Previous Lecture:
- OOP thinking
- Defining a class:
  - Instance variables
  - Instance methods
  - Getters and setters

Today's Lecture:
- Defining a class:
  - Constructors
  - Keyword this
  - Method toString
  - Static variables and methods

Reading: Sec 4.4, start reading Sec 5.1

Constructor
- A constructor is used to create objects
- Each class has a default constructor
- You can define your own constructor:
  \[
  \text{modifier class-name} \{ \text{parameter-list} \} \{
  \text{statements-list}
  \}
  \]
- Use public as the modifier for now
- an instance method that has no return type

public class IntervalClient {
    public static void main(String[] args) {
        Interval in1 = new Interval(3,1);
    }
}
class Interval {
    ...
    public Interval(double base, double w) {
        this.base = base;
        width = w;
    }
    ...
}

More instance methods with input parameters
- Write an instance method
  \[
  \text{expand(double f)}
  \]
  that expands the Interval by a factor of \( f \).
- What should be the method header?
- Parameter of primitive type
/** Expand this Interval by a factor of f */
public void expand(double f) {
    width *= f;
}

public class Client {
    public static void main(String[] args) {
        Interval i1 = new Interval(0.2, 0.7);
        double x = 2;
        i1.expand(x);
        System.out.println(i1.getEnd());
    }
}

Non-primitive input parameter
- Write an instance method
  public boolean isIn(Interval i) 
  that returns the boolean value true if the instance is in Interval i. Return false otherwise.

- Parameter of non-primitive type: pass-by-reference
  I.e., Reference is copied; object itself is not copied

/** ="this Interval is in i" */
public boolean isIn(Interval i) {
    return (getBase() >= i.getBase() &&
            getEnd() <= i.getEnd());
}

public boolean isIn(Interval i) {
    if (getBase() >= i.getBase() &&
        getEnd() <= i.getEnd() == true)
        return true;
    else
        return false;
}
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);
        if (i2.isIn(i1))
            System.out.println("Interval i2 "
                + "is in Interval i1.");
        else
            System.out.println("Interval i2 "
                + "is not in Interval i1.");
    }
}

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

class Interval {
    private double base;  // low end
    private double width; // interval width

    public Interval(double base, double w) {
        this.base = base;
        width = w;
    }

    /** =String description of Interval */
    public String toString() {
        return ";" + getBase() + "," + getEnd() + "\n";
    }
}

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);
        if (i2.isIn(i1))
            System.out.println(i2 + "is in" +
                i1);
        else
            System.out.println(i2 + "is not in" +
                i1);
    }
}
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 `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?

```java
/* =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;
}
```