Exercise 1 – Comparables and Comparators
A Point2D is a class that represents 2D immutable points in the plane. It encapsulates a two-dimensional point with real-value coordinates. A partial code for a Point2D class is given below.

```java
import java.util.Arrays;
import java.util.Comparator;

public final class Point2D implements Comparable<Point2D> {
    private final double x;  // x coordinate
    private final double y;  // y coordinate

    //Compares two points by x-coordinate.
    public static final Comparator<Point2D> X_ORDER = new XOrder();

    //Compares two points by y-coordinate.
    public static final Comparator<Point2D> Y_ORDER = new YOrder();

    //creates a new 2D Point
    public Point2D(double x, double y) {
        this.x = x;
        this.y = y;
    }

    //Returns the square of the Euclidean distance between this point and that point.
    public double distanceSquaredTo(Point2D that) {
        double dx = this.x - that.x;
        double dy = this.y - that.y;
        return dx*dx + dy*dy;
    }
}
```

Exercise 1 - Comparables and Comparators
A Point2D is a class that represents 2D immutable points in the plane. It encapsulates a two-dimensional point with real-value coordinates. A partial code for a Point2D class is given below.

```java
import java.util.Arrays;
import java.util.Comparator;

public final class Point2D implements Comparable<Point2D> {
    private final double x;  // x coordinate
    private final double y;  // y coordinate

    //Compares two points by x-coordinate.
    public static final Comparator<Point2D> X_ORDER = new XOrder();

    //Compares two points by y-coordinate.
    public static final Comparator<Point2D> Y_ORDER = new YOrder();

    //creates a new 2D Point
    public Point2D(double x, double y) {
        this.x = x;
        this.y = y;
    }

    //Returns the square of the Euclidean distance between this point and that point.
    public double distanceSquaredTo(Point2D that) {
        double dx = this.x - that.x;
        double dy = this.y - that.y;
        return dx*dx + dy*dy;
    }
}
```

Exercise 1 – Comparables and Comparators
A Point2D is a class that represents 2D immutable points in the plane. It encapsulates a two-dimensional point with real-value coordinates. A partial code for a Point2D class is given below.

```java
import java.util.Arrays;
import java.util.Comparator;

public final class Point2D implements Comparable<Point2D> {
    private final double x;  // x coordinate
    private final double y;  // y coordinate

    //Compares two points by x-coordinate.
    public static final Comparator<Point2D> X_ORDER = new XOrder();

    //Compares two points by y-coordinate.
    public static final Comparator<Point2D> Y_ORDER = new YOrder();

    //creates a new 2D Point
    public Point2D(double x, double y) {
        this.x = x;
        this.y = y;
    }

    //Returns the square of the Euclidean distance between this point and that point.
    public double distanceSquaredTo(Point2D that) {
        double dx = this.x - that.x;
        double dy = this.y - that.y;
        return dx*dx + dy*dy;
    }
}
```

© Fall 2017 by COS 226 Staff.

September 28, 2017
*/
public int compareTo(Point2D that) {
    // to be completed
}

// Compares two points by distance to this point. Returns a Comparator.
public Comparator<Point2D> distanceToOrder(Point2D p) {
    return new DistanceToOrder(p);
}

// compare points according to their x-coordinate
private static class XOrder implements Comparator<Point2D> {
    public int compare(Point2D p, Point2D q) {
        // to be completed
    }
}

// compare points according to their y-coordinate
private static class YOrder implements Comparator<Point2D> {
    public int compare(Point2D p, Point2D q) {
        // to be completed
    }
}

// compare points according to their distance to this point
private class DistanceToOrder implements Comparator<Point2D> {
    public int compare(Point2D p, Point2D q) {
        // to be completed
    }
}

// returns a String representation of this point
public String toString() {
    return "(" + x + "," + y + ")";
}

public static void main(String[] args) {
    // client code to be completed
}
A. (Group Activity) Read the code in Point2D class and answer the following questions. Please write only brief answers in the space provided.

- What are the instance variables of the Point2D class?

- Why is it necessary to include a method called compareTo() in Point2D class?

- Name the three comparators declared in the above code.

B. (Individual Activity) Complete the code below that implements x-order compare() method.

```java
// compare points according to their x-coordinate
private static class XOrder implements Comparator<Point2D> {
    public int compare(Point2D p, Point2D q) {
        //complete code below
    }
}
```

C. (Individual Activity) Complete the code below that implements y-order compare() method.

```java
// compare points according to their y-coordinate
private static class YOrder implements Comparator<Point2D> {
    public int compare(Point2D p, Point2D q) {
        //complete code below
    }
}
```

D. (Individual Activity) Complete the code below that implements distanceTo-order compare() method.

```java
// compare two points p and q according to their distance to this point.
private class DistanceToOrder implements Comparator<Point2D> {
    Point2D origin;
    public DistanceToOrder(Point2D p) {
        origin = p;
    }
    public int compare(Point2D p, Point2D q) {
        //complete code below
    }
}
```
E. (Group Activity) Complete the missing code in compareTo() method

```java
/**
 * Compares two points by y-coordinate, breaking ties by x-coordinate.
 * Formally, the invoking point (x0, y0) < (x1, y1)
 * if and only if either y0 < y1 or if y0 == y1 and x0 < x1.
 */
public int compareTo(Point2D that) {
    //complete code below
}
```

F. (Group Activity) Here is some client/tester code for using the Point2D class. Complete the code as listed below.

```java
public static void main(String[] args) {
    int n = Integer.parseInt(args[2]);
    Point2D[] points = new Point2D[n];
    for (int i = 0; i < n; i++) {
        int x = StdRandom.uniform(10);
        int y = StdRandom.uniform(10);
        points[i] = new Point2D(x, y);
    }
    Point2D origin = new Point2D(0, 0);
    //sort the points array by x-order
    //sort the points array by y-order
    //sort the points array by default order (defined by compareTo())
    //sort the points array by distance to the origin
}
```
Exercise 2 – Counting Compares
Suppose that you have an array of length $2n$ consisting of $n$ B’s followed by $n$ A’s. Below is the array when $n = 10$.

B B B B B B B B A A A A A A A A

A. How many compares does it take to insertion sort (ascending order) the array, as a function of $n$? Use tilde notation to simplify your answer.

B. How many compares does it take to selection sort (ascending order) the array, as a function of $n$? Use tilde notation to simplify your answer.
Exercise 3 – 3-way Merge Sort

3-way merge sort is a modification of the merge sort algorithm that considers 3 sub arrays instead of 2 sub arrays.

A. Given three sorted subarrays of length $n/3$ each, design an algorithm to merge them into a sorted array of length $n$. As a function of $n$, how many compares does your algorithm make in the worst case? Use tilde notation to simplify your answer.

B. Argue that number of compares to sort an array of size $n$ using 3-way merge sort is still linearithmic.

C. Given a choice, would you choose 3-way or 2-way merge sort? Justify your answer.