Individual class (Objekt)
2 Attribute : int [] order, double cost.
Konstructor(nur int [] order) , getter und Setter werden erstellt.

Method Crossover: Individual crossover(Individual partner)
Um gleich Ort nicht 2 mals zu besuchen, benutzt man Partially Mapped Crossover(PMX),PMX tauschen nicht nur Elemente aus 2 Eltern, auch tauschen die Elemente in den Eltern. Hier benutzt man Collections.swap() Methode. Man bekommt am Ende ein Kind.

Method Mutation : void mutate(float mutationChances)
zu mutieren zufällig benutzt man auch Collections.swap(), die 2 getauschte Orte werden werden mit random.nextInt() erzeugt. Dann am Ende bekommt die Eltern ein Mutationskind.

Projekt class(main) Part 1:

1. Initiealisierung:
   die Quantität der Population (int populationNr) ,
   die Quantität der Generation (int generationStopCount),
   die Quantität der Parents(int parentsNr),
   die Rate der Mutation(float mutationRate)
   werden die Werte gegeben.

2.Method: makeRandomOrder(double [] []) : int[]

Die Datei von Orts (von Datei)werde gelesen und im Array (double [] [] koord)gespeichert.
Int city speichert einen Ort.
Int [] order speichert die Reihenfolge.
boolean [] visited ist true, wenn der Ort besucht wird, der Wert der City wird nach int [] order zugewiesen , wenn nein, es ist falsch.

Man schriebt ein for-loop, do : eine neue wert wurde von (int)(Math.random()*koord.length) nach int city zugeweisen, solange(while) der Ort besucht wurde(inzwischen order[i]=city, visited[city]=true).

Return order.

3. Method: double calCost(int [] order, double[][] koord)
mit Hilfe des makeRandomOrder bekommt man die Werte von int [] order,
koord ist ein 2 Dimention matrix, cost zwischen 2 Orte kann man berechenen:
cost+=Math.sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb));

Projekt class(main) Part 2:
1.Individual objekte werden in ArrayList gespeichert.(List<Individual> population=new ArrayList<>()).

2.Mit for-Schleife werden new Individual Objekte erzeugt.
population.add(new Individual(makeRandomOrder(koord)));

3.Mit for-Schleife werden cost aller Individual Objekte berechnet.
x.setCost(calCost(x.getOrder(),koord));

4.Collections.sort(population).
for-Schleife: entfernen die Induvidual Objekte, die nicht als Eltern werden.

```
for(int j=populationNr-1,j<parentsNr-1;j--){
                    population.remove(j);
                }
```

5.for Schleife: das erst Element von population (best parent)wird durch crossover() Methode mit aller Elemente Kinder zeugen, Kinder werden im ArrayList gespeichert. Das zweite Element (Zweite best parent) wird rückwärts ein Kind mit best parent ein Kind zeugen.

wenn cost des Kind ist 0, Mutation wird beginnen.