Lecture 7
CSE 11 Fall 2013

Method Signatures

- (not covered in the book)
- The name of method + order and types of arguments == Signature
- Sun (Location iLoc, double dia; DrawingCanvas canvas)
  - Sun(Location, double, DrawingCanvas)
- Sun (double x, double y, double dia; DrawingCanvas canvas)
  - Sun(double, double, double, DrawingCanvas)
- Java matches the signature of the method call with the signature of the method definition to find the correct method to call.

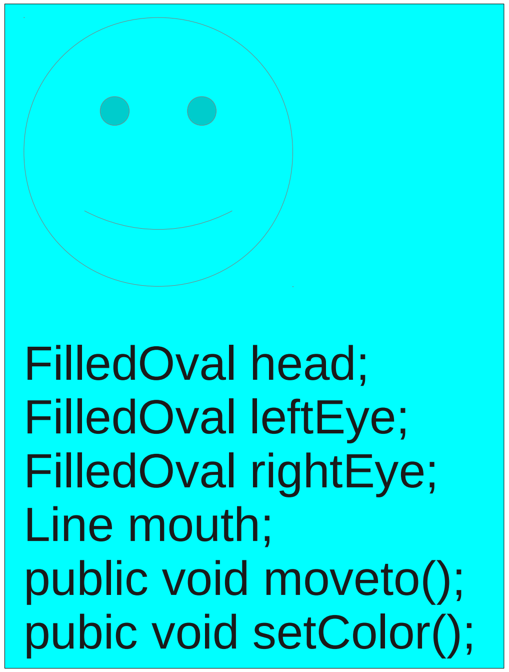
Methods, Logically

- Methods with the same name should perform roughly the same function (logically)
- If the same method name (but with a different signature) does something completely different
  - Confusion about the logic of the program
  - Difficult debugging/troubleshooting
- Don't try to be “clever”, if you need a different method name, declare it.

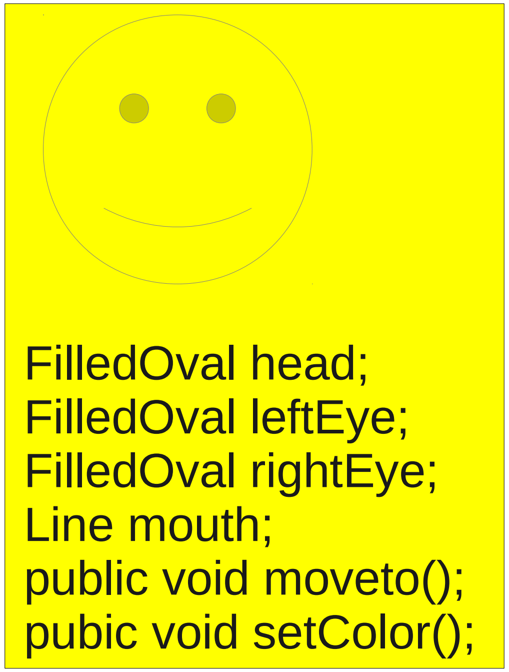
Methods (Logically)

- Methods with the same name should perform roughly the same function (logically)
- If the same method name (but with a different signature) does something completely different
  - Confusion about the logic of the program
  - Difficult debugging/troubleshooting
- Don't try to be “clever”, if you need a different method, declare it.
- Method names should reflect what they do

Instances, Revisited



```
FilledOval head;
FilledOval leftEye;
FilledOval rightEye;
Line mouth;
public void moveto();
pubic void setColor();
```



```
FilledOval head;
FilledOval leftEye;
FilledOval rightEye;
Line mouth;
public void moveto();
pubic void setColor();
```

- Think of instances as not only having state variables (instance variables), but also instance methods
- When a method is invoked (running) it knows which instance it is part of. The keyword **`this`** is how you reference the particular instance inside of a method

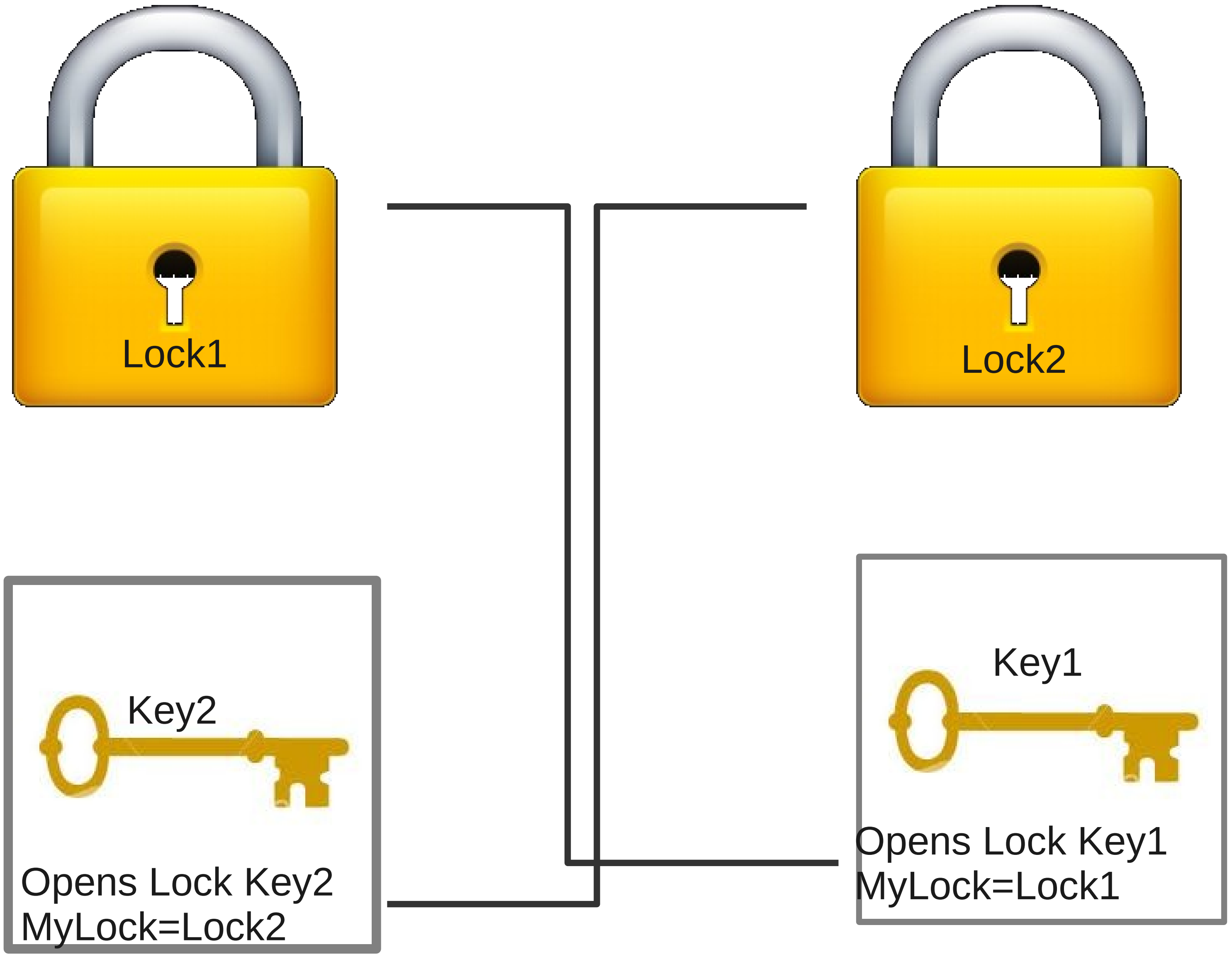
this

- refers to the “this instance” of an object

```
public class Key {
    private Lock myLock;
    public Key (Lock theLock) {  // Constructor, typo in book
          myLock = theLock;      // Note: new is NOT used here
    }
    public Lock getMyLock() {
        return myLock;
    }
}
public class Lock {
    public Key createKey() {
         return new Key(this);
    }
}
```

Whichever Lock instance is being asked to run the createKey() method

Lock and Key Objects



How example in book works

- Quick sample

```
Lock bikeLock, bikeLock2;
Key bikeKey, bikeKey2;

bikeLock = new Lock();     // need a lock instance
bikeKey = bikeLock.createKey();  // key to this lock

BikeLock2 = new Lock();
BikeKey2 bikeLock2.createKey();

if (bikeKey.getMyLock() == bikeLock )
     System.out.println(“These are the same object”);

if (bikeKey2.getMyLock() == bikeLock )
     System.out.println(“This should not print!”);
```

Putting it All together (From PR#2)

Class Name
Instance Variables
Methods

WeightBox
Line rope FilledRect box;
void setColor(Color) void hoist(double) double getRopeLength()

Overloaded Methods

- Same method name, Different method signature

- Remember, Java distinguishes method signatures on <u>types in the argument list</u>
  - What you name your arguments is irrelevant!
- Following have identical signatures
  - `Public Line (double x, double y, double endX, double endY)`
  - `Public Line (double x, double y, double length, double angle)`
- If you declare both, it is an error! (they have the identical signature)
-

Overloaded Methods

- Are VERY common
  - int Math.abs(int), double Math.abs(double), ...
- Are VERY useful
  - Want absolute value? Math.abs (“easy to remember and understand)
- Especially Constructors!