Отчет о тестировании с помощью Junit

Тестирование метода getAllUsers():

В этом тесте сравнивается ожидаемый список пользователей с текущим и если в списках есть разница, то тест завершится ошибкой.

@Test  
public void getAllUsers() {  
 List<User> *expected* = User.*getAllUsers*();  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
 *actual*.add(user1);  
 *actual*.add(user2);  
  
 *assertEquals*(*expected*, *actual*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода testGetAllUsers():

В этом тесте сравнивается ожидаемый список пользователей, с определенным полом, с текущим списком пользователей и если в списках есть разница, то тест завершится ошибкой.

@Test  
public void testGetAllUsers() {  
 List<User> *expected* = User.*getAllUsers*(Sex.*FEMALE*);  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user1);  
 *actual*.add(user2);  
  
 *assertEquals*(*expected*, *actual*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода getHowManyUsers():

В этом тесте сравнивается ожидаемый размер списка пользователей с текущим и если в размерах списков есть разница, то тест завершится ошибкой.

@Test  
public void getHowManyUsers() {  
 List<User> *expected* = User.*getAllUsers*();  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
 *actual*.add(user1);  
 *actual*.add(user2);  
  
 *assertEquals*(*expected*.size(), *actual*.size());  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода testGetHowManyUsers():

В этом тесте сравнивается ожидаемый размер списка пользователей с конкретным полом, с текущим размером списка пользователей и если в размерах списков есть разница, то тест завершится ошибкой.

@Test  
public void testGetHowManyUsers() {  
 List<User> *expected* = User.*getAllUsers*(Sex.*MALE*);  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
  
 *assertEquals*(*expected*.size(), *actual*.size());  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода getAllAgeUsers():

В этом тесте сравнивается возраст всех пользователей в списке с возрастом всех пользователей текущего списка и если есть разница с общим возрастом в двух списках, то тест завершится ошибкой.

@Test  
public void getAllAgeUsers() {  
 List<User> *expected* = User.*getAllUsers*();  
  
 int *expectedCountAge* = 0;  
 for (User *user* : *expected*) {  
 *expectedCountAge* += *user*.getAge();  
 }  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
 *actual*.add(user1);  
 *actual*.add(user2);  
  
 int *actualCountAge* = 0;  
 for (User *user* : *actual*) {  
 *actualCountAge* += *user*.getAge();  
 }  
  
 *assertEquals*(*expectedCountAge*, *actualCountAge*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода testGetAllAgeUsers():

В этом тесте сравнивается возраст всех пользователей с определенным полом в списке с возрастом всех пользователей текущего списка и если есть разница с общим возрастом в двух списках, то тест завершится ошибкой.

@Test  
public void testGetAllAgeUsers() {  
 List<User> *expected* = User.*getAllUsers*(Sex.*MALE*);  
  
 int *expectedCountAge* = 0;  
 for (User *user* : *expected*) {  
 *expectedCountAge* += *user*.getAge();  
 }  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
  
 int *actualCountAge* = 0;  
 for (User *user* : *actual*) {  
 *actualCountAge* += *user*.getAge();  
 }  
  
 *assertEquals*(*expectedCountAge*, *actualCountAge*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода getAverageAgeOfAllUsers():

В этом тесте сравнивается средний возраст всех пользователей в списке со средним возрастом пользователей в текущем списке и если есть разница, то тест завершится ошибкой.

@Test  
public void getAverageAgeOfAllUsers() {  
 List<User> *expected* = User.*getAllUsers*();  
  
 int *expectedCountAge* = 0;  
 for (User *user* : *expected*) {  
 *expectedCountAge* += *user*.getAge();  
 }  
   
 int *expectedAverageAge* = *expectedCountAge* / *expected*.size();  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
 *actual*.add(user1);  
 *actual*.add(user2);  
  
 int *actualCountAge* = 0;  
 for (User *user* : *actual*) {  
 *actualCountAge* += *user*.getAge();  
 }  
   
 int *actualAverageAge* = *actualCountAge* / *expected*.size();  
  
 *assertEquals*(*expectedAverageAge*, *actualAverageAge*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно

Тестирование метода testGetAverageAgeOfAllUsers():

В этом тесте сравнивается средний возраст всех пользователей с определенным полом в списке, со средним возрастом пользователей в текущем списке и если есть разница, то тест завершится ошибкой.

@Test  
public void testGetAverageAgeOfAllUsers() {  
 List<User> *expected* = User.*getAllUsers*(Sex.*MALE*);  
  
 int *expectedCountAge* = 0;  
 for (User *user* : *expected*) {  
 *expectedCountAge* += *user*.getAge();  
 }  
  
 int *expectedAverageAge* = *expectedCountAge* / *expected*.size();  
  
 List<User> *actual* = new ArrayList<>();  
 *actual*.add(user);  
  
 int *actualCountAge* = 0;  
 for (User *user* : *actual*) {  
 *actualCountAge* += *user*.getAge();  
 }  
  
 int *actualAverageAge* = *actualCountAge* / *expected*.size();  
  
 *assertEquals*(*expectedAverageAge*, *actualAverageAge*);  
}

Ожидаемый результат: Тест пройден

Результат теста: Тест пройден успешно