```
Funktion generiereGruppenListeVonPaarListe():
    starterPopulation = generiereStarterPopulation()              // Erzeuge 100 zufällige Gruppenlisten
    sortiere(starterPopulation nach Fitness absteigend)           // Jede Gruppenliste hat eine Fitnessbewertung, höhere Fitness -> bessere Gruppenliste
    topZehnIndividuen = starterPopulation.subList(0, 10)          // Extrahiere die 10 besten Gruppenlisten
    currentPopulation = topZehnIndividuen                         // Aktuelle Generation, hier werden weitere Mutationen eingefügt

    für i von 0 bis 500:                                          // Wir führen insgesamt 500 Mutationen durch
        für jeden parent in topZehnIndividuen:                    // topZehnIndividuen umfasst immer die 10 besten Individuen der letzten Generation
            currentPopulation.addAll(mutateParent(parent))        // Für jeden Parent werden 10 neue Listen erstellt durch Mutationen und in die aktuelle Population eingefügt, siehe mutateParent funktion unten
        sortiere(currentPopulation nach Fitness)                  // Sortieren, damit die besten Listen vorne stehen
        topZehnIndividuen = currentPopulation.subList(0, 10)      // Extrahieren die 10 besten und setzen diese als neue topZehnIndividuen

        wenn Fitness über 10 Generationen nicht gestiegen und Fitness positiv:
            breche den Loop ab
            returne bestes Individuum(topZehnIndividuen[0])       // Bestes Individuum ist stets an der Stelle topZehnIndividuen[0]

    wenn nach 500 Generationen Fitness immer noch negativ:        // Negativ bedeutet, es wurde keine gültige Gruppenliste gefunden
        löse schlechtestes Cluster auf                            // Lösen wir das schlechteste Cluster auf
        füge alle Paare in diesem Cluster in Successorliste ein   // Fügen alle diese Paare in die Successorliste ein
        führe erneut 500 Generationen durch                       // Führen das selbe erneut durch, der Algorithmus findet nun wahrscheinlicher ein Ergebnis, da insgesamt weniger Gruppen gebildet werden müssen


Funktion mutateParent(parent):                                    // Die Methode erstellt von einem Parent 10 weitere Gruppenlisten durch Mutationen
    List newGeneratedLists                                        // Enthält unsere generierten Listen
    newGeneratedLists.addAll(parent.swapPairsBetweenClusters(5))  // Generiere 5 Listen durch Vertauschen der Paare zwischen Gruppenclustern
    newGeneratedLists.addAll(parent.swapPairsBetweenClustersAndSuccessorList(5))  // Generiere 5 Listen durch Vertauschen der Paare zwischen Cluster und Successorliste

    für jede liste in newGeneratedLists:
        wenn liste.KitchenIsUsedAtSameTime():                     // Prüfe, ob eine Küche zum selben Dish mehrfach benutzt wird
            liste.setFitness(-10000)                              // Liste schlecht bewerten, dass sie schnell rausgefiltert wird
        liste.bewerteFitness()                                    // Jede Liste wird bewertet, ungültige Listen erhalten -10000 Fitness

    returne newGeneratedLists                                     // Wir returnen die insgesamt 10 generierten Listen



Zusätzliche Informationen:
- Ein GruppenCluster besteht aus 9 Paaren, welche 9 Gruppen bilden. In jedem Cluster gibt es fest definierte Laufwege, woraus durch Fitnessbewertung der beste gewählt wird, all diese Laufwege erfüllen alle wichtige Kriterien
- In jeder Gruppenliste sind sowohl die Gruppencluster nach Fitness sortiert, als auch die Paare in dem Gruppencluster, dadurch können wir schwache Paare und schwache Cluster bevorzugt durch Mutationen verändern
- Hierzu wird eine CostumRandom Klasse verwendet die zufällig Zahlen generiert, aber mit  höherer Wahrscheinlichkeit kleinere Zahlen, dadurch, dass nicht nur kleinere Zahlen generiert werden, ist es dem Algorithmus möglich
  sich über mehrere Generationen aus lokalen Minima/Maxima zu befreien um noch bessere Ergebnisse zu erzielen
- Ist ein Cluster ungültig erhält es eine stark negative Fitnessbewertung und hat dadurch bei Mutationen immer Priorität
- Jegliche Fitnessbewertungen sind  abhängig von den gewählten Kriteren des Verwenders (AgeDifference, GenderDiversity...)
- Wenn eine Gruppenliste eine positive Fitnessbewertung hat, MUSS sie gültig sein und alle Kriterien erfüllen, es kann keine finale Gruppenliste mit negativer Fitness geben, da bei jeder unerlaubten Konstellation
  die Fitness so stark negativ bewertet wird, dass diese insgesamt Negativ sein muss
```