Haskell. All rights reserved.

Java Objects and References

REVIEW: What is an Object?

An INSTANTIATION of a class

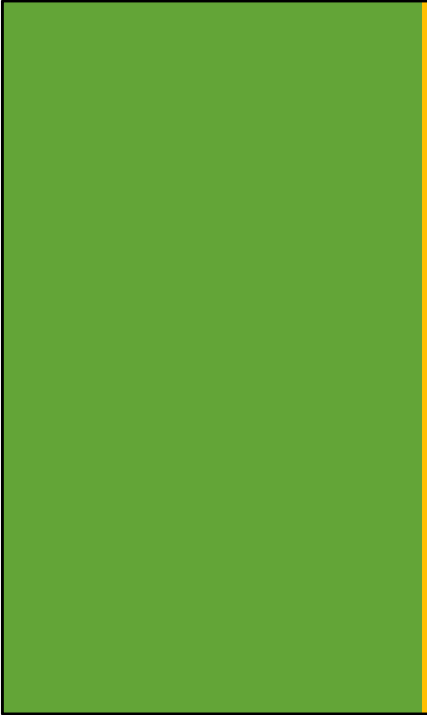
A VARIABLE whose type is some class

A THING that contains class variables and methods

We (or others) can use it to access the variables (if public) and to call the methods

It has STATE (determined by variables) and FUNCTIONALITY (thru the methods)

LOOK AT Border.java program



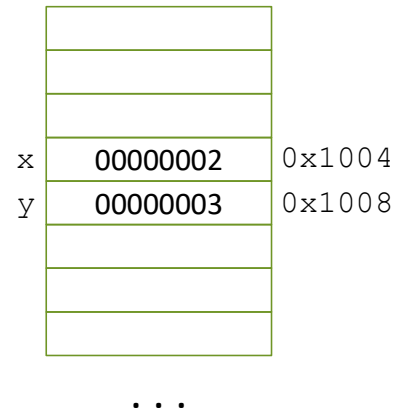
Primitive Types vs Object Types

Primitive Types

Remember all 8?

Each variable of a "primitive type" has its own value, stored in a dedicated memory location.

```
int x = 2  
int y = x;  
y += 1;  
System.out.println(x); // prints 2
```



Primitive Types

When passed as method arguments, primitive-type variables are copied by value

```
class Doubler { ...
```

See Doubler.java program

Object Variables

References and Objects

- An object is not created when you declare a variable

```
Doubler myDoubler;
```

- An object is created only when you call the new operator

```
Doubler myDoubler = new Doubler();
```

Your declaration only creates a reference i.e a name that refers to some object.

Object Variables

Ok then, what object is referred to by

```
Doubler myDoubler;
```

???

The reference is to a special Java value called `null`.

- You can check if a reference is `null`. `if (myDoubler == null) { ...`
- You can assign `null` to a reference.
- But of course you cannot call any class methods on `null`, cannot assign values to `null`, etc.

Now for the tricky part...

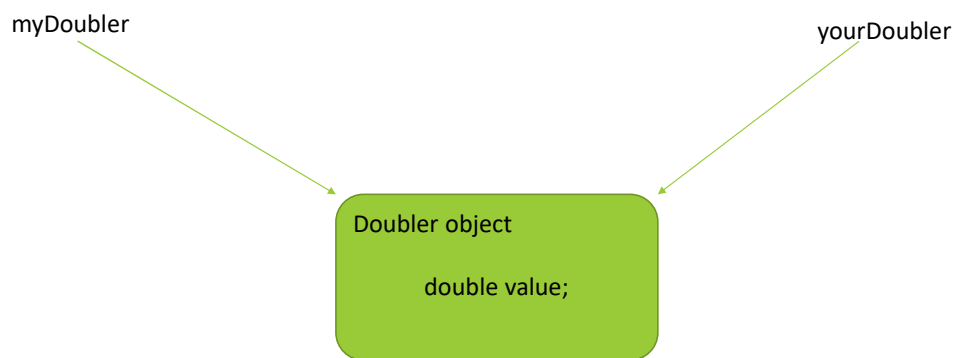
Multiple references (multiple names) can refer to the same object

```
Doubler myDoubler = new Doubler();  
myDoubler.set(1.0);  
Doubler yourDoubler = myDoubler;  
yourDoubler.set(10.0);  
System.out.println(myDoubler.get());  
???
```

Prints 20. Do in Eclipse—do not draw picture yet.

REMEMBER: l-value and r-value discussion! Variable name on the left = reference.
On the right = object referenced.

Now for the tricky part...



THIS DOES NOT HAPPEN WITH PRIMITIVE DATA TYPE VARIABLES: THEY JUST HOLD A VALUE.

A related tricky part

Java never passes an object as an argument to a method.

Java always passes a reference as a parameter to a method.

```
void ProcessNumbers(Doubler ddd) {  
    ddd.set(20.0);  
}  
  
Doubler myDoubler = new Doubler();  
myDoubler.set(1.0);  
ProcessNumbers(myDoubler);  
System.out.println(myDoubler.get()); // prints ???
```

Prints 40.0. DRAW IT OUT!

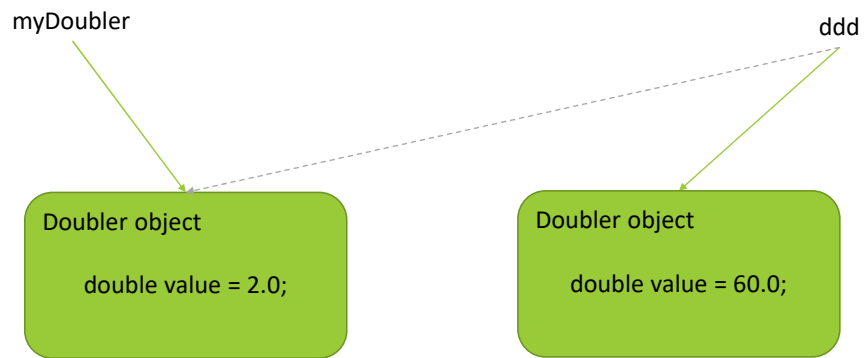
Now a SUPER TRICKY part

As method arguments, Java references are passed by value.

```
void ProcessNumbers(Doubler ddd) {  
    ddd = new Doubler();  
    ddd.set(30.0);  
}  
  
Doubler myDoubler = new Doubler();  
myDoubler.set(1.0);  
ProcessNumbers(myDoubler);  
System.out.println(myDoubler.get()); // prints ???
```

Prints 2.0. DRAW IT OUT!
THIS IS TRICKY!

Now a SUPER TRICKY part



Python is very similar!

```
#!/usr/bin/python  
  
def changeList( myList ):  
    myList.append(99)  
    myList = [1,2,3,4]  
  
theList = [10,20,30]  
changeList(theList)  
print(theList)
```

Prints 10,20,30,99

A related issue – comparing References

```
Doubler myDoubler = new Doubler();  
myDoubler.set(3);  
  
Doubler yourDoubler = new Doubler();  
yourDoubler.set(3);  
  
System.out.println(myDoubler == yourDoubler);  
// true or false?
```

false

A related issue – comparing References

The "==" operator and other logical operators compare if two references point to the same object

There is another method called "equals()". You can program it to do whatever you want.

```
class Doubler {  
    boolean equals(Doubler other) {  
        return (value == other.value);  
    }  
}
```

Can put any code we want into equals(), but very confusing if we don't do the right thing...

So...

```
Doubler myDoubler = new Doubler();  
myDoubler.set(3);
```

```
Doubler yourDoubler = new Doubler();  
yourDoubler.set(3);
```

```
System.out.println(myDoubler == yourDoubler);  
System.out.println(myDoubler.equals(yourDoubler));
```

False, True

Random
Numbers



Why would we want a computer to make random numbers?

Simulate or study processing of random inputs



Can computers make random numbers?

Not exactly, but...

PSEUDOrandom numbers

WORK VERY WELL—in Java. Not so much in the old days, esp Windows.

(Pseudo) Random numbers have a Distribution

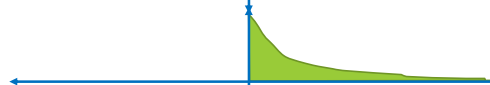
Uniform



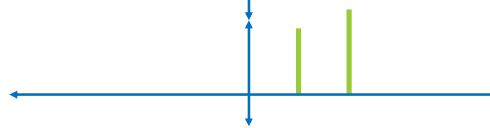
Gaussian



Exponential



Binomial



Java Random library

```
import Java.util.Random;  
Random myGenerator = new Random();  
myGenerator.nextDouble(); // pseudorandom number in [0.0, 1.0)  
myGenerator.nextGaussian(); // "normal" Gaussian values
```

If we want Exponential, Binomial, Poisson, etc we must create them ourselves.

nextInt(), nextLong(), nextBoolean(), etc.
nextInt(UpperLimit) – useful!

Richer than simple Math.random()