Unit Test Documentation

Meilenstein 2

Überblick

Dieses Dokument stellt eine detaillierte Übersicht über die durchgeführten Unit-Tests für die PairController-Klasse dar. Die Unit-Tests wurden entwickelt, um sicherzustellen, dass die Funktionalität der wichtigsten Methoden innerhalb der Klasse ordnungsgemäß implementiert ist und korrekt arbeitet. Dabei liegt ein besonderer Fokus auf der Verarbeitung von Teilnehmerdaten aus CSV-Dateien.

Die Testfälle sind strukturiert und dokumentiert, um einen klaren Einblick in das Testvorgehen, die erwarteten Ergebnisse und etwaige Fehler zu bieten. Jeder Testfall wird mit dem Namen der getesteten Methode, einer Beschreibung der durchgeführten Tests, eventuell aufgetretenen Fehlern und der Angabe, ob eine vollständige Anweisungsabdeckung erreicht wurde, präsentiert.

Das Ziel dieses Dokuments ist es, die Qualität und Zuverlässigkeit der Registration-Klasse sicherzustellen und eine solide Grundlage für die Entwicklung und Wartung des Systems zu schaffen.

public class PairControllerTest

Methode	# Tests	# Fehler	Voll. Abd.
testPrintPairs()	1	0	Ja
testFoodIncompatibility()	1	0	Ja
testFoodCompatibility()	1	0	Ja
testPairsWithHighKitchenOccupation()	1	0	Ja
testPairDistanceIsNotZero()	1	0	Ja
testMeatEatersWithAnyFoodEater()	1	0	Ja
testVeggieAndVeganerFoodPreference()	1	0	Ja

Testfall 1: testFoodIncompatibility

Vorbedingung: Zwei Teilnehmer mit inkompatiblen Essenspräferenzen (MEAT und VEGAN) sind in der Teilnehmerliste.
Ablauf: Erstelle eine Liste mit den Teilnehmern participantFleischi und participantVegan. 1. Initialisiere den PairGeneratorService mit dieser Liste und der Partylocation. 2. Rufe die generatePairs-Methode des PairGeneratorService auf. 3. Überprüfe, ob die generierten Paare leer sind.
Erwartetes Verhalten: Es werden keine Paare generiert.

Tatsächliches Verhalten: Die Liste der generierten Paare ist leer. Der Test verläuft erfolgreich.

Testfall 2: testFoodCompatibility

Vorbedingung: Eine vollständige Teilnehmerliste mit verschiedenen Essenspräferenzen ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob alle generierten Paare kompatible Essenspräferenzen haben.
Erwartetes Verhalten: Alle Paare haben kompatible Essenspräferenzen.
Tatsächliches Verhalten: Alle generierten Paare haben kompatible Essenspräferenzen. Der Test verläuft erfolgreich.

Testfall 3: testFoodCompatibility

Vorbedingung: Eine vollständige Teilnehmerliste mit verschiedenen Essenspräferenzen ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob alle Paare mit mindestens einem Fleischesser kompatible Essenspräferenzen haben.
Erwartetes Verhalten: Fleischesser sind nur mit Fleischessern oder Teilnehmern ohne spezielle Essenspräferenzen gepaart.
Tatsächliches Verhalten:

Alle Fleischesser sind korrekt gepaart. Der Test verläuft erfolgreich.

Testfall 4: testVeggieAndVeganerFoodPreference

Vorbedingung: Eine vollständige Teilnehmerliste mit verschiedenen Essenspräferenzen ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob alle Paare mit VEGAN und VEGGIE die Hauptessenspräferenz VEGAN haben.
Erwartetes Verhalten: Paare mit VEGAN und VEGGIE haben die Hauptessenspräferenz VEGAN.
Tatsächliches Verhalten: Die Hauptessenspräferenz ist korrekt gesetzt. Der Test verläuft erfolgreich.

Testfall 5: testPairsWithHighKitchenOccupation

Vorbedingung: Eine vollständige Teilnehmerliste ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob keine Küche von mehr als 3 Paaren genutzt wird.
Erwartetes Verhalten: Keine Küche wird von mehr als 3 Paaren genutzt.
Tatsächliches Verhalten: Die Küchenverteilung ist korrekt. Der Test verläuft erfolgreich

Testfall 6: testKitchenAvailabilty

Vorbedingung: Eine vollständige Teilnehmerliste ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob alle Paare gültige Küchenkoordinaten haben.
Erwartetes Verhalten: Alle Paare haben gültige Küchenkoordinaten
Tatsächliches Verhalten: Alle Paare haben gültige Küchenkoordinaten. Der Test verläuft erfolgreich.

Testfall 7: testPairDistanceIsNotZero

Vorbedingung: Eine vollständige Teilnehmerliste ist registriert..
Ablauf: Initialisiere den PairGeneratorService mit der vollständigen Teilnehmerliste und der Partylocation. Rufe die generatePairs-Methode des PairGeneratorService auf. Überprüfe, ob keine Paare mit identischen Küchenkoordinaten existieren.
Erwartetes Verhalten: Keine Paare haben unterschiedliche Küchenkoordinaten.
Tatsächliches Verhalten: Alle Paare haben unterschiedliche Küchenkoordinaten. Der Test verläuft erfolgreich.

```
public class PairsValidatorServiceTest
```

Methode	# Tests	# Fehler	Voll. Abd.
Validate()	1	0	Ja

Testfall 1: validate() - Überprüfung gültiger Paare

Vorbedingung: Zwei Teilnehmer mit gleichen Essenspräferenzen (MEAT) und verfügbaren Küchen sind in einem Paar (validPair).
Ablauf: Initialisiere den PairsValidatorService mit einer Liste, die das gültige Paar (validPair) enthält. Rufe die validate-Methode des PairsValidatorService auf.
Erwartetes Verhalten: Die generierten Gruppen sind nicht null und nicht leer.
Tatsächliches Verhalten: Korrektes Verhlaten

```
public class GroupGeneratorServiceTest
```

Methode	# Tests	# Fehler	Voll. Abd.
testGenerateGroups()	1	0	Ja
testGenerateGroupsCreatesCorrectGroups()	1	0	Ja
testToCSV()	6	0	Ja
testUniquniessOfPairs()	1	0	Ja
testInvalidFoodPreferences()	1	0	Ja
testInvalidFoodPreferences2()	1	0	Ja
testGroupPreferenceOfAnyFoodEaters()	1	0	Ja

Testfall 1: testGenerateGroups

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe, ob die generierten Gruppen nicht null und nicht leer sind.
Erwartetes Verhalten: Die generierten Gruppen sind nicht null und nicht leer.
Tatsächliches Verhalten: Keine Ausnahme wird geworfen. Der Test verläuft erfolgreich.

Testfall 2: testGenerateGroupsCreatesCorrectGroups

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. **Erwartetes Verhalten:** Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe, ob die generierten Gruppen nicht null und nicht leer sind. Überprüfe, ob die Anzahl der Gruppen für jeden Kurs (Vorspeise, Hauptgericht, Dessert) gleich ist. Überprüfe, ob jede Gruppe genau 3 Paare enthält.
Tatsächliches Verhalten: Die Anzahl der Gruppen für jeden Kurs ist gleich. Jede Gruppe enthält genau 3 Paare.

Testfall 3: testUniquniessOfPairs

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe für jedes Paar, ob es in den drei verschiedenen Kursen (Vorspeise, Hauptgericht, Dessert) jeweils verschiedene Paare treffen
Erwartetes Verhalten: Jedes Paar trifft 6 anderen Paare
Tatsächliches Verhalten:

Korrektes Verhlaten

Testfall 4: testToCSV

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Rufe die Methode toCSV mit einem Ausgabepfad auf, um die Gruppendaten in eine CSV-Datei zu schreiben. Überprüfe, ob die Datei existiert und nicht leer ist.
Erwartetes Verhalten: Die CSV-Datei existiert und ist nicht leer.
Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 5: testToCSVThrowsException

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Rufe die Methode toCSV mit einem ungültigen Dateipfad auf und überprüfe, ob eine Ausnahme ausgelöst wird.
Erwartetes Verhalten: Eine Ausnahme wird ausgelöst.

Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 6: testToCSVforColumns

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Rufe die Methode toCSV mit einem Ausgabepfad auf, um die Gruppendaten in eine CSV-Datei zu schreiben. Überprüfe, ob die erste Zeile der CSV-Datei die erwarteten Spaltennamen enthält.
Erwartetes Verhalten: Die erste Zeile der CSV-Datei enthält die erwarteten Spaltennamen.
Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 7: testToCSVWritesDataCorrectly

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Rufe die Methode toCSV mit einem Ausgabepfad auf, um die Gruppendaten in eine CSV-Datei zu schreiben. Überprüfe, ob jede Zeile der CSV-Datei 12 Datenfelder enthält.

Erwartetes Verhalten: Jede Zeile der CSV-Datei enthält 12 Datenfelder.
Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 8: testInvalidFoodPreferences

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe, ob Gruppen mit der Essenspräferenz "MEAT" keine veganen oder vegetarischen Paare enthalten.
Erwartetes Verhalten: Gruppen mit der Essenspräferenz "MEAT" enthalten keine veganen oder vegetarischen Paare.
Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 9: testInvalidFoodPreferences2

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe, ob vegane oder vegetarische Gruppen keine Fleischesser enthalten.
Erwartetes Verhalten: Vegane oder vegetarische Gruppen enthalten keine Fleischesser.

Tatsächliches Verhalten: Korrektes Verhlaten

Testfall 10: testGroupPreferenceOfAnyFoodEaters

Vorbedingung: Eine Liste von Paaren (pairs) ist initialisiert. Eine Partylocation (partyLocation) ist initialisiert.
Ablauf: Initialisiere den GroupGeneratorService mit den Paaren und der Partylocation. Rufe die Methode generateGroups auf, um Gruppen zu erzeugen. Überprüfe, ob Gruppen von Teilnehmern ohne spezielle Essenspräferenzen als "MEAT" klassifiziert werden.
Erwartetes Verhalten: Vegane oder vegetarische Gruppen enthalten keine Fleischesser.
Tatsächliches Verhalten: Korrektes Verhlaten