```java
import java.util.ArrayList;
import java.util.Collections;
class City {
    int x;
    int y;

    // Constructs a randomly placed city
    public City(){
        this.x = (int)(Math.random()*200);
        this.y = (int)(Math.random()*200);
    }

    // Constructs a city at chosen x, y location
    public City(int x, int y){
        this.x = x;
        this.y = y;
    }

    // Gets city's x coordinate
    public int getX(){
        return this.x;
    }

    // Gets city's y coordinate
    public int getY(){
        return this.y;
    }

    // Gets the distance to given city
    public double distanceTo(City city){
        int xDistance = Math.abs(getX() - city.getX());
        int yDistance = Math.abs(getY() - city.getY());
        double distance = Math.sqrt( (xDistance*xDistance) + (yDistance*yDistance) );

        return distance;
    }

    @Override
    public String toString(){
        return getX()+", "+getY();
    }

}

 class TourManager {

    // Holds our cities
    private static ArrayList destinationCities = new ArrayList<City>();

    // Adds a destination city
    public static void addCity(City city) {
        destinationCities.add(city);
    }

    // Get a city
    public static City getCity(int index){
        return (City)destinationCities.get(index);
    }

    // Get the number of destination cities
    public static int numberOfCities(){
        return destinationCities.size();
    }

}
```

```
}
 class Tour{

    // Holds our tour of cities
    private ArrayList tour = new ArrayList<City>();
    // Cache
    private int distance = 0;

    // Constructs a blank tour
    public Tour(){
        for (int i = 0; i < TourManager.numberOfCities(); i++) {
            tour.add(null);
        }
    }

    // Constructs a tour from another tour
    public Tour(ArrayList tour){
        this.tour = (ArrayList) tour.clone();
    }

    // Returns tour information
    public ArrayList getTour(){
        return tour;
    }

    // Creates a random individual
    public void generateIndividual() {
        // Loop through all our destination cities and add them to our tour
        for (int cityIndex = 0; cityIndex < TourManager.numberOfCities(); cityIndex++) {
          setCity(cityIndex, TourManager.getCity(cityIndex));
        }
        // Randomly reorder the tour
        Collections.shuffle(tour);
    }

    // Gets a city from the tour
    public City getCity(int tourPosition) {
        return (City)tour.get(tourPosition);
    }

    // Sets a city in a certain position within a tour
    public void setCity(int tourPosition, City city) {
        tour.set(tourPosition, city);
        // If the tours been altered we need to reset the fitness and distance
        distance = 0;
    }

    // Gets the total distance of the tour
    public int getDistance(){
        if (distance == 0) {
            int tourDistance = 0;
            // Loop through our tour's cities
            for (int cityIndex=0; cityIndex < tourSize(); cityIndex++) {
                // Get city we're traveling from
                City fromCity = getCity(cityIndex);
                // City we're traveling to
                City destinationCity;
                // Check we're not on our tour's last city, if we are set our
                // tour's final destination city to our starting city
                if(cityIndex+1 < tourSize()){
                    destinationCity = getCity(cityIndex+1);
                }
                else{
                    destinationCity = getCity(0);
                }
                // Get the distance between the two cities
                tourDistance += fromCity.distanceTo(destinationCity);
            }
            distance = tourDistance;
        }
        return distance;
    }

    // Get number of cities on our tour
    public int tourSize() {
```

```
        return tour.size();
    }

    @Override
    public String toString() {
        String geneString = "|";
        for (int i = 0; i < tourSize(); i++) {
            geneString += getCity(i)+"|";
        }
        return geneString;
    }
}
public class SimulatedAnnealing {

    // Calculate the acceptance probability
    public static double acceptanceProbability(int energy, int newEnergy, double tempera
        // If the new solution is better, accept it
        if (newEnergy < energy) {
            return 1.0;
        }
        // If the new solution is worse, calculate an acceptance probability
        return Math.exp((energy - newEnergy) / temperature);
    }

    public static void main(String[] args) {
        // Create and add our cities
        City city = new City(60, 200);
        TourManager.addCity(city);
        City city2 = new City(180, 200);
        TourManager.addCity(city2);
        City city3 = new City(80, 180);
        TourManager.addCity(city3);
        City city4 = new City(140, 180);
        TourManager.addCity(city4);
        City city5 = new City(20, 160);
        TourManager.addCity(city5);
        City city6 = new City(100, 160);
        TourManager.addCity(city6);
        City city7 = new City(200, 160);
        TourManager.addCity(city7);
        City city8 = new City(140, 140);
        TourManager.addCity(city8);
        City city9 = new City(40, 120);
        TourManager.addCity(city9);
        City city10 = new City(100, 120);
        TourManager.addCity(city10);
        City city11 = new City(180, 100);
        TourManager.addCity(city11);
        City city12 = new City(60, 80);
        TourManager.addCity(city12);
        City city13 = new City(120, 80);
        TourManager.addCity(city13);
        City city14 = new City(180, 60);
        TourManager.addCity(city14);
        City city15 = new City(20, 40);
        TourManager.addCity(city15);
        City city16 = new City(100, 40);
        TourManager.addCity(city16);
        City city17 = new City(200, 40);
        TourManager.addCity(city17);
        City city18 = new City(20, 20);
        TourManager.addCity(city18);
        City city19 = new City(60, 20);
        TourManager.addCity(city19);
        City city20 = new City(160, 20);
        TourManager.addCity(city20);

        // Set initial temp
        double temp = 10000;

        // Cooling rate
        double coolingRate = 0.003;

        // Initialize intial solution
        Tour currentSolution = new Tour();
```

```
215             currentSolution.generateIndividual();
216
217             System.out.println("Initial solution distance: " + currentSolution.getDistance()
218
219             // Set as current best
220             Tour best = new Tour(currentSolution.getTour());
221
222             // Loop until system has cooled
223             while (temp > 1) {
224                 // Create new neighbour tour
225                 Tour newSolution = new Tour(currentSolution.getTour());
226
227                 // Get a random positions in the tour
228                 int tourPos1 = (int) (newSolution.tourSize() * Math.random());
229                 int tourPos2 = (int) (newSolution.tourSize() * Math.random());
230
231                 // Get the cities at selected positions in the tour
232                 City citySwap1 = newSolution.getCity(tourPos1);
233                 City citySwap2 = newSolution.getCity(tourPos2);
234
235                 // Swap them
236                 newSolution.setCity(tourPos2, citySwap1);
237                 newSolution.setCity(tourPos1, citySwap2);
238
239                 // Get energy of solutions
240                 int currentEnergy = currentSolution.getDistance();
241                 int neighbourEnergy = newSolution.getDistance();
242
243                 // Decide if we should accept the neighbour
244                 if (acceptanceProbability(currentEnergy, neighbourEnergy, temp) > Math.rando
245                     currentSolution = new Tour(newSolution.getTour());
246                 }
247
248                 // Keep track of the best solution found
249                 if (currentSolution.getDistance() < best.getDistance()) {
250                     best = new Tour(currentSolution.getTour());
251                 }
252
253                 // Cool system
254                 temp *= 1-coolingRate;
255             }
256
257             System.out.println("Final solution distance: " + best.getDistance());
258             System.out.println("Tour: " + best);
259         }
260     }
261
```

Execute Mode, Version, Inputs & Arguments

CommandLine Arguments

Result

**compiled and executed in 0.863 sec(s)**

```
Initial solution distance: 2516
Final solution distance: 863
Tour: |20, 160|40, 120|60, 80|20, 40|20, 20|60, 20|100, 40|120, 80|160, 20|200, 40|180, 60|180
```

**Note:**

**1. For file operations - upload files using upload button, Files will be upload to /uploads folder. You can read those files in program from /uploads folder. To write a file from your program, write files to '/myfiles' folder. Please note the uploaded files stored in the server only for the current session.**

**2. For detailed documentation check - Our Documentation, or check our Youtube channel.**

Thanks for using our

# Online Java Compiler IDE

to execute your program



## Know Your JDoodle

- JDoodle Supports 76+ Languages with Multiple Versions and 2 DBs. **Click here** to see all.
- Fullscreen - side-by-side code and output is available. click the "⛶" icon near execute button to switch.
- Dark Theme available. Click on "•••" icon near execute button and select dark theme.
- You can embed code from JDoodle directly into your website/blog. **Click here** to know more.
- JDoodle offers an API service. You can execute programs just by calling our API. **Click here** to know more.
- If you like JDoodle, Please share us in Social Media. **Click here** to share.
- Check our **Documentation Page** for more info.

JDoodle is serving the programming community since 2013

## JDoodle For Your Organisation

- Do you have any specific compiler requirements?
- Do you want to integrate compilers with your website, webapp, mobile app, courses?
- Do you need more than our **Embed** and **API** features?
- Looking for Multiple Files, Connecting to DB, Debugging, etc.?
- Are you building any innovative solution for your students or recruitment?
- Want to run JDoodle in-house?
- Custom Domain, White labelled pages for your institute?

Contact us - We are happy to help!