

## IntSeq.java

```
1  /**
2   * Sequences of integers represented as arrays.
3   *
4   * <br/><strong>Code</strong>: <a href="IntSeq_java.html">For online reading</a>;
5   * <a href=" ../IntSeq.java">Java source folder</a>.
6   */
7  public class IntSeq {
8      private int[] data;
9      /**
10     * Create a sequence object from a given array.
11     *
12     * @param toCopy the array.
13     */
14     public IntSeq(int[] toCopy) {
15         data = new int[toCopy.length];
16         for (int i = 0; i < toCopy.length; i++)
17             data[i] = toCopy[i];
18     }
19     /**
20     * Get the length of the sequence.
21     *
22     * @return the length
23     */
24     public int size() { return data.length; }
25     /**
26     * Get an element of the sequence.
27     *
28     * @param i the index of the element
29     * @return the element
30     */
31     public int get(int i) { return data[i]; }
32     /**
33     * Does a word occur at a given position in this sequence?
34     *
35     * @param word the word.
36     * @param pos the position.
37     * @return whether the word occurs in this sequence at <code>pos</code>.
38     */
39     public boolean occursAt(IntSeq word, int pos) {
40         for (int i = 0; i < word.size(); i++) {
41             if (i + pos >= size() ||
42                 data[i+pos] != word.get(i))
43                 return false;
44         }
45         return true;
46     }
47     /**
48     * Find the position of the first occurrence of a word in this sequence.
49     *
50     * @param word the word.
51     * @return the position of <code>word</code>'s first occurrence, or
52     * -1 if <code>word</code> does not occur.
53     */
54     public int find(IntSeq word) {
```

```
55     int where = -1;
56     for (int pos = 0; pos <= size(); pos++)
57         if (occursAt(word, pos)) {
58             where = pos;
59             break;
60         }
61     return where;
62 }
63 public String toString() {
64     StringBuffer b = new StringBuffer();
65     for (int i = 0; i < data.length; i++) {
66         if (i > 0)
67             b.append(' ');
68         b.append(data[i]);
69     }
70     return b.toString();
71 }
72 }
```

## Ticket.java variant

```
1  /**
2   * Tickets with unique serial numbers and prices,
3   */
4  public class Ticket {
5      private static int issued;
6      private static double currentPrice = 5.0, revenue = 0.0;
7      private int number;
8      private double price;
9      /**
10     * Get the number of tickets issued so far
11     * @return the number of tickets so far
12     */
13     public static int getIssued() {
14         return issued;
15     }
16     /**
17     * Set the price for tickets issued from now on.
18     * @param price a price
19     */
20     public static void setCurrentPrice(double price) {
21         currentPrice = price;
22     }
23     /**
24     * Get revenue from issued tickets
25     * @return the revenue
26     */
27     public static double getRevenue() {
28         return revenue;
29     }
30     /**
31     * Create a new ticket, with its own unique serial number.
32     */
33     public Ticket() {
34         number = issued++;
35         price = currentPrice;
36         revenue += price;
37     }
38     /**
39     * Get the ticket's serial number
40     * @return the serial number
41     */
42     public int getNumber() {
43         return number;
44     }
45     /**
46     * Get this ticket's price.
47     * @return a price
48     */
49     public double getPrice() {
50         return price;
51     }
52     public String toString() {
53         return number + "@" + price;
54     }
55 }
```

## Shape classes and interfaces

### Displaceable.java

```
1  /**
2   * Type of shapes that have a location and can be moved.
3   * <br/>
4   * <strong>Code</strong>: <a href="Displaceable_java.html">For online reading</a>;
5   * <a href=" ../Displaceable.java">Java source</a>.
6   */
7  public interface Displaceable {
8      /**
9       * Get the X coordinate of this shape's location.
10     *
11     * @return the X coordinate
12     */
13     public double getX();
14     /**
15     * Get the X coordinate of this shape's location.
16     *
17     * @return the X coordinate
18     */
19     public double getY();
20     /**
21     * Move this shape by a given displacement vector.
22     *
23     * @param dx the X length of the displacement
24     * @param dy the Y length of the displacement
25     */
26     public void move(double dx, double dy);
27 }
```

### Area.java

```
1  /**
2   * Type of shapes that have area.
3   * <br/>
4   * <strong>Code</strong>: <a href="Area_java.html">For online reading</a>;
5   * <a href=" ../Area.java">Java source</a>.
6   */
7  public interface Area {
8      /**
9       * Get the area of this shape.
10     *
11     * @return the area
12     */
13     public double getArea();
14 }
```

### Point.java

```
1  /**
2   * A point on the plane, with <code>double</code> coordinates.
3   * <br/>
4   * <strong>Code</strong>: <a href="Point_java.html">For online reading</a>;
5   * <a href=" ../Point.java">Java source</a>.
6   */
```

```

7 public class Point implements Displaceable {
8     private double x, y;
9     /**
10      * Create a point at a given location.
11      * @param anX the X coordinate of the point
12      * @param aY the Y coordinate of the point
13      */
14     public Point(double anX, double aY) {
15         x = anX;
16         y = aY;
17     }
18     /**
19      * Get the X coordinate.
20      * @return the X coordinate
21      */
22     public double getX() { return x; }
23     /**
24      * Get the Y coordinate.
25      * @return the Y coordinate
26      */
27     public double getY() { return y; }
28     /**
29      * Move point by <var>dx</var>, <var>dy</var>.
30      * @param dx X displacement
31      * @param dy Y displacement
32      */
33     public void move(double dx, double dy) {
34         x = x + dx;
35         y = y + dy;
36     }
37     public String toString() {
38         return "(" + x + ", " + y + ")";
39     }
40     public String altToString() {
41         return super.toString();
42     }
43 }

```

## Circle.java

```

1 /**
2  * A circle specified by its center and radius.
3  * <br/>
4  * <strong>Code</strong>: <a href="Circle_java.html">For online reading</a>;
5  * <a href=" ../Circle.java">Java source</a>.
6  */
7 public class Circle implements Displaceable, Area {
8     private Point center;
9     private double radius;
10    /**
11     * Create a <code>Circle</code> instance with given center
12     * <code>Point</code> and radius.
13     * @param center the center <code>Point</code>
14     * @param radius the radius
15     */
16    public Circle(Point center, double radius) {

```

```

17     this.center = center;
18     this.radius = radius;
19 }
20 /**
21  * Get the center of this circle
22  * @return the center of this circle
23  */
24 public Point getCenter() {
25     return center;
26 }
27 /**
28  * Get the radius of this circle
29  * @return the radius
30  */
31 public double getRadius() {
32     return radius;
33 }
34 /**
35  * Get the X coordinate of this circle's center.
36  *
37  * @return the X coordinate
38  */
39 public double getX() { return center.getX(); }
40 /**
41  * Get the Y coordinate of this circle's center.
42  *
43  * @return the Y coordinate
44  */
45 public double getY() { return center.getY(); };
46 /**
47  * Translate this circle by translating its center.
48  * @param dx the x-coordinate displacement
49  * @param dy the y-coordinate displacement
50  */
51 public void move(double dx, double dy) {
52     center.move(dx, dy);
53 }
54 /**
55  * Get the area of the circle.
56  *
57  * @return the area
58  */
59 public double getArea() {
60     return Math.PI * radius * radius;
61 }
62 public String toString() {
63     return center + "->" + radius;
64 }
65 }

```

## Rectangle.java

```

1  /**
2  * A rectangle defined by its lower-left corner, width, and height.
3  * <br/>
4  * <strong>Code</strong>: <a href="Rectangle_java.html">For online reading</a>;

```

```

5  * <a href="../Rectangle.java">Java source</a>.
6  */
7  public class Rectangle implements Displaceable, Area {
8      private Point lowerLeft;
9      private double width, height;
10     /**
11      * A new rectangle with given lower-left corner point, width, and height.
12      *
13      * @param lowerLeft the lower left corner
14      * @param width the width
15      * @param height the height
16      */
17     public Rectangle(Point lowerLeft, double width, double height) {
18         this.lowerLeft = lowerLeft;
19         this.width = width;
20         this.height = height;
21     }
22     /**
23      * Get the X coordinate of the rectangle's lower-left corner.
24      *
25      * @return the X coordinate
26      */
27     public double getX() { return lowerLeft.getX(); }
28     /**
29      * Get the Y coordinate of the rectangle's lower-left corner.
30      *
31      * @return the Y coordinate
32      */
33     public double getY() { return lowerLeft.getY(); }
34     /**
35      * Get the width of the rectangle.
36      *
37      * @return the width
38      */
39     public double getWidth() { return width; }
40     /**
41      * Get the height of the rectangle.
42      *
43      * @return the height
44      */
45     public double getHeight() { return height; }
46     /**
47      * Move the rectangle by a displacement vector.
48      *
49      * @param dx the X displacement
50      * @param dy the Y displacement
51      */
52     public void move(double dx, double dy) {
53         lowerLeft.move(dx, dy);
54     }
55     /**
56      * Get the area of the rectangle.
57      *
58      * @return the area
59      */
60     public double getArea() {

```

```

61     return width * height;
62 }
63 public String toString() {
64     return lowerLeft + "[" + width + " * " + height + "];
65 }
66 }

```

### Square.java

```

1  /**
2   * A Rectangle with identical width and height.
3   *   

4   * Code: For online reading;
5   * Java source.
6   */
7  public class Square extends Rectangle {
8      /**
9       * A new square with given lower-left corner and side.
10     *
11     * @param lowerLeft X lower-left corner
12     * @param side the side length
13     */
14     public Square(Point lowerLeft, double side) {
15         super(lowerLeft, side, side);
16     }
17     public String toString() {
18         return "(" + getX() + "," + getY() + ")" + "[" + getWidth() + "^2]";
19     }
20 }

```