```
1   public class PointST<Value> {
2
3       private BST<Point2D, Value> st;
4
5       // construct an empty symbol table of points
6       public PointST() {
7           st = new BST<Point2D, Value>();
8       }//end constructor
9
10      // is the symbol table empty?
11      public boolean isEmpty() {
12          return st.isEmpty();
13      }//end isEmpty
14
15      // number of points in the ST
16      public int size() {
17          return st.size();
18      }//end size
19
20      // add the point p to the ST or if it already exists, update
21      public void insert(Point2D p, Value v) {
22          st.put(p, v);
23      }//end insert
24
25      // returns value mapped to by p
26      public Value get(Point2D p) {
27          return st.get(p);
28      }//end get
29
30      // does the ST contain the point p?
31      public boolean contains(Point2D p) {
32          return st.contains(p);
33      }//end contains
34
35      // draw points to standard draw
36      public void draw() {
37          Iterable<Point2D> list = st.keys();
38          for(Point2D key: list) {
39              key.draw();
40          }
41      }//end draw
42
43      // all points in the ST that are inside the rectangle
44      public Iterable<Point2D> range(RectHV rect) {
45
46          //get an iterable list of all points within the y range
47          Point2D ymin = new Point2D(rect.ymin(),rect.xmin());
48          Point2D ymax = new Point2D(rect.ymax(),rect.xmax());
49          Iterable<Point2D> list = st.keys(ymin,ymax);
50
51          //determine which of those points are within x range
52          Queue<Point2D> queue = new Queue<Point2D>();
53          for(Point2D key: list) {
54              if(rect.xmin() <= key.x() && key.x() <= rect.xmax()) {
55                  queue.enqueue(key);
56              }
57          }//end loop
58
59          return queue;
60
61      }//end range
62
63      // a nearest neighbor in the ST to p; null if set is empty
64      public Point2D nearest(Point2D p) {
65
66          //return null if empty set
67          if(st.isEmpty()) {
68              return null;
69          }
70
```

```
71          //temp variables to iterate through points
72          Point2D near = st.min();
73          double dist = p.distanceSquaredTo(near);
74          Iterable<Point2D> list = st.keys();
75
76          //loop through all the points, checking distance to each
77          for(Point2D key: list) {
78              //check if this is closer than current point
79              if(p.distanceSquaredTo(key) < dist && p.compareTo(key) != 0) {
80                  near = key;
81                  dist = p.distanceSquaredTo(key);
82              }
83          }//end loop
84
85          return near;
86      }//end nearest
87
88      // unit testing of the methods (not graded)
89      public static void main(String[] args) {
90
91          //test constructor, insert, and draw
92          PointST<Integer> points = new PointST<Integer>();
93          for (int i = 0; i < 100; i++) {
94              int x = StdRandom.uniform(100);
95              int y = StdRandom.uniform(100);
96              points.insert(new Point2D(x, y),1);
97          }
98          StdDraw.setCanvasSize(600, 600);
99          StdDraw.setXscale(0, 100);
100         StdDraw.setYscale(0, 100);
101         StdDraw.setPenRadius(.01);
102         points.draw();
103
104         //test range
105         RectHV rect = new RectHV(20, 20, 40, 40);
106         rect.draw();
107         Iterable<Point2D> list = points.range(rect);
108         StdDraw.setPenColor(StdDraw.RED);
109         StdDraw.setPenRadius(.02);
110         for(Point2D key: list) {
111             key.draw();
112         }//end loop
113
114         //test nearest
115         Point2D center = new Point2D(50, 50);
116         points.insert(center,1);
117         StdDraw.setPenColor(StdDraw.BLUE);
118         center.draw();
119         StdDraw.setPenColor(StdDraw.BLACK);
120         StdDraw.setPenRadius(.005);
121         center.drawTo(points.nearest(center));
122
123
124     }//end main
125
126 }//end class
```