

Lösningsförslag, omkontrollskrivning PTDC

2014–08–21

```
1. public class Map {
    private ArrayList<City> cities;
    private double[][] dist;

    public Map(String fileName) {
        cities = new ArrayList<City>();
        Scanner scan = null;
        try {
            scan = new Scanner(new File(fileName));
        } catch (FileNotFoundException e) {
            System.err.println("Cannot open: " + fileName);
            System.exit(1);
        }
        while (scan.hasNextDouble()) {
            double x = scan.nextDouble();
            double y = scan.nextDouble();
            cities.add(new City(x, y));
        }
        int n = cities.size();
        dist = new double[n][n];
        for (int i = 1; i < n; i++) {
            for (int j = 0; j < i; j++) {
                double dx = cities.get(i).getX() - cities.get(j).getX();
                double dy = cities.get(i).getY() - cities.get(j).getY();
                dist[i][j] = dist[j][i] = Math.hypot(dx, dy);
            }
        }
    }

    public int getSize() {
        return cities.size();
    }

    public City getCity(int i) {
        return cities.get(i);
    }

    public double getDist(int i, int j) {
        return dist[i][j];
    }
}

2. public class Tour {
    private Map map;
    private int[] tour;

    public Tour(Map map) {
        this.map = map;
        tour = new int[map.getSize()];
    }
}
```

```

public void createRandomTour() {
    for (int i = 0; i < tour.length; i++) {
        tour[i] = i;
    }
    Random rand = new Random();
    for (int i = tour.length - 1; i > 0; i--) {
        int j = rand.nextInt(i + 1);
        swap(i, j);
    }
}

private void swap(int i, int j) {
    int temp = tour[i];
    tour[i] = tour[j];
    tour[j] = temp;
}

public void draw(Window w) {
    City start = map.getCity(tour[0]);
    w.moveTo(start.getX(), start.getY());
    for (int i = 1; i < tour.length; i++) {
        City c = map.getCity(tour[i]);
        w.lineTo(c.getX(), c.getY());
    }
    w.lineTo(start.getX(), start.getY());
}

public double getLength() {
    double length = 0;
    for (int i = 1; i < tour.length; i++) {
        length += map.getDist(tour[i - 1], tour[i]);
    }
    length += map.getDist(tour[tour.length - 1], tour[0]);
    return length;
}
}

3. public void createNearestNeighborTour() {
    boolean[] visited = new boolean[map.getSize()];
    int current = 0;
    tour[0] = current;
    visited[current] = true;
    for (int i = 1; i < tour.length; i++) {
        double minDist = Double.MAX_VALUE;
        int minIndex = 0;
        for (int j = 0; j < tour.length; j++) {
            if (!visited[j] && map.getDist(current, j) < minDist) {
                minDist = map.getDist(current, j);
                minIndex = j;
            }
        }
        current = minIndex;
        tour[i] = current;
        visited[current] = true;
    }
}

```

```
4.    public void improve2Opt() {
        boolean improved = true;
        while (improved) {
            improved = false;
            int i = 0;
            while (i < tour.length - 2 && !improved) {
                int j = i + 2;
                while (j < tour.length && !improved) {
                    if (getDifference(i, j) < 0) {
                        reverse(i + 1, j);
                        improved = true;
                    } else {
                        j++;
                    }
                }
                i++;
            }
        }
    }

    private double getDifference(int i, int j) {
        int j1 = (j + 1) % tour.length;
        double d1 = map.getDist(tour[i], tour[i + 1]);
        double d2 = map.getDist(tour[j], tour[j1]);
        double d3 = map.getDist(tour[i], tour[j]);
        double d4 = map.getDist(tour[j1], tour[i + 1]);
        return (d3 + d4) - (d1 + d2);
    }

    private void reverse(int low, int high) {
        while (high > low) {
            swap(high, low);
            high--;
            low++;
        }
    }
}
```
