Announcements:
- Project 5 due Thursday 4/14 at 6pm
- Prelim 3 on Tuesday, 4/19

Previous Lecture:
- Defining a class:
  - Constructors
  - Methods with input parameters
  - Method toString

Today's Lecture:
- Defining a class:
  - static variables and methods
  - Method overloading
  - OOP review

Reading:
- JV: Sec 6.1-6.4, Sec 6.7

Static Variables & Methods
- Shared by all instances of a class
- Only one copy no matter how many objects have been instantiated
- Keyword: static
- Examples:
  - A constant used by the whole class
  - A variable to keep track of how many Intervals have been created
  - A method that doesn't need to reference fields

```java
class Interval {
    private double base;  // low end
    private double width; // interval width
    public static final double MAXwidth= 5; //...
    public Interval(double b, double w) {
        setBase(b);
       setWidth(w);
    }
    public void setBase(double base) {
        this.base= base;
    }
    /* Set width to w, w<=MAXwidth */
    public void setWidth(double w) {
        width= Math.min(w,MAXwidth);
    }
}
```

Class (static) method
Write a class method

\[
\text{overlap(Interval a, Interval b)}
\]

that returns a new Interval representing the overlap between Intervals a and b. (Return null if there’s no overlap)

What is the method header?
The overlap’s left
is the rightmost
of the two
original lefts

The overlap’s right
is the leftmost of
the two original
rights

No overlap if OLeft>ORight

/* =the overlapped Interval between
Intervals a and b */
public static Interval overlap(Interval a, 
Interval b) {
Interval olap; // overlapped interval
double left, right; // olap’s left & right
left = Math.max(a.getBase(),b.getBase());
right = Math.min(a.getEnd(),b.getEnd());
if ( (right-left) <= 0 )
  olap= null;
else
  olap= new Interval(left, right-left);
return olap;
}

public class Client {
  public static void main(String[] args){
    Interval i1= new Interval(0.2,0.7);
    Interval i2= new Interval(
        Math.random(),0.2);
    Interval o= Interval.overlap(i1,i2);
  }
}

An instance overlap method
- Write an instance method
  overlap(...) 
  that returns a new Interval if two Intervals overlap. Return null otherwise.
- What is the method header? What should be the parameters, if any?
- Are the static and instance versions very different?

Method overloading
- Different methods can have the same name
- A method has a signature: method name and the parameter types (including the order)
- In a class, all methods must have different signatures
- E.g., the abs method in the Math class
class Interval {
    private double base;  // low end
    private double width; // interval width
    public static final double maxWidth=5;
    public Interval(double b, double w) {
        setBase(b);
        setWidth(w);
    }
    public Interval() {} // An Interval with base b and maxWdith */
    public Interval(double b) {
        setBase(b);
        setWidth(maxWidth);
    }
    // other methods below
}

Method invocation
- Suppose there are 3 intervals: i1, i2, i3
- You know that i1 and i2 overlap
- Write code to find if the overlapped interval of i1 and i2 is in interval i3
  Interval i1 = new Interval(…);
  Interval i2 = new Interval(…);
  Interval i3 = new Interval(…);
  // Assume i1 and i2 overlap
  if (                             )
    System.out.println("in i3");
  else
    System.out.println("not in i3");

Review questions
- Where do you put declarations for instance and static variables?
  - Inside a method definition (e.g., inside method main)?
  - or
  - Immediately inside the class definition (immediately inside class Interval { … })?

Review questions, cont’d
- If a variable does not store a value of primitive type, what does it store?
  - What is the keyword for “protecting” instance variables? (I.e., not allow access from another class.)
- What is the keyword for instantiating an object?

What’s wrong with this?
/* Try to swap two values */
public class SwapValues {
    public static void main(String[] args){
        int x = Keyboard.readInt();
        int y = Keyboard.readInt();
        swap(x,y);
    }
    public static void swap(int x, int y) {
        int tmp;
        tmp = x;
        x = y;
        y = tmp;
    }
} //class SwapValues

A server class
   A client class

Data within objects should be protected: private
Provide only a set of methods for public access.