Erklärung des Paarbildungsalgorithmus
Der Paarbildungsalgorithmus ist darauf ausgelegt, Teilnehmer eines Koch-Events namens Spinfood in Paare zu unterteilen, die anschließend in Gruppen zusammenkommen, um verschiedene Gänge eines Menüs zuzubereiten und zu genießen. Die Paare können sich entweder zusammen anmelden oder einzeln, wobei letztere durch den Algorithmus zusammengeführt werden. Der Algorithmus zielt darauf ab, verschiedene Kriterien zu erfüllen, um eine möglichst hohe Zufriedenheit der Teilnehmer zu gewährleisten. Die Kriterien beinhalten Altersunterschiede, Geschlechterverhältnis, Essensvorlieben, Küchenverfügbarkeit und Wegstrecken zwischen den Orten.
Das Starten des Algorithmus findet in der Methode „getPairList“ statt, welcher man eine Liste von „Criterion“ übergibt. (Diese Liste wird vom Benutzer mit der Reihenfolge der Wichtigkeit der Kriterien übergeben).
Die Hauptmethode zum Starten des Algorithmus:

```
public static PairList getPairList(List<Criterion> criterion) {  3 usages   Parviz Moskalenko +1 *
    List<Pair> registeredTogetherPairs = getRegisteredTogetherPairs(); // get Pairs which registered together
    //create new Pairing constraints object, which handles relaxation of constraints in given order
    PairPairingConstraints constraints = new PairPairingConstraints(criterion);
    List<Pair> generatedPairs = getGeneratedPairs(constraints); //get generated Pairs
    registeredTogetherPairs.addAll(generatedPairs); //join Pairs which registered together and generated Pairs
    //Return a new PairList, which contains the Pairs which registered together + generated pairs and the successors which did not find partner
    return new PairList(registeredTogetherPairs, successors);
}
```

1. Initialisierung und Verarbeitung der Anmeldungen
Zunächst werden die Teilnehmer in zwei Listen aufgeteilt:

Paaranmeldungen: Teilnehmer, die sich bereits als Paar angemeldet haben.
Einzelanmeldungen: Teilnehmer, die sich alleine angemeldet haben und wahrscheinlich einen Partner zugewiesen bekommen.
Die Verarbeitung dieser Anmeldungen erfolgt durch die Methoden getRegisteredTogetherPairs und getRegisteredAloneParticipants, die entsprechende Listen zurückgeben.

2. Paarbildung für Einzelanmeldungen
Der Kern des Algorithmus für die Bildung neuer Paare aus Einzelanmeldungen findet in der Methode getGeneratedPairs statt. Hier werden die Einzelanmeldungen paarweise kombiniert, basierend auf einer Reihe von Kriterien, die in einer Liste von Criterion Objekten übergeben werden.

Initialisierung:

Eine Liste der Einzelanmeldungen wird erstellt.
Ein Boolean-Array ***used*** wird verwendet, um zu verfolgen, welche Teilnehmer bereits in einem Paar sind.
Eine Liste ***successors*** wird erstellt, um die Teilnehmer zu speichern, die keinen Partner gefunden haben.
Paarbildung:

Es wird durch die Liste der Teilnehmer mit zwei for-Schleifen iteriert, um Paare zu bilden. Dabei werden die Kriterien überprüft, die in ***PairPairingConstraints*** definiert sind.
Falls ein Paar gültig ist und die Kriterien erfüllt sind, wird es zur Liste der gebildeten Paare hinzugefügt und die entsprechenden Teilnehmer als verwendet markiert. Nachdem einmal über alle Teilnehmer iteriert wurde, werden die Teilnehmer, die keinen Partner gefunden haben, zur List der ***successors*** hinzugefügt und es wird geprüft, ob mehr als 1 Teilnehmer in der List der ***successors*** ist. Sollte dies der Fall sein, wird das Verhältnis der Nachrücker zu den Teilnehmern berechnet. (***successorsRate***) Je nach dem, auf welchem Platz das Kriterium 10, also die Minimierung der Anzahl, sodass ein Teilnehmer keinen Partner findet, in der Liste der Criterions steht, wird eine bestimmte Rate akzeptiert. Also z.B.: Wichtigstes Kriterium → erlaube nur eine Rate von 5%. Für jeden niedrigeren Platz erlaube 5% mehr. Sollte nun am Ende der Iteration die Rate akzeptabel sein,

wird die Paarbildung abgebrochen und die gejointen Pairs werden zurückgegeben. Wenn die Rate nicht akzeptabel ist, werden die soft-Constraints ein wenig gelockert und der ganze Prozess beginnt von vorne. (Dies wird so oft wiederholt, bis die soft-Constraints völlig gelockert sind)
Wenn alle soft-Constraints gelockert wurden, wird noch ein letztes mal geprüft, ob überhaupt Paare entstanden sind. Sollte dies nicht der Fall sein, wird noch einmal eine Paarbildung mit zwei for-Schleifen durchgeführt, wobei nur die „harten" Kriterien beachtet werden. (Dies wird nur gemacht, wenn noch keine Paare gefunden wurden, da man ja eigentlich nicht möchte, dass die Paare nur nach den harten Kriterien gebildet werden, aber im Notfall soll dieser Zustand behandelt werden).
Am Ende werden die ***joinedPairs*** zurückgegeben.

Relaxation der Kriterien:
Wird von der Klasse ***PairPairingConstraints*** gehandlet, wobei die übergebene Liste der „Criterions" beachtet wird. Jedes mal wenn ***relaxConstraints()*** aufgerufen wird, wird das nächste Kriterium was ansteht gelockert. (Z.B.: relaxConstraints() wird aufgerufen und das nächste Constraint was ansteht ist Gender Diversity. Dann wird relaxedGenderDiversity = true gestellt, und in der isValid() Methode, wird nichtmehr drauf geachtet, ob das Geschlecht unterschiedlich ist.

4. Ergebnis und Nachrückerliste
Die Liste der gebildeten Paare und die Liste der Teilnehmer, die keinen Partner gefunden haben (Nachrückerliste), werden in einem PairList Objekt zurückgegeben.

Wie gut erfüllt der Algorithmus die Projektanforderungen:

1: Bei der Zusammenstellung eines Pärchens, wird diesem eine Hauptessensvorliebe zugewiesen. Diese ergibt sich wie folgt: Sind die Essensvorlieben der beiden Teilnehmer gleich, ist diese auch die Hauptessensvorliebe. Bei einem Fleischliebhaber (im folgenden liebevoll Fleischi genannt) und Egali ergibt sich als Hauptvorliebe Fleischi. Ist ein Fleischi/Egali mit einem Veggie/Veganer in einem Pärchen, ist der Fleischi/Egali unterlegen. Ein Veganer und ein Veggie haben als Hauptessenvorliebe Veganer. Für die Wahl des Gerichts einer Gruppe ist die Hauptessensvorliebe der Gruppe entscheidend. Diese ergibt sich ebenfalls entsprechend der obigen Bedingungen aus den Hauptessensvorlieben der Pärchen der Gruppe. Sollten alle Pärchen Egalis sein, ergibt sich als Vorliebe der Gruppe Fleischi. Damit sich nicht zuviele Fleischis/Egalis anpassen müssen, darf zudem in gemischten Gruppen höchstens ein Fleischi-/Egali-Pärchen vorhanden sein. Die Veggie- /Veganer-Pärchen müssen also die Mehrheit bilden.

→ Es wird darauf geachtet, dass die Paare am besten keine Distanz von der Essensvorliebe haben, aber nach mehrmaligem lockern und je nach dem, an welcher Stelle das Kriterium steht, wird am Ende zusammengestellt, auch wenn die Distanz der Essensvorliebe etwas größer ist. Restliche Anforderungen → Gruppenalgorithmus

2. Es ist darauf zu achten, dass ein Pärchen in allen drei Gängen mit unterschiedlichen Pärchen speist. D.h., im Laufe des Abends soll ein Pärchen mit sechs anderen Pärchen speisen, in jedem Gang mit zwei anderen Pärchen. Zudem muss jedes Pärchen genau einmal am Abend kochen.

→ Gruppenalgorithmus

3. Jedes Kochpaar muss über mindestens eine Küche verfügen. Da Küchen immer knapp sind, sollte es vermieden werden, Pärchen zu bilden, in denen beide Personen eine Küche stellen könnten. Das wäre eine Verschwendung an Küchen, denn je Paar wird nur eine Küche beansprucht. Pärchen ohne Küchen sind ungültig, da sie nicht kochen und keine Gäste empfangen können. Wenn zwei Küchen vorhanden sind und es sich um den Gang der Nachspeise handelt, wird in der Küche gekocht, die näher an der After-Dinner-Party Location ist.

→ Es wird versucht die Küchenanzahl bei Paaren bei 1 zu halten, wenn das Constraint gelockert wird, wird aber auch >1 zugelasssen. In jedem Fall hat aber ein Paar immer mindestens eine Küche.

4. Manchmal melden sich bei Spinfood sehr große WGs an. Bei mehr als 3 Anmeldungen aus einer WG, die jeweils eine Küche stellen, ist eine Überbelegung der Küche nicht zu vermeiden. Gibt es z.B. 4 Pärchenanmeldungen aus einer WG, in denen jeweils die Küche als nutzbar angegeben wird, wird entweder die Vor, Haupt- oder Nachspeise doppelt belegt sein. Dann wird es zu eng in den Küchen. Eine Küche darf letztlich nie von mehr als 3 Pärchen genutzt werden.

→ Gruppenalgorithmus

5. Zwei Teilnehmende können nur dann ein Pärchen bilden, wenn die Distanz zwischen ihnen nicht 0 ist, d.h. wenn sie nicht in demselben Haus wohnen.

→ Dies wird immer eingehalten durch den check von isValid() von Pair, welches false zurückgibt, sollten sie im selben Haus wohnen.

6. Vorzugsweise sollen Pärchen und Gruppen mit identischen Essensvorlieben erstellt werden. Geht dies nicht auf, sollten die Vorlieben zumindest möglichst ähnlich sein. Die Vorliebenabweichung der Pärchen und Gruppen soll also möglichst gering sein. Besonders hilfreich ist hier die Personengruppe der Egalis. Sie können als Lückenfüller verwendet werden und sollten bei der Pärchenbildung vorzugsweise mit einem Fleischi kombiniert werden. Auch bei der Gruppenbildung sollten Egali-Pärchen vorzugsweise mit Fleischi-Pärchen in eine Gruppe. Ansonsten kommt es zu einer erhöhten Anzahl an Abmeldungen kurz nach der Bekanntgabe der Zusammenstellung der Pärchen und Gruppen.

→ Es wird versucht, die Distanz der Essensvorliebe von Pärchen nicht größer als 1 werden zu lassen, bis das Constraint gelockert wurde (je nach dem welche Wichtigkeit, wird dies dann früher oder später gelockert), Rest Gruppenalgorithmus.

7. Die Altersdifferenz bei der Pärchenbildung sollte minimiert werden. So wollen wir z.B. verhindern, dass junge Frauen mit deutlich älteren Männern kombiniert werden.

→ Es wird zuerst versucht mit einer Alterdifferenz(Range, nicht absolut) von 0 zu pairen. Sollte dies nicht zufriedenstellend sein, wird die eine range von 1 zugelassen und so weiter, bis die range maximal ist und jedes Alter mit jedem Alter zusammen gepaart wird.

8. Die Geschlechter sollen möglichst durchmischt werden. Das heißt, vorzugsweise werden Pärchen mit unterschiedlichen Geschlechtern gebildet. Auch in den Gruppen soll die Geschlechterdiversität hoch sein.

→ Wird versucht einzuhalten, bis das Kriterium gelockert wird, werden erstmal nur Paare erlaubt, die ein unterschiedliches Geschlecht haben.

9. Die Weglänge eines Pärchens soll möglichst klein sein, damit die Wegzeiten zwischen den Gängen nicht so groß sind. Falls möglich, sollte die Laufrichtung von Gang zu Gang in Richtung After-Dinner Party gehen. Das heißt, wenn die AfterDinner-Party in der Innenstadt stattfinden sollte, dann sollten die Vorspeisen weiter entfernt vom Stadtkern, die Hauptgerichte etwas näher und die Nachspeisen tendenziell in der Näher des Stadtkerns stattfinden. So nähern sich die Teilnehmenden von Gang zu Gang immer mehr der After-Dinner-Party.

→ Gruppenalgorithmus

10. Natürlich kann es sein, dass es unter den gegebenen Voraussetzungen, nicht möglich ist, jedem Teilnehmenden eine/n KochpartnerIn zuzulosen. Die Anzahl der Personen, die kein/e PartnerIn bekommen, soll aber minimiert werden. Je mehr Zuordnungen zustande kommen, desto besser. Es ist zu erwarten, dass einige Teilnehmende keine/n KochpartnerIn erhalten werden. Diese Personen werden zu Nachrückenden und zugeordnet, sobald sich andere Teilnehmende vom Event abmelden

→ Wird mit dem Verhältnis gehandelt, von Nachrückern zur Anzahl der Participants. Dies wird je nach Stellung des Kriteriums versucht gering zu halten.

Argumentation der Algorithmuswahl:
Der iterative Algorithmus der Paarbildung wurde gewählt, da er flexibel genug ist, um eine Vielzahl von Kriterien zu berücksichtigen und anzupassen. Durch die iterative Natur können die Kriterien in mehreren Durchläufen gelockert werden, um sicherzustellen, dass so viele Paare wie möglich gebildet werden, während gleichzeitig die wichtigsten Anforderungen erfüllt bleiben. Dies ermöglicht eine hohe Zufriedenheit der Teilnehmer und eine effiziente Nutzung der verfügbaren Ressourcen, wie Küchen. Der Algorithmus hat eine Laufzeit von O(n^2), was bei der gegebenen Teilnehmeranzahl relativ schnell berechnet wird.