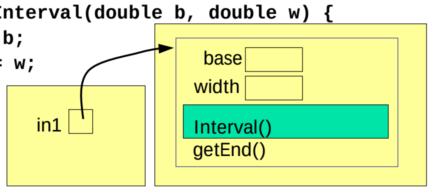
- Previous Lecture:
  - Instance variables
  - Instance methods, getters and setters
  - Constructor
- Today's Lecture:
  - Review
  - Defining a class:
    - Constructor
    - Keyword this
    - Method toString *(in lab this week)*
    - Methods with parameters
- Reading: Sec 9.1-9.8

```
public class Client {
  public static void main(String() args) {
    Interval in1= new Interval(3,0.1);
  }
}
class Interval {
  private double base, width;

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

Create an object

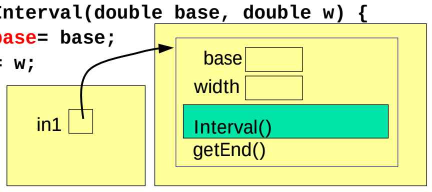


```
public Interval(double b, double w) {
     this.base= b;
     this.width= w;
}
```

- Keyword **this** returns a reference to the object itself, so **this.base** is "this" object's field **base**
- Use keyword **this** when it is necessary. (It is not necessary in the example above.)

```
public class IntervalClient {
  public static void main(String() args) {
    Interval in1= new Interval(3,1);
  }
}
class Interval {
  private double base, width;

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



More instance methods with input parameters

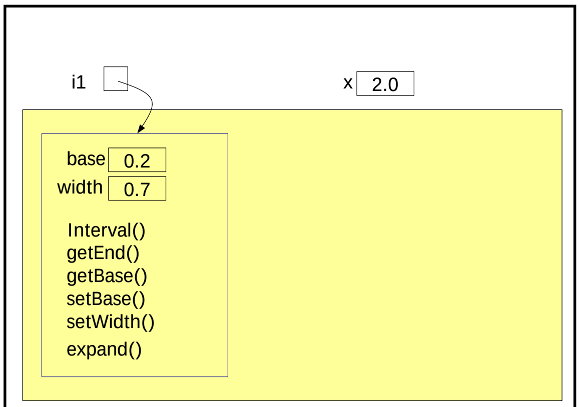
- Write an instance method

  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());
  }
}
```



```
/** Expand this Interval by a
 *  factor of f
 */
public void expand(double f) {
  setWidth(width*f);
}
```

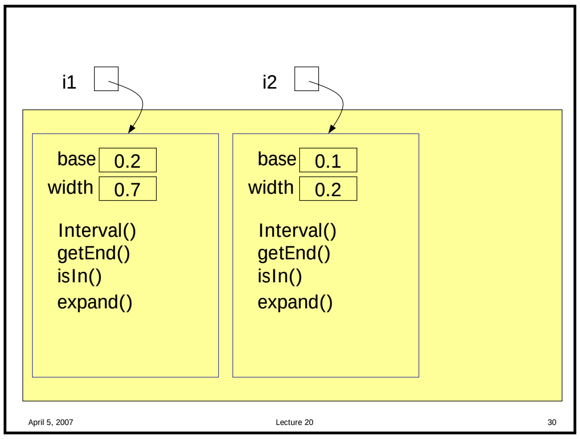
Use available methods when possible!

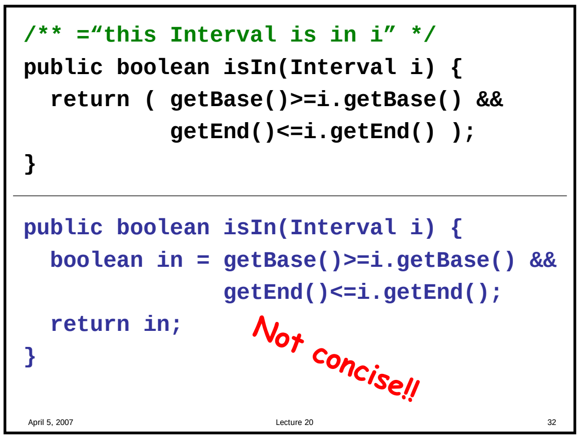
Non-primitive input parameter

- Write an instance method
  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 Interval i} */
public boolean isIn(Interval i) {
  return (     getBase()>=i.getBase() &&
               getEnd()<=i.getEnd() );
}
```

```
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);
  }
}
```

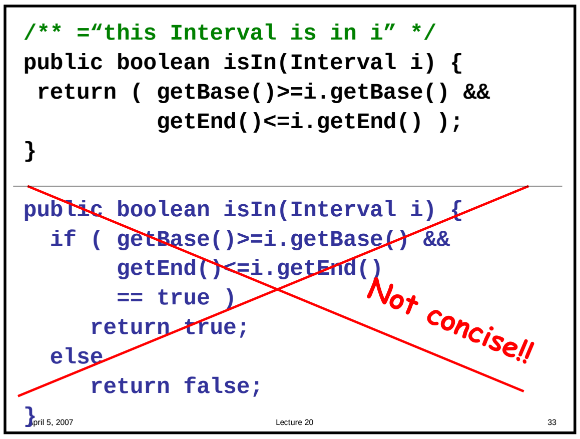




```
/** ="this Interval is in i" */
public boolean isIn(Interval i) {
  return ( getBase()>=i.getBase() &&
           getEnd()<=i.getEnd() );
}
```

```
public boolean isIn(Interval i) {
  boolean in = getBase()>=i.getBase() &&
               getEnd()<=i.getEnd();
  return in;
}
```

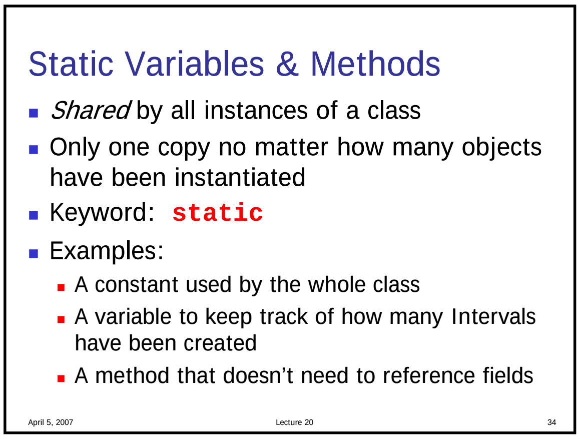
Not concise!!



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

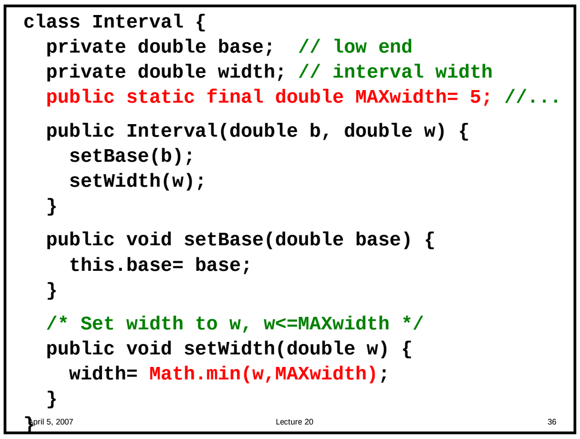
```
~~public boolean isIn(Interval i) {
  if ( getBase()>=i.getBase() &&
       getEnd()<=i.getEnd()
       == true )
     return true;
  else
     return false;
}~~
```

Not concise!!



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

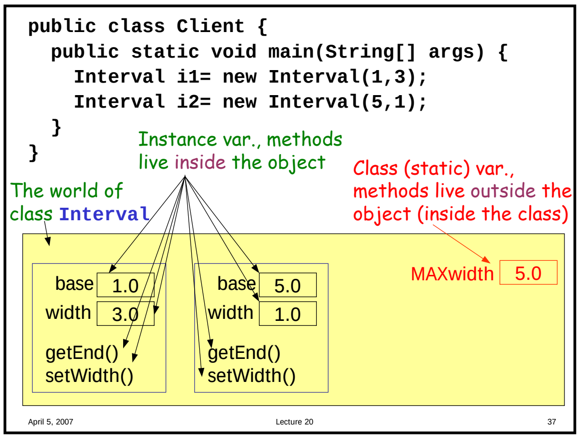


```
class Interval {
  private double base;  // low end
  private double width; // interval width
  public static final double MAXwidth= 5; //...

  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);
  }
}
```



```
public class Client {
  public static void main(String[] args) {
    Interval i1= new Interval(1,3);
    Interval i2= new Interval(5,1);
  }
}
```

Instance var., methods live inside the object
Class (static) var., methods live outside the object (inside the class)
The world of class Interval
base 1.0
width 3.0
getEnd()
setWidth()
base 5.0
width 1.0
getEnd()
setWidth()
MAXwidth 5.0

```
class Car {

  private String make;

  public boolean isManual;

  public static int totalCars=0;

  public Car(String s) {
    make= s;
    totalCars++;
  }

  public void setMake(String m) { make= m; }

  public String getMake() { return make; }
}


public class Client {
  public static void main(String[] args) {
    //Are the following statements valid?

    Car c1;
    System.out.println(c1.getMake());
    System.out.println(Car.getMake());
    System.out.println(c1.isManual);


    System.out.println(Car.totalCars);


    System.out.println(m);


    c1= new Car("VW");
    System.out.println(c1.getMake());
    System.out.println(c1.isManual);
    System.out.println(c1.make);

    Car c2= new Car("Ford");

    System.out.println(Car.totalCars);
    System.out.println(c1.totalCars);
    System.out.println(c2.totalCars);

  }
}
```