Class Variables & Methods

- Shared by all instances of a class
- Only one copy no matter how many objects have been instantiated
- Examples:
  - A variable to keep track of how many Intervals have been created
  - A constant used by the whole class

Class (static) method

- Write a class method
  
  `overlap(Interval a, Interval b)`

  that returns a new `Interval` if Intervals `a` and `b` overlap. Return null otherwise.

- What is the method header?

```java
class Interval {
    private double base;  // low end
    private double range; // interval width
    public final static double maxWidth=5;

    // Constructor, range always <= 5
    public Interval(double b, double r) {
        base = b;
        range = Math.min(r,maxWidth);
    }

    // Other methods below ...
}
```

```java
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);
    }
}
```

1. The overlap's left is the rightmost of the two original lefts
2. The overlap's right is the leftmost of the two original rights
3. No overlap if OLeft>ORight

The world of class Interval

- `overlap()`
- `base`
- `range`
- `maxWidth`
- `isIn()`
- `getEnd()`
- `expand()`

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);
    }
}
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

Method toString()
- Every object has default method toString
- Automatically invoked by print, println

    Interval a = new Interval(1,2);
    System.out.println(a);

- Some default text will be printed unless you define a toString method

Review: parameter passing
- Primitive type: “pass by value”
- Non-primitive type: “pass by reference”
- Let’s consider swapping two values...
  - double x = Math.random();
  - double y = Math.random();
  - double tmp = x;
  - x = y;
  - y = tmp;
  
    //Try to swap 2 values. What’s wrong with this?
    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 = x;
            x = y;
            y = tmp;
        }
    }

    class Interval {
        private double base; // low end
        private double range; // interval width
        public final static double maxWidth=5;
        public Interval(double base, double r) {
            this.base = base;
            range = Math.min(r,maxWidth);
        }
        public Interval() {} 
        public Interval(double base) {
            this.base = base;
            range = maxWidth;
        }
    }