Lecture 5: Access Control, Class Scope, Packages, Java API

Overview

- Review
- Access control
- Class scope
- Packages
- Java API

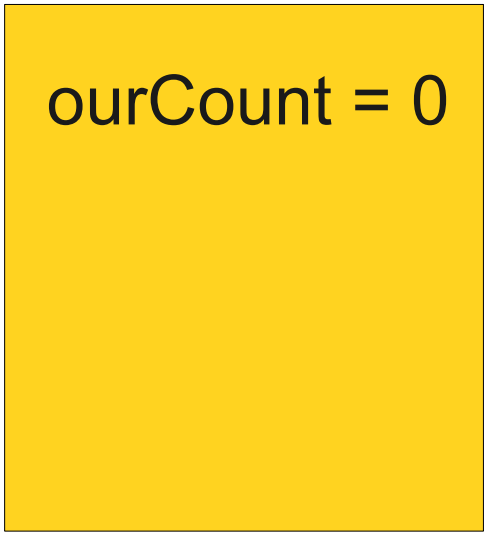
```
public class Counter {
   int myCount = 0;
   static int ourCount = 0;
   void increment() {
      myCount++;
      ourCount++;
   }
   public static void main(String[] args) {
      Counter counter1 = new Counter();
      Counter counter2 = new Counter();
      counter1.increment();
      counter1.increment();
      counter2.increment();
      System.out.println("Counter 1: " +
counter1.myCount + " " + counter1.ourCount);
      System.out.println("Counter 2: " +
counter2.myCount + " " + counter2.ourCount);
   }
}
```

```
public class Counter {
   int myCount = 0;
   static int ourCount = 0;
   void increment() {
      myCount++;
      ourCount++;
   }
   public static void main(String[] args) {
      Counter counter1 = new Counter();
      Counter counter2 = new Counter();
      counter1.increment();
      counter1.increment();
      counter2.increment();
      System.out.println("Counter 1: " +
counter1.myCount + " " + counter1.ourCount);
      System.out.println("Counter 2: " +
counter2.myCount + " " + counter2.ourCount);
   }
}
```

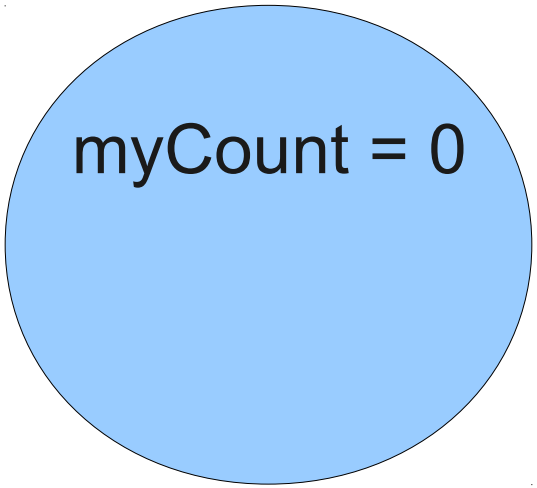
Fields

Method

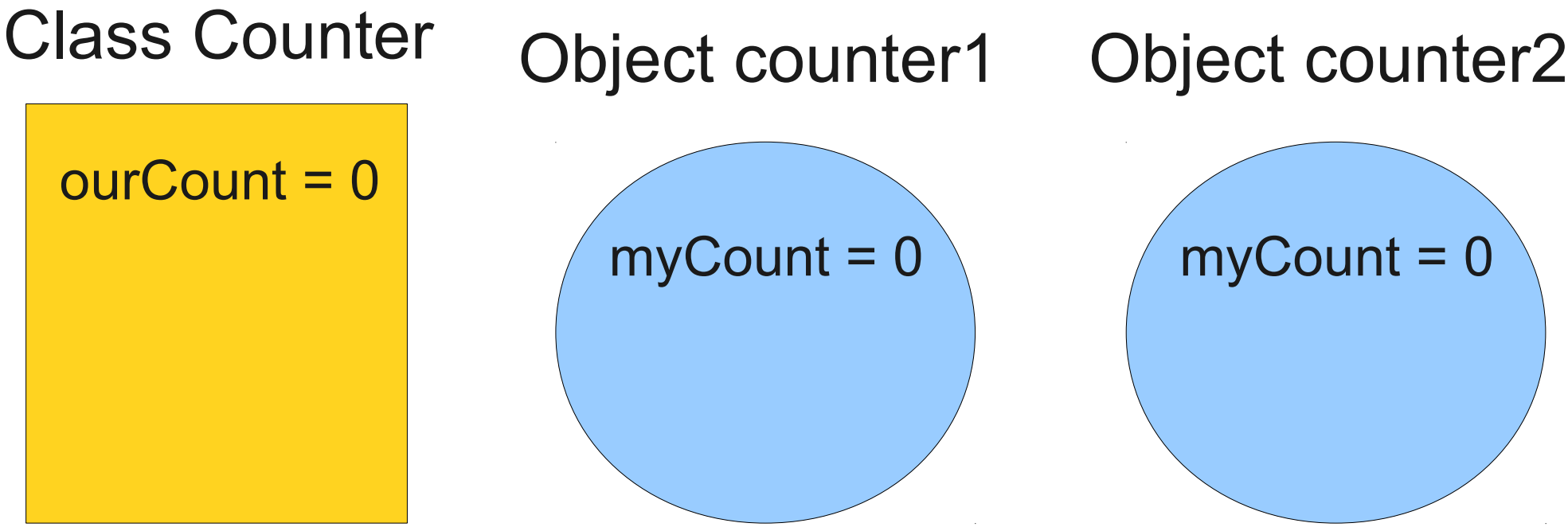
Class Counter



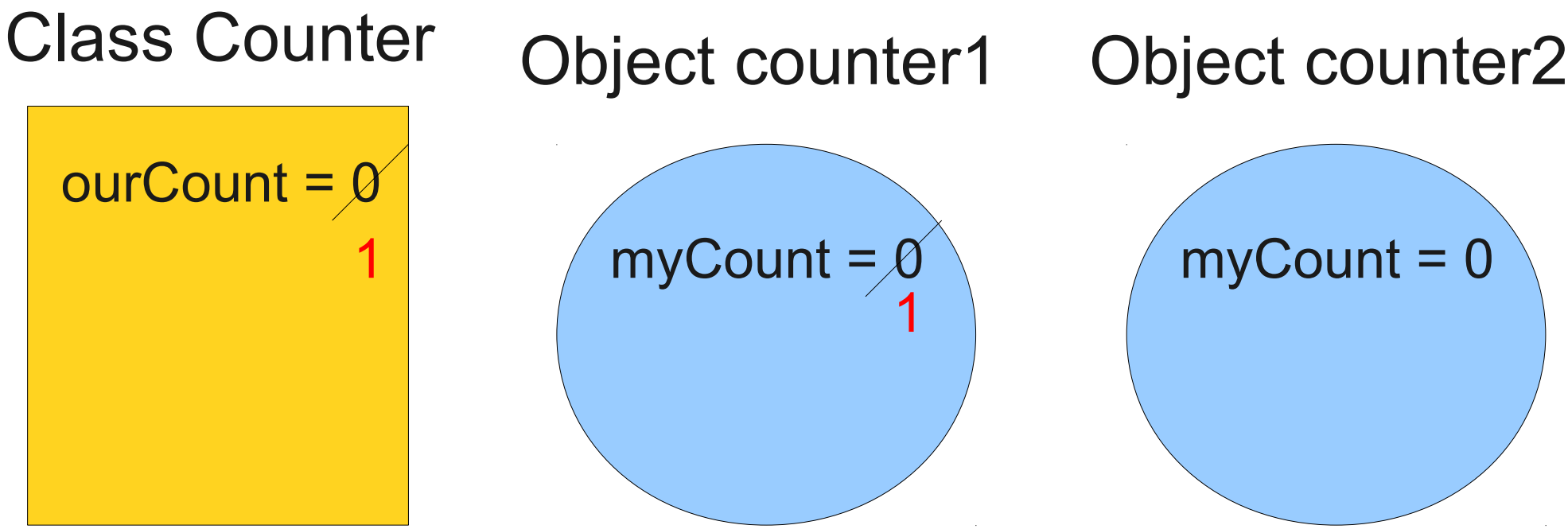
Object counter1



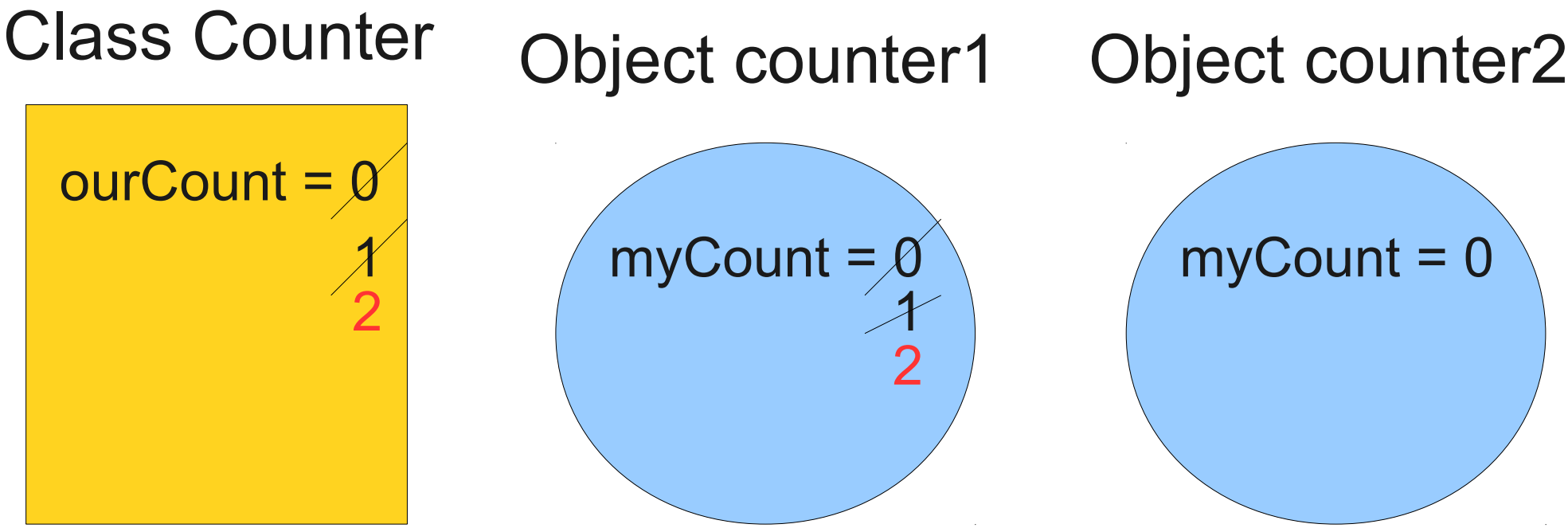

```
Counter counter1 = new Counter();
```



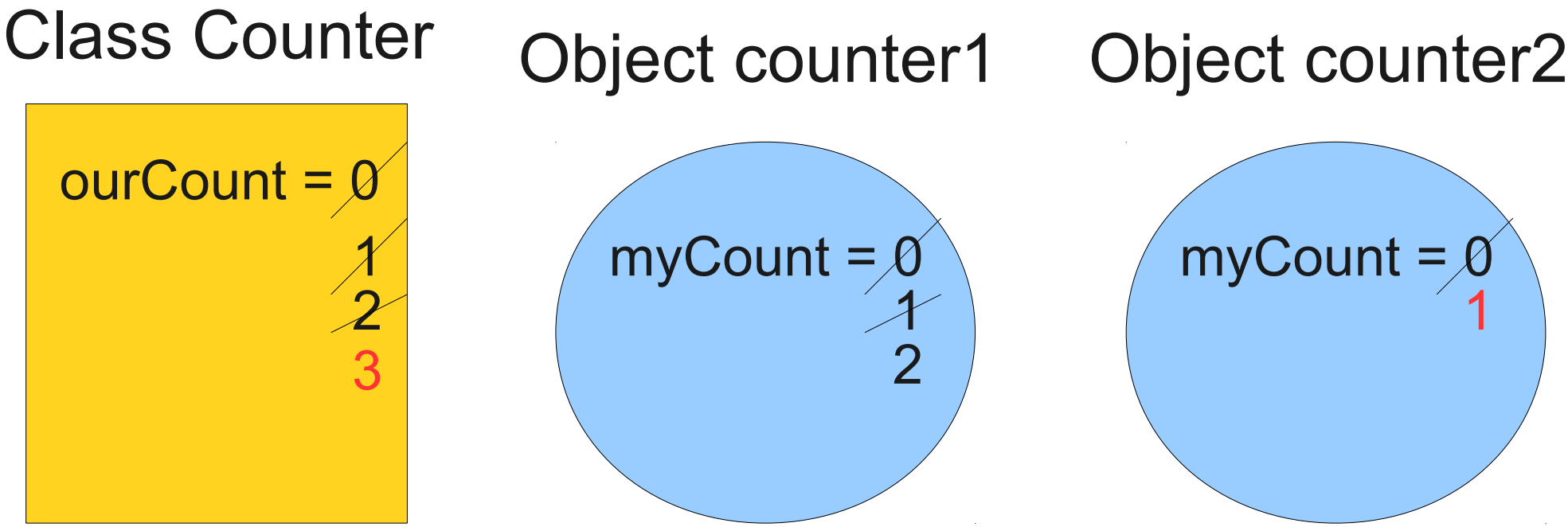

```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
```




```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
```




```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
counter1.increment();
```




```
Counter counter1 = new Counter();
Counter counter2 = new Counter();
counter1.increment();
counter1.increment();
counter2.increment();
```

Overview

- Review
- Access control
- Class scope
- Packages
- Java API

Access Control

```
public class CreditCard {
   String cardNumber;
   double expenses;
   void charge(double amount) {
      expenses = expenses + amount;
   }
   String getCardNumber(String password) {
      if (password.equals("SECRET!3*!")) {
         return cardNumber;
      }
         return "jerkface";
   }
}
```

Mr. MeanGuy

```
public class Malicious {
   public static void main(String[] args) {
      maliciousMethod(new CreditCard());
   }
   static void maliciousMethod(CreditCard card)
   {
      card.expenses = 0;
      System.out.println(card.cardNumber);
   }
}
```

Public vs. Private

- Public: others can use this
- Private: only the class can use this

public/private applies to any
field or **method**

Access Control

```
public class CreditCard {
   String cardNumber;
   double expenses;
   void charge(double amount) {
      expenses = expenses + amount;
   }
   String getCardNumber(String password) {
      if (password.equals("SECRET!3*!")) {
         return cardNumber;
      }
         return "jerkface";
   }
}
```

Access Control DONE RIGHT

```
public class CreditCard {
   private String cardNumber;
   private double expenses;
   public void charge(double amount) {
      expenses = expenses + amount;
   }
   public String getCardNumber(String password)
   {
      if (password.equals("SECRET!3*!")) {
         return cardNumber;
      }
      return "jerkface";
   }
}
```

Why Access Control

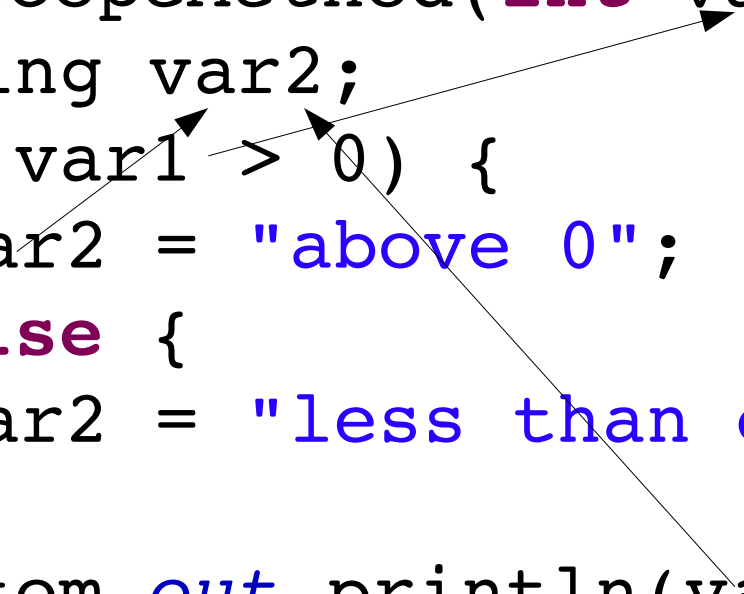
- Protect private information (sorta)
- Clarify how others should use your class
- Keep implementation separate from interface

Overview

- Review
- Access control
- Class scope
- Packages
- Java API

Scope Review

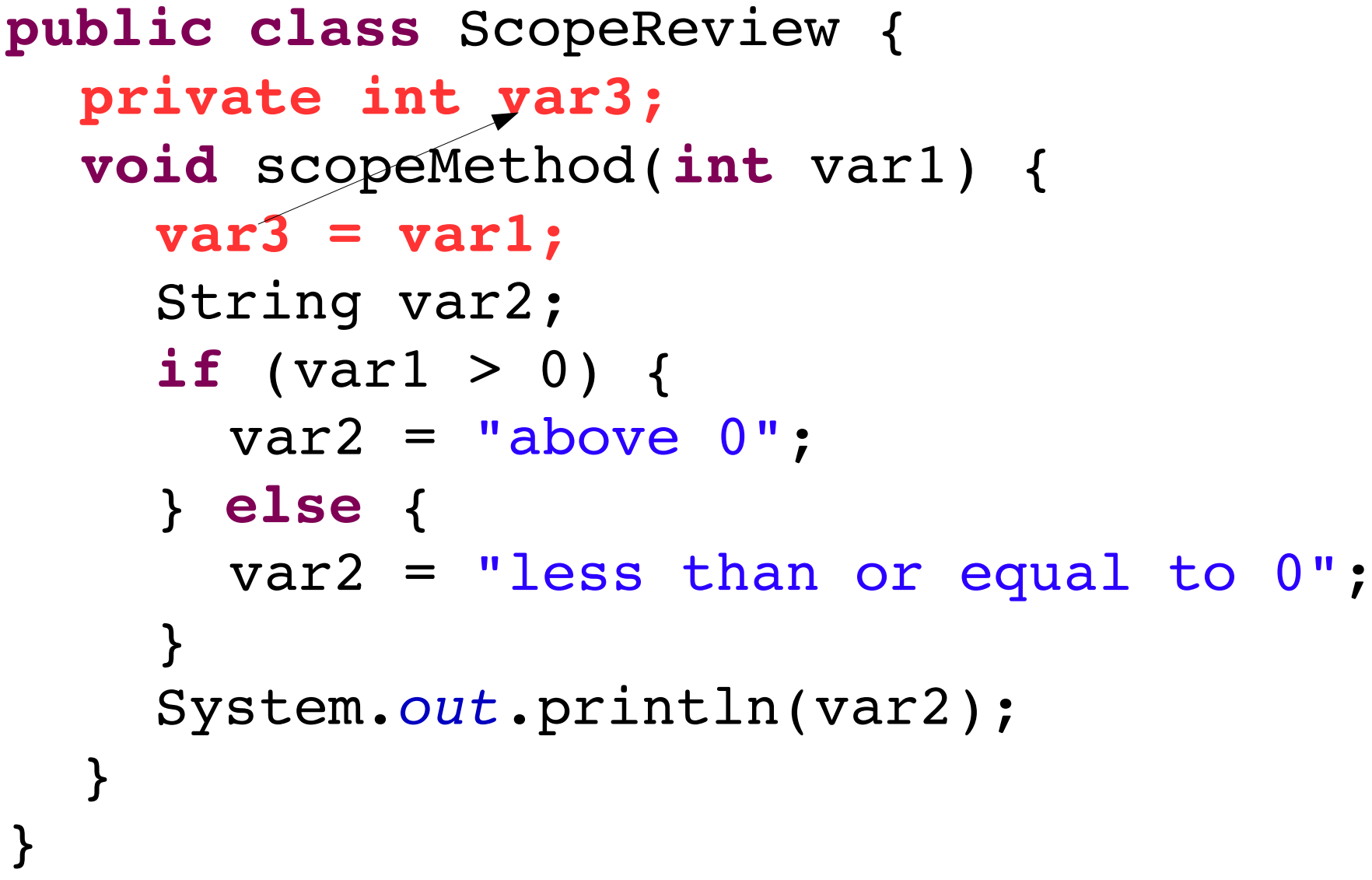
```
public class ScopeReview {
   void scopeMethod(int var1) {
      String var2;
      if (var1 > 0) {
         var2 = "above 0";
      } else {
         var2 = "less than or equal to 0";
      }
      System.out.println(var2);
   }
}
```



Scope Review

```
public class ScopeReview {
   private int var3;
   void scopeMethod(int var1) {
      var3 = var1;
      String var2;
      if (var1 > 0) {
         var2 = "above 0";
      } else {
         var2 = "less than or equal to 0";
      }
      System.out.println(var2);
   }
}
```

Class Scope



```
public class ScopeReview {
   private int var3;
   void scopeMethod(int var1) {
      var3 = var1;
      String var2;
      if (var1 > 0) {
         var2 = "above 0";
      } else {
         var2 = "less than or equal to 0";
      }
      System.out.println(var2);
   }
}
```

Scope

Just like methods, variables are accessible inside {}

- Previous lessons: method-level scope

```
void method(int arg1) {
    int arg2 = arg1 + 1;
}
```

- This lesson: class-level scope

```
class Example {
    int memberVariable;
    void setVariable(int newVal) {
        memberVariable += newVal;
    }
}
```

Only method-level 'servings' is updated

```
public class Baby {
  int servings;
  void feed(int servings) {
    servings = servings + servings;
  }
  void poop() {
    System.out.println("All better!");
    servings = 0;
  }
}
```

'this' keyword

- Clarifies scope
- Means 'my object'

Usage:

```
class Example {
        int memberVariable;
        void setVariable(int newVal) {
            this.memberVariable += newVal;
        }
    }
```

Only method-level 'servings' is updated

```
public class Baby {
  int servings;
  void feed(int servings) {
    servings = servings + servings;
  }
  void poop() {
    System.out.println("All better!");
    servings = 0;
  }
}
```

Object-level 'servings' is updated

```
public class Baby {
  int servings;
  void feed(int servings) {
    this.servings =
      this.servings + servings;
  }
  void poop() {
    System.out.println("All better!");
    servings = 0;
  }
}
```

Overview

- Review
- Access control
- Class scope
- Packages
- Java API

Packages

- Each class belongs to a package
- Classes in the same package serve a similar purpose
- Packages are just directories
- Classes in other packages need to be imported

Defining Packages

```
package path.to.package.foo;
class Foo {
    ...
}
```

Using Packages

```
import path.to.package.foo.Foo;
import path.to.package.foo.*;
```

```
package parenttools;

public class BabyFood {

}
```

```
package parenttools;

public class Baby {

}
```

```
package adult;

import parenttools.Baby;
import parenttools.BabyFood;

public class Parent {
 public static void main(String[] args) {
    Baby baby = new Baby();
    baby.feed(new BabyFood());
 }
}
```

Eclipse Demo

Why Packages?

- Combine similar functionality
  - org.boston.libraries.Library
  - org.boston.libraries.Book
- Separate similar names
  - shopping.List
  - packing.List

Special Packages

All classes “see” classes in the same package (no import needed)

All classes “see” classes in java.lang

Example: java.lang.String; java.lang.System

Overview

- Review
- Access control
- Class scope
- Packages
- Java API

Java API

Java includes lots of packages/classes

Reuse classes to avoid extra work

http://java.sun.com/javase/6/docs/api/

Arrays with items

Create the array bigger than you need

Track the next “available” slot

```
Book[] books = new Book[10];
int nextIndex = 0;

books[nextIndex] = b;
nextIndex = nextIndex + 1;
```

Arrays with items

Create the array bigger than you need

Track the next “available” slot

```
Book[] books = new Book[10];
int nextIndex = 0;

books[nextIndex] = b;
nextIndex = nextIndex + 1;
```

What if the library expands?

ArrayList

Modifiable list

Internally implemented with arrays

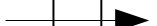
Features

- Get/put items by index
- Add items
- Delete items
- Loop over all items

Array → ArrayList

```
Book[] books =
      new Book[10];
int nextIndex = 0;


books[nextIndex] = b;
nextIndex += 1;
```



```
ArrayList<Book> books
= new ArrayList<Book>();


books.add(b);
```

```
import java.util.ArrayList;
class ArrayListExample {
    public static void main(String[] arguments) {
        ArrayList<String> strings = new ArrayList<String>();
        strings.add("Evan");
        strings.add("Eugene");
        strings.add("Adam");

        System.out.println(strings.size());
        System.out.println(strings.get(0));
        System.out.println(strings.get(1));

        strings.set(0, "Goodbye");
        strings.remove(1);
        for (int i = 0; i < strings.size(); i++) {
              System.out.println(strings.get(i));
         }
        for (String s : strings) {
            System.out.println(s);
        }
    }
}
```

Sets

Like an ArrayList, but

- Only one copy of each object, and
- No array index

Features

- Add objects to the set
- Remove objects from the set
- Is an object in the set?

TreeSet: Sorted (lowest to highest)

HashSet: Unordered (pseudo-random)

```
import java.util.TreeSet;

class SetExample {
    public static void main(String[] arguments) {
        TreeSet<String> strings = new TreeSet<String>();
        strings.add("Evan");
        strings.add("Eugene");
        strings.add("Adam");

        System.out.println(strings.size());
        System.out.println(strings.first());
        System.out.println(strings.last());

        strings.remove("Eugene");

        for (String s : strings) {
            System.out.println(s);
        }
    }
}
```

Maps

Stores a (*key*, *value*) pair of objects

Look up the *key*, get back the *value*

Example: Address Book

- Map from names to email addresses

TreeMap: Sorted (lowest to highest)

HashMap: Unordered (pseudo-random)

```
public static void main(String[] arguments) {
    HashMap<String, String> strings = new HashMap<String, String>();
    strings.put("Evan", "email1@mit.edu");
    strings.put("Eugene", "email2@mit.edu");
    strings.put("Adam", "email3@mit.edu”);

    System.out.println(strings.size());
    strings.remove("Evan");
    System.out.println(strings.get("Eugene"));

    for (String s : strings.keySet()) {
        System.out.println(s);
    }
    for (String s : strings.values()) {
        System.out.println(s);
    }
    for (Map.Entry<String, String> pairs : strings.entrySet()) {
        System.out.println(pairs);
    }
}
```

Warnings

Using TreeSet/TreeMap?

Read about Comparable interface

Using HashSet/HashMap?

Read about equals, hashCode methods

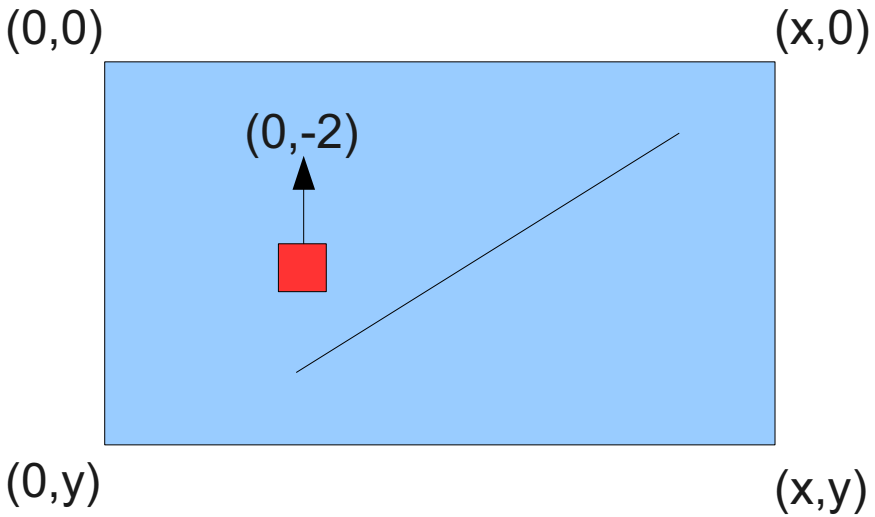
Note: This only matters for classes you build, not for java built-in types.

Summary

- Review
- Access control
- Class scope
- Packages
- Java API

Assignment: Graphics

- http://java.sun.com/javase/6/docs/api/java/awt/Graphics.html
- http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html



6.092 Introduction to Programming in Java

January (IAP) 2010