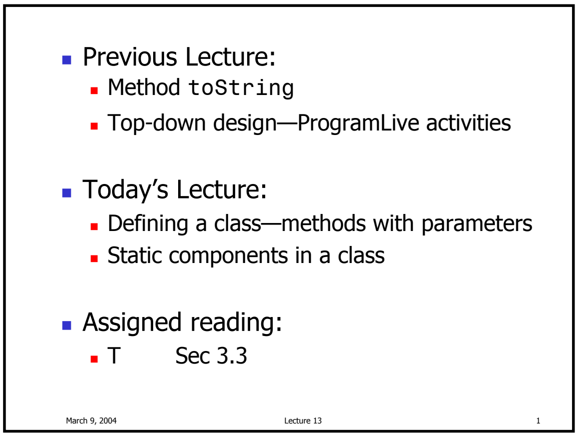
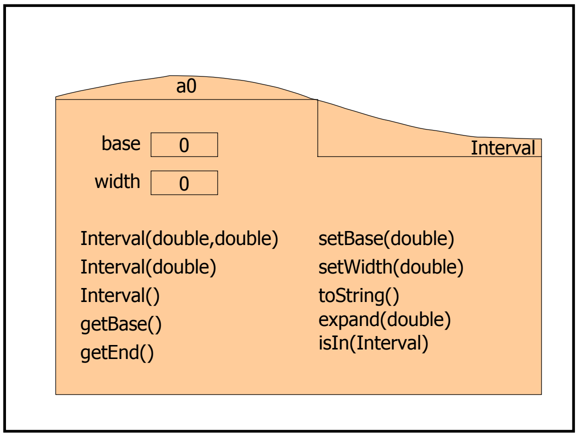


- Previous Lecture:
  - Method `toString`
  - Top-down design—ProgramLive activities
- Today's Lecture:
  - Defining a class—methods with parameters
  - Static components in a class
- Assigned reading:
  - T Sec 3.3



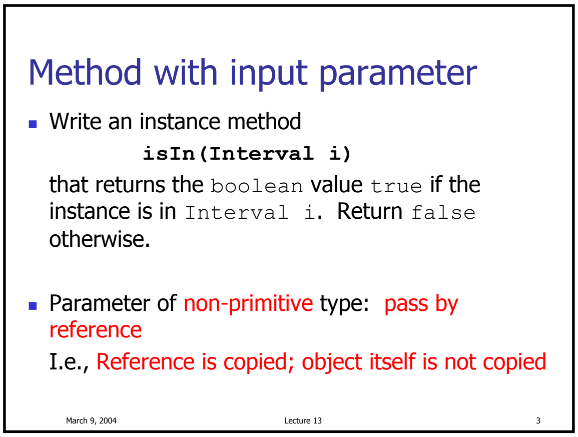
a0

Interval

base 0

width 0

Interval(double,double)	setBase(double)
Interval(double)	setWidth(double)
Interval()	toString()
getBase()	expand(double)
getEnd()	isIn(Interval)



Method with input parameter

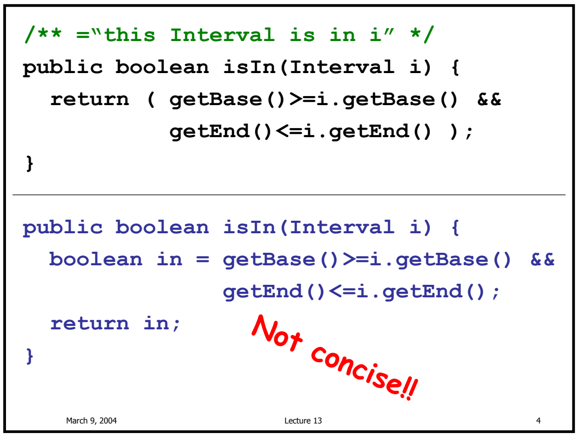
- Write an instance method

  `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

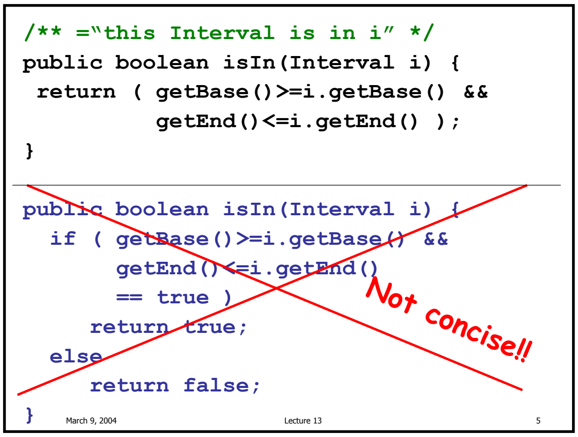
  I.e., Reference is copied; object itself is not copied



```
/** ="this Interval is in 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;
}
```

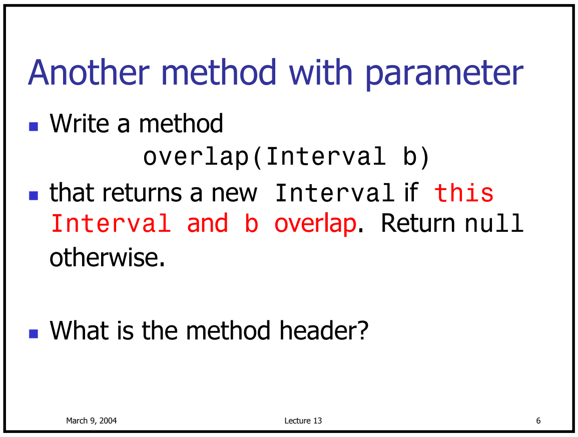
Not concise!!



```
/** ="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!!



Another method with parameter

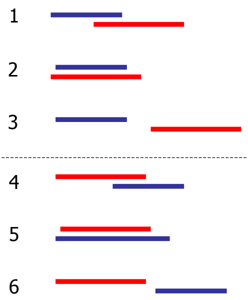
- Write a method

  `overlap(Interval b)`

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

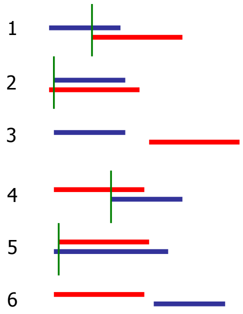
- What is the method header?

```
/** = the overlapped Interval between
 *  this Interval and Interval b */
```

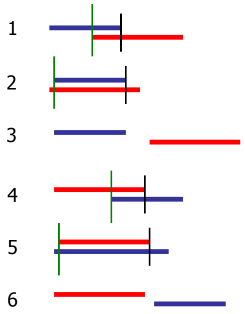


blueHigh < redHigh

redHigh < blueHigh

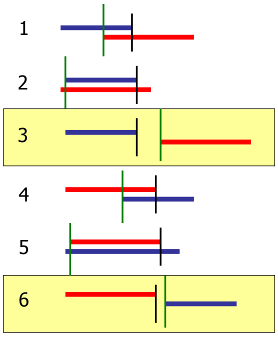


The overlap's left is the rightmost of the two original lefts



The overlap's left is the rightmost of the two original lefts

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

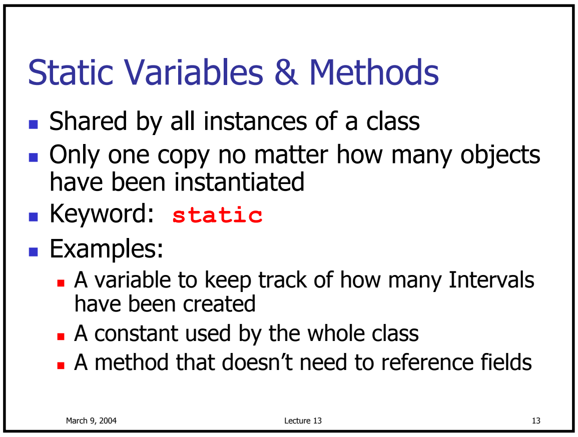


The overlap's left is the rightmost of the two original lefts

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

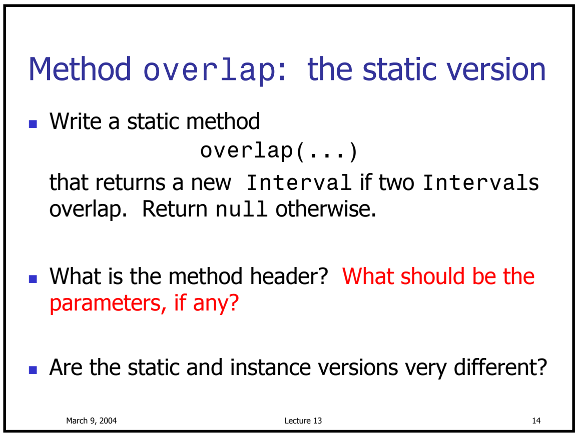
No overlap if OLeft>ORight

```
/** Client class that uses Interval class */
public class ClientInterval {
  public static void main(String[] args){

    Interval i1= new Interval(0.2,0.5);
    Interval i2= new Interval(
                     Math.random(),0.2);

    Interval o= i1.overlap(i2);

      System.out.println(o);
  }
}
```



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 variable to keep track of how many Intervals have been created
  - A constant used by the whole class
  - A method that doesn't need to reference fields



Method `overlap`: the static version

- Write a static 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?