1 Couple-Manager

```
void givePeopleWithoutPartner(List singles, strictnessLevel, Location partyLoc)
calcCouples(strictnessLevel)


void calcCouples(Strictness strictnessLevel) {
switch (strictnessLevel)
case A:
matchSingles(createMatrix(vegetarians), vegetarians)
matchSingles(createMatrix(mixedGroup), mixedGroup)
break
case B:
//with and without kitchen
break
case C:
//maximum number off matches

}

createMatrix(List singles) {

for (int i = 0  i < size  i++) :
for (int j = 0  j < size  j++) :
matrix[i][j] = new NumberBox(calculateCost(singles.get(i), singles.get(j)))

return matrix
}

calculateCost(Person p1, Person p2) {

if (!p1.hasSameFoodPreference(p2)) {
    cost += calculateFoodPreferenceCost(p1, p2)

cost += calculateKitchenDistanceCost(p1, p2)
cost += calculateAgeDifferenceCost(p1, p2)
cost += calculateGenderCost(p1, p2)

return cost
}

matchSingles( matrix, List people) {
adjustMatrix(matrix)
formCouples(matrix, people)
}
```

```
matchSinglesWithAdjustment(matrix, List group1, List group2) {
adjustMatrix(matrix)
matrix = splitMatrix(matrix)
formCouples(findZeroCoordinates(matrix), group1, group2)
}

// Helper Methods
private List<Person> filterByKitchenAvailability(List<Person> participants, boolean hasK
return participants.stream().filter(p -> p.hasKitchen() == hasKitchen).collect(Collector
}

private void formCouples(matrix, List singles) {
    for (int i = 0  i < numberBox.length  i++) :
        for (int j = 0  j < numberBox[i].length  j++) :
            if (numberBox[i][j].getNumber() == 0.0)
                Couple couple = new Couple(people.get(i), people.get(j))
    }

    for (int i = 0  i < numberBox.length  i++) {
        if (people.get(i).getPartner() == null)
            stillSingle.add(people.get(i))
    }
}
```

2 Group-Manager

```
//preprocess input
input = filterUnusableKitchen(allCouples)
while !(input|9)
    removePersonFrom(input)
rank = x,y->(x.attibute0-y.attribute0)*attribute0Weight
        +...
        + (x.attibuten-y.attributen)*attributenWeight
//match 9 couples
sortByLocation(input)
for (input.size / 9)
    cluster = Couple[9]
    foreach course
        //fill each course-layer in cluster,
        // a,b,c are the hosts of each layer
         a = minRankComparedToCluster(input)
         cluster.add(a)
         b = minRankComparedToCluster(input)
         cluster.add(b)
         c = minRankComparedToCluster(input)
         cluster.add(c)
    // create 9 groups, in accordance to the cluster
    output.add(createGroupCluster(cluster))

return output
```