readCSVTest

read\_FileTest

*Precondition :*

A .csv file that is containing personal data of 237 Participants is available

*Precess:*

This csv file is read by read\_File method

*Expected Behavior:*

This file is read succesfully and the data are saved as list of string array. The length of the list is 238, because the Header of the .csv data is also saved as a String array.

*Actual Behavior:*

addParticipantTest

*Precondition :*

A csv file is read successfully and this data is saved as List of string array. The List contains 237 personal data of 237

Participants

*Precess:*  
User calls function "addParticipant" with this list as input.

*Expected Behavior:*  
A participant List that contains 310 Participants, a pair List that contains 73 Pairs is created. 164 Participants that is not in any pair are recorded. There is no duplicate in participant List as the total number of Participants is equal to alone Participant and 2 \* number of Pairs

*Actual Behavior:*

addPartyLocationTest

*Precondition :*

PartyLongitude and PartyLatitude in class „Event“ are empty. And Data of PartyLocation is read succesfully.

*Precess:*  
This function is called by constructor in readCSV and this method is populated with the read data as a list of string arrays.

First Array contains attributes(Longitude, Latitude). And second array has (5.0,6.0)

*Expected Behavior:*  
The entered Latitude and longitude are converted to double variables and stored in the event class partylocation and partylatitude. After that, it is possible to refer to these numbers from the event class.(5.0 as Longitude,.6.0 as Latitude)

*Actual Behavior:*

ParticipantTest

assignAgeRangeTest

*Precondition :*

Object "Participant" has been completely created. (All instances of this object are satisfied)

*Precess:*

This method is called in the constructor when the participant's object is created.

*Expected Behavior:*

AGERANGE is classified into the following nine categories:  
　　  age                               AGE\_RANGE(int)  
  0<= age <= 17                          LessThan18 (0)  
18<= age <= 23                          LessThan24 (1)  
24<= age <= 27                          LessThan28(2)  
28<= age <= 30                          LessThan31(3)  
31<= age <= 35                          LessThan36(4)  
36<= age <= 41                          LessThan41(5)  
42<= age <= 46                          LessThan47(6)  
47<= age <= 56                          LessThan57(7)  
57<= age                                    MoreThan57(8)  
The participants is given the above enumlations according to their age, which can also be obtained as integer values

All inputss are minimum and maximum values in their respective definition ranges above.

*Actual Behavior:*

assignFoodPreferenceTest

*Precondition :*

Object "Participant" has been completely created. (All instances of this object are satisfied)

*Precess:*

foodPreference of this object is confirmed by this method with enumulation "FOOD\_PREFERENCE"

*Expected Behavior:*

AGERANGE is classified into the following nine categories:  
　　  Eingabe(String:foodPreference)                               FOOD\_PREFERENCE(Enum)  
                “meat“ meat

„none“ none

„veggie“ veggie

„vegan“ vegan

After using this method, foodPreference of the participant is saved as FOOD\_PREFERENCE(enum) instead of String.  
*Actual Behavior:*

PairTest

calculateAgeDiferrence

*Precondition :*

Object "Pair" has been completely created. (All instances of this object are satisfied). And each participant contains enumlation AGE\_RANGE as age.(This Pair contains P1:age20(RANGE 1),P5:age15(RANGE 0))

*Precess:*

This methos calculates the agediferrence of Pair. After that the result is returned in data typ int

*Expected Behavior:*

Two Particpant of Pair contain integer Value(0-8) of AGE\_RANGE. And the diffrence of these values is calculated. This result is absolut value. The calculation results of this test is 1 (|0-1|).

*Actual Behavior:*

calculateSexDiversity

*Precondition :*

Object "Pair" has been completely created. (All instances of this object are satisfied). And each participant contains enumlation SEX as gender.

*Precess:*

This method is called by Class ListManegement to buid the group. And it calculates the gender diversity of Pair. After that the result is returned in data typ double.

*Expected Behavior:*

If the gender of the pair is female and other gender, 0 is return value in double. Otherwise, the return value is 0.5 in double.

*Actual Behavior:*

calculateDistanceBetweenKitchens

*Precondition :*

Object "Pair" has been completely created. (All instances of this object are satisfied). And at least one participant has a kitchen available for use

*Precess:*

This method is called by Class ListManegement to buid the group. And it calculates the distance betweenKitchens of Pair. After that the result is returned in data typ double.

*Expected Behavior:*

If the pair owns only one available kitchen, the return value is 0. If both of the pair own a kitchen, the distance between the kitchens is calculated and this is output as the return value.

*Actual Behavior:*

calculateDistanceBetweenKitchens

*Precondition :*

Object "Pair" has been completely created. (All instances of this object are satisfied). And And each participant contains enumlation food\_preference.

*Precess:*

This method is called by Class ListManegement to buid the group. And it calculates the score of foodpreferences. After that the result is returned in data typ int.

*Expected Behavior:*

Calculate the difference in absolute values of the integers that the foodpreference has. The result is output.

*Actual Behavior:*

**GroupTest**

**calculateFoodPreference()**

*Precondition:*

The **Group** object **g1** is created with specific **Pair** objects.

*Process:*

The **calculateFoodMatchScore** method is called on **g1**.

*Expected Behavior:*

The calculated food match score should be equal to 0.666.

*Actual Behavior:*

The calculated food match score matches the expected value.

**calculateSexDiversity()**

*Precondition:*

The **Group** object **g1** is created with specific **Pair** objects.

*Process:*

The **calculateSexDiversity** method is called on **g1**.

*Expected Behavior:*

The calculated sex diversity should be equal to 0.5.

*Actual Behavior:*

The calculated sex diversity matches the expected value.

**calculateSexDiversity()**

*Precondition*:

A new **Group** object **g2** is created with different **Pair** objects, including one pair with a female participant.

*Process*:

The **calculateSexDiversity** method is called on **g2**.

*Expected Behavior*:

The calculated sex diversity should be equal to 1.0.

*Actual Behavior:*

The calculated sex diversity matches the expected value.

**calculateSexDiversity()**

*Precondition*:

*Process*:

*Expected Behavior*:

*Actual Behavior*

**calculateSexDiversity()**

*Precondition*:

A new **Group** object **g3** is created with different **Pair** objects, including two pairs with female participants.

*Process*:

The **calculateSexDiversity** method is called on **g3**.

*Expected Behavior*:

The calculated sex diversity should be equal to 0.5.

*Actual Behavior*

The calculated sex diversity matches the expected value.

**calculateSexDiversity()**

*Precondition*:

A new **Group** object **g4** is created with different **Pair** objects, none of which have female participants.

*Process*:

The **calculateSexDiversity** method is called on **g4**.

*Expected Behavior*:

The calculated sex diversity should be equal to 1.5.

*Actual Behavior*

The calculated sex diversity matches the expected value.

**calculateDistanceBetweenKitchens()**

*Precondition:*

The **Group** object **g1** is created with specific **Pair** objects.

*Process:*

The **calculateDistanceBetweenKitchens** method is called on each **Pair** within **g1**, and their values are summed.

*Expected Behavior:*

The calculated total kitchen distance should be equal to 437.

*Actual Behavior*:

The calculated total kitchen distance matches the expected value.

**calculatePairAgeDifference()**

*Precondition:*

The **Group** object **g1** is created with specific **Pair** objects.

*Process:*

The **calculatePairAgeDifference** method is called on **g1**.

*Expected Behavior:*

The calculated age difference between pairs should be equal to 2.

*Actual Behavior:*

The calculated age difference between pairs matches the expected value.

ListManagementTest

makeBestPairTest\_noKitchen

*Precondition :*

Two Participants, whoes kitchens are not available,are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement with input either of partcipants

*Expected Behavior:*

Output is null.

*Actual Behavior:*

Output is null.

makeBestPairTest\_MeatWithVeggie

*Precondition :*

Two Participants(These food preference are meat and veggie) are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement with input either of partcipants

*Expected Behavior:*

Output is null.

*Actual Behavior:*

Output is null.

makeBestPairTest\_VeganWithVeggie

*Precondition :*

Two Participants(These food preference are vegan(p6) and veggie(p3)) are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement with input participant of veggie(p3)

*Expected Behavior:*

A pair of p3 and p6 is created and the ID of p3, "p3", is returned as an output

.

*Actual Behavior:*

Output is „p3“ as ID.

makeBestPairTest\_checkWithFoodPreference

*Precondition :*

4 Participants(These food preference are meat(p1) ,veggie(p3),none(p4),vegan(p5)) are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement with input participant of meat (p1)

*Expected Behavior:*

A pair of p1 and p4 is created and the ID of p4, "p4", is returned as an output

.

*Actual Behavior:*

Output is „p4“ as ID.

makeBestPairTest\_checkWithAgeDifference

*Precondition :*

3 Participants(p1 : foodPreference(meat) , age(20) , p4 : foodPreference(none),age(15), p7:foodPreference(none),age(85)) are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement with input participant of meat (p1)

*Expected Behavior:*

A pair of p1 and p4 is created and the ID of p4, "p4", is returned as an output

.

*Actual Behavior:*

Output is „p4“ as ID.

makeBestPairListTest\_unpossible

*Precondition :*

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement.

*Expected Behavior:*

The size of pairList is 0 after call the function.

*Actual Behavior:*

The size of pairList is 0

makeBestPairListTest

*Precondition :*

4 Participants(p1 : foodPreference(meat) , age(20), Gender(male), p3 : foodPreference(veggie),age(60),Gender(female) p4:foodPreference(none),age(15),gender(other),p6:foodPreference(veggie),age(15),gender(male)) are in unmatchedParticipantsList of Class“DataList“.

pairList in Class „DataList“ contains nothing. And listManagement contains this DataList.

*Precess:*

This method is called on listManagement.

*Expected Behavior:*

A pair of p4 and p6 is created.

*Actual Behavior:*

A pair of p4 and p6 is created.

makeBestGroupValidPairLIst()

*Precondition*:

Six Pairs that are made from Twelve Participants are introduced. 3 Pairs consist of meat-eater and another 3 pairs consist of Vegans or Participants that prefer Veggie. The Group List for Course 01 in DataList is empty and all Pairs are added into the Pair List in Data List.

*Process*:

This method is called on List Management

*Expected Behavior*:

The Group List for Course 1 in Data List contains 2 valid Group.

*Actual Behavior:*

makeBestGroupInvalidPairLIst()

*Precondition*:

Three Pairs that are made from Six Participants are introduced.1 Pair consists of meat-eater and another 2 pairs consist of Vegans or Participants that prefer Veggie. All Pairs are added into the Pair List in Data List.

*Process*:

This method is called on List Management

*Expected Behavior*:

A Group is formed that consists of 3 Pairs that have Participants that prefer meat.

*Actual Behavior:*

makeGroupForVeganOrVeggiePair()

*Precondition*:

Four Pairs that are made from Eight Participants are added into tempPairList in List Management. 1 Pair consists of meat-eater and another 3 pair consist of Vegans or Participants that prefer Veggie.

*Process*:

This method is called on List Management

*Expected Behavior*:

A Group is formed that consists of 3 Pairs that have Participants that prefer vegan or veggie.

*Actual Behavior:*

makeGroupForMeatPair()

*Precondition*:

Four Pairs that are made from Eight Participants are added into tempPairList in List Management. 3 Pairs consist of meat-eater and another 1 pair consists of Vegans or Participants that prefer Veggie.

*Process*:

This method is called on List Management

*Expected Behavior*:

A Group is formed that consists of 3 Pairs that have Participants that prefer meat

*Actual Behavior:*

makeBestGroupNotEnoughPair()

*Precondition*:

Two Pairs that are made from Four Participants are added into tempPairList in List Management. Both consist of meat-eater

*Process*:

This method is called on List Management

*Expected Behavior*:

The method return Null, as there is not enough Pair to form a valid Group.

*Actual Behavior:*