Pseudocodes

Hauptmethode der Pärchenbildung: `createPairs`	
Eingabe: `int foodPreference, int ageDifference, int genderDiversity`, in welchen einmal 1, einmal 2 und einmal 3 eingegeben sein sollten. 1 entspricht das wichtigste Kriterium. **Ausgabe:** fertig erstellte Pärchen im Event.	
1	Finde alle Teilnehmende, die noch in keinem Pärchen enthalten sind.
2	Teile diese Teilnehmende auf 3 Listen: die mit einer verfügbaren Küche, die mit einer Notfallküche und die ohne Küche.
3	Bilde Paare mithilfe der Methode `handlePairMatchingWithPreference` mit strenger Erfüllung vom wichtigsten Kriterium (`restrictedToPriorityOnePreference` auf true) nach folgenden Kombinationen und folgender Reihenfolge: i) die mit einer verfügbaren Küche & die ohne Küche ii) die mit einer verfügbaren Küche & die mit einer Notfallküche iii) die mit einer verfügbaren Küche & die mit einer verfügbaren Küche iv) die mit einer Notfallküche & die ohne Küche v) die mit einer Notfallküche & die mit einer Notfallküche
4	Wiederhole Schritt 3 mit lockerer Erfüllung vom wichtigsten Kriterium (`restrictedToPriorityOnePreference` auf false).

Hilfsmethode der Pärchenbildung: `handlePairMatchingWithPreference`	
Eingabe: `List<Participant> participantList1, List<Participant> participantList2,int foodPreference, int ageDifference, int genderDiversity, boolean restrictedToPriorityOnePreference, SpinfoodEvent event.` **Ausgabe:** fertig erstellte Pärchen von den gegebenen Listen.	
1	Entferne alle Teilnehmende, die aktuell schon in einem Pärchen enthalten sind.
2	Gruppiere die Teilnehmende nach dem wichtigsten Kriterium, falls `restrictedToPriorityOnePreference` true ist.
3	`for` each `group` from `participantList1`:
4	`for` each `participant` in `group`:
5	Überspringe, falls `participant` schon in einem Pärchen enthalten ist.
6	Nimm die Küche von `participant` als die für das Paar.
7	Finde alle Teilnehmenden von der entsprechender Gruppe von `participantList2`, die mit `participant` als ein Pärchen bilden können.
8	Entferne die Teilnehmenden, die schon in einem Pärchen sind.
9	Sortiere die gültigen Kandidaten nach der Wichtigkeit der Kriterien.
10	`if` ein gültiger Kandidat existiert:
11	Nimm die erste bzw. beste Kandidat als `partner`
12	`if partner` eine Küche hat, die näher zur Party ist:
13	Nimm die Küche von `partner` als die für das Paar
14	Erstelle ein neues Pair von `participant` und `partner`.

Hauptmethode der Gruppenbildung: `createGroups`	
Eingabe: `int foodPreference, int ageDifference, int genderDiversity, int pathLength, int numberOfElements`, in welchen einmal 1, einmal 2, einmal 3, einmal 4 und einmal 5 eingegeben sein sollten. 1 entspricht das wichtigste Kriterium. **Ausgabe:** fertig erstellte Gruppen im Event.	
1	Finde alle Pärchen, die noch keine Gruppe haben.
2	Entferne Pärchen, die WG-Küchen haben, die schon von anderen Pärchen für 3 Gänge im Event bereitgestellt sind.
3	`while` mehr als 9 Pärchen vorhanden sind:
4	Nimm das erste Pärchen für einen neuen `cluster`.
5	Finde mögliche Pärchen für diesen `cluster` mithilfe `findPairsForCluster`.
6	Initialisere eine `boolean successful` auf `true;`
7	`if (cluster.size() != 9):`
8	Setze `successful` auf `false;`
9	`else`:
10	Sortiere die Pärchen im `cluster` nach Abstand der Küche zur Party.
11	Bilde 9 entsprechenden Gruppen aus den Pärchen vom `cluster`.
12	`if` eine von diesen Gruppen keine Küche für den Gang vorhanden ist:
13	Setze `successful` auf `false;`
14	`if successful`:
15	Füge alle Gruppen im Event hinzu.

Hilfsmethode der Gruppenbildung: `findPairsForCluster`	
Eingabe: `List<Pair> validCandidates, List<Pair> cluster, List<Pair> availablePairs, List<Pair> eventPairs, int foodPreference, int ageDifference, int genderDiversity, int pathLength, int numberOfElements, Location partyLocation` **Ausgabe:** fertig erstellter Cluster.	
1	`while` (`cluster`.size() < 9) und (es gibt noch gültigen Kandidat):
2	Sortiere die gültigen Kandidaten nach Wichtigkeit der Kriterien.
3	Initialisiere `Iterator<Pair> iterator = validCandidates.iterator();`
4	`while(cluster.size() < 9 && iterator.hasNext())`
5	Pair `bestPair` = iterator.next();
6	Kontrolliere, ob `bestPair` zu allen Pärchen im `cluster` gültig ist.
7	`if` gültig:
8	Füge im `cluster` hinzu.