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

class Point {
	double x;
	double y;
	public Point(int x, int y) {
		this.x = x;
		this.y = y;
	}
}

public class KNearestPoint {

	public static double getDist(Point p) {
		return p.x * p.x + p.y * p.y;
	}

	public static Point[] findKNearestPoints2(Point[] points, int k) {
		if (k <= 0 || points == null || points.length == 0) return new Point[0];

		Arrays.sort(points, new Comparator<Point>() {
			@Override
			public int compare (Point p1, Point p2) {
				double d1 = getDist(p1);
				double d2 = getDist(p2);
				if (d1 > d2) return 1;
				else if (d2 > d1) return -1;
				else return 0;
			}
		});

		int len = points.length >= k ? k : points.length;
		Point[] res = new Point[len];

		while (len > 0) {
			res[len-1] = points[len-1];
			len--;
		}

		return res;
	}

	public static Point[] findKNearestPoints1(Point[] points, int k) {
		if (k <= 0 || points == null || points.length == 0) return new Point[0];

		PriorityQueue<Point> pq = new PriorityQueue<>(k, new Comparator<Point>() {
			@Override
			public int compare (Point p1, Point p2) {
				double d1 = getDist(p1);
				double d2 = getDist(p2);
				if (d1 > d2) return -1;
				else if (d2 > d1) return 1;
```

```
                else return 0;
            }
        });

        for (Point point : points) {
            pq.offer(point);
            if (pq.size() > k) {
                pq.poll();
            }
        }

        int len = pq.size();
        Point[] res = new Point[len];
        for (int i = len-1; i >= 0; i--) {
            res[i] = pq.poll();
        }

        return res;
    }

    public static void main(String[] args) {
        Point[] arr = new Point[4];
        arr[0] = new Point(1, 0);
        arr[1] = new Point(2, 1);
        arr[2] = new Point(1, 5);
        arr[3] = new Point(1, 1);

        for (Point point : findKNearestPoints2(arr, 10)) {
            System.out.println(point.x + " " + point.y);
        }
    }
}
```