COS126 Spring08 Programming Exam 2 (Overlap version)

```
/* Name:
 * Precept:
 * Comment: Creates a data type called "Segment" that consists of two numbers
 * on a number line. It also tests to see if the input segments overlap each
 * other, and prints out the pairs of segments that do overlap.
 */

public class Segment
{
    private double x0, x1; // co-ordinates of the endpoints of the segment

    // constructor to initialize Segment
    public Segment(double x0, double x1)
    {
        this.x0 = x0; // initialize the co-ordinates
        this.x1 = x1; // to the given values
    }

    // Is 'x' contained in this segment?
    public boolean contains(double x)
    {
        return (x0 < x && x <= x1);
    }

    // Does this Segment overlap 'that'?
    public boolean overlaps(Segment that)
    {
        // This is equivalent to asking: "does the right endpoint of
        // either of these two segments fall within the other segment?"

        // Note that this isn't the same as check the left endpoint,
        // because it isn't counted as part of the segment.

        return (this.contains(that.x1) || that.contains(this.x1));
    }

    // Return a string representation of this Segment.
    public String toString()
    {
        return x0 + " -- " + x1;
    }
    // test client
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]); // get number of inputs

        Segment[] segs = new Segment[N]; // create array to
                                         // hold the Segments

        // Read in all of the input.
        for (int i = 0; i < N; i++)
        {
            double x0 = StdIn.readDouble(); // read in the
            double x1 = StdIn.readDouble(); // co-ordinates

            segs[i] = new Segment(x0, x1); // create Segment objects
        }

        // Now check for pairs of overlapping Segments.
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < i; j++)
            {
                // If the ith and jth Segments overlap, print out a message.
                if (segs[j].overlaps(segs[i]))
                    System.out.println(segs[j] + " overlaps " + segs[i]);
            }
        }
    }
}
```

COS126 Spring08 Programming Exam 2 (No overlap version)

```
/* Name:
 * Precept:
 * Comment: Creates a data type called "Segment" that consists of two numbers
 * on a number line. It also tests to see if the input segments overlap each
 * other, and prints out the segments that don't overlap any other segment.
 */

public class Segment {
  // Instance variables
  private double x0; // Low end
  private double x1; // High end

  // Constructor
  public Segment(double x0, double x1) {
    this.x0 = x0;
    this.x1 = x1;
  }

  // Is 'x' contained in this segment?
  public boolean contains(double x) {
    return (x >= this.x0 && x < this.x1);
  }

  // Tests to see if two segments overlap each other
  public boolean overlaps(Segment that) {
    return (this.contains(that.x0) || that.contains(this.x0));
  }

  public String toString() {
    return (this.x0 + " -- " + this.x1);
  }
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);  // Reads in N

    Segment[] list = new Segment[N];    // Creates new array of Segments call "list"
    for (int i = 0; i < N; i++) {       // Reads in the data for "list"
      double temp1 = StdIn.readDouble();
      double temp2 = StdIn.readDouble();
      list[i] = new Segment(temp1, temp2);
    }

    for (int i = 0; i < N; i++) {
       // noOverlap is true if a segment does not overlap with any other in "list"
       boolean noOverlap = true;

      // for a given list[i], it is tested for overlapping against all other
      // segments in "list", except for itself.
      for (int j = 0; j < N; j++) {
        if (i != j) {
          if (list[i].overlaps(list[j]))
            noOverlap = false;
        }
      }
      // Given that noOverlap is true, that particular list[i] is printed.
      if (noOverlap) {
        System.out.println(list[i]);
      }
    }
  }
}
```