Politechnika Warszawska
Wydział Elektryczny

Data utworzenia: 24/11/2015
Ostatnia modyfikacja: 26/11/2015

Projekt 1 - Cosplay

Testowanie i weryfikacja oprogramowania

Autorzy: Jadczak Krzysztof, Jereczek Michał, Łopatka Jagoda, Porowski Dariusz

Spis treści

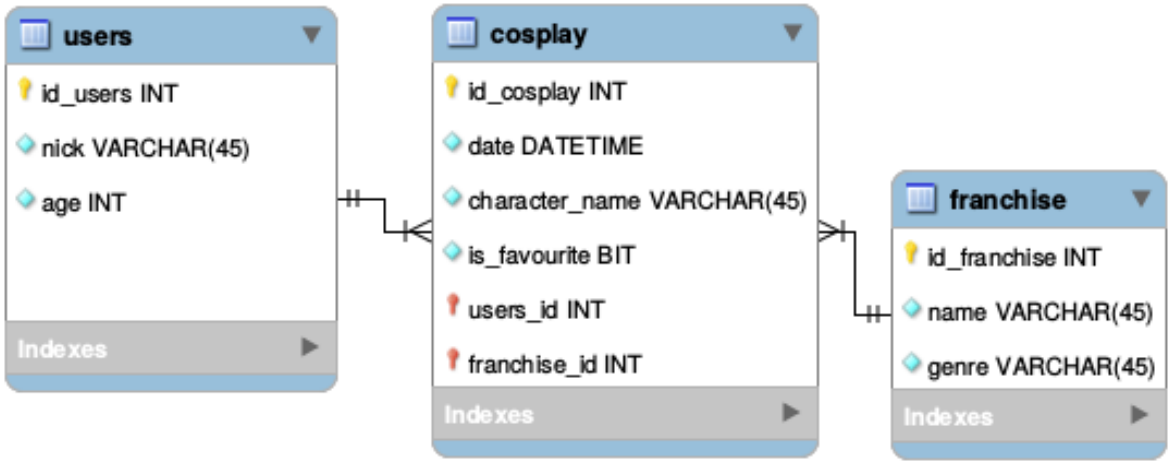
1 Program

1.1 Krótki opis programu

Naszym programem jest narzędzie pozwalające na wykonywanie operacji na bazie danych o cosplayer'ach. Baza ta umożliwia przetrzymywanie informacji o jej użytkownikach (cosplayer'ach) oraz bazie postaci, za które się oni przebierali. Ponadto baza umożliwia nam klasyfikację postaci wedle uniwersum, do którego należą.

Głównym produktem naszej pracy jest klasa CosplayDatabaseAPI, dzięki której jesteśmy w stanie wykonywać podstawowe operacje na bazie danych. Cały projekt zamieszczony jest pod adresem `https://github.com/trzye/TIWO-1`.

1.2 Baza danych



1.3 Klasa CosplayDatabaseAPI

Nasz program udostępnia klasę CosplayDatabaseAPI. Klasa ta posiada statyczne metody pozwalające na różne operacje na naszej bazie danych. Ich opis jest przedstawiony poniżej.

- **addCosplay** - Dodanie nowej informacji o cosplay'u.
- **addFranchise** - Dodawanie uniwersum do bazy danych.
- **addUser** - Dodawanie użytkownika do bazy danych.
- **changeUserAge** - Zmiana wieku użytkownika.
- **deleteUserAndHisCosplayData** - Usuwanie wszystkich danych użytkownika.
- **getUserData** - Zwracanie informacji o użytkowniku.

Dokładny opis argumentów oraz wyrzucanych wyjątków przedstawiony jest w dokumentacji Javadoc dołączonej do programu.

2 Testy

Testy klasy CosplayDatabaseAPI zostały przeprowadzone na poziomie kodu z wykorzystaniem możliwości jUnit.

2.1 AddCosplayMethodTester

Klasa testująca obsługę dodawania nowej informacji o cosplay'u - addCosplay(Timestamp date, Boolean isFavourite, String characterName, String franchiseName, String userNick) Sprawdzane wyjątki:

- CantFindTheUserException - nie można znaleźć userNick w bazie użytkowników
- EmptyStringException - w argumentach są puste Stringi
- CantFindFranchiseException - nie można znaleźć franchiseName w bazie uniwersów
- StringLongerThan45Exception - argument String większy niż 45 znaków (limit bazy danych)
- NullPointerException - argument został wpisany jako null

2.1.1 Poprawne dane

Stworzono nową informację o cosplay'u z prawidłowymi danymi przy użyciu metod

- CosplayDatabaseAPI.addUser(nick, age)
- CosplayDatabaseAPI.addFranchise(name, genre)
- CosplayDatabaseAPI.addCosplay(data, isFavourite, characterName, name, nick)

i porównano te dane z danymi, które trafiły do bazy, za pomocą metod z org.junit.Assert.

2.1.2 Niepoprawne dane - CantFindTheUserException

Podczas tego testu wcześniej nie został dodany do bazy żaden użytkownik i zostało sprawdzone czy metoda addCosplay wyrzuci ten wyjątek. Drugim przypadkiem, który został sprawdzony to utworzenie zmiennej String nickUser, która nie pokrywa się z żadnym użytkownikiem w bazie.

2.1.3 Niepoprawne dane - EmptyStringException

W tym teście w polu argumentów metody addCosplay, które są typu String wpisano puste ciągi. Zbadano wszystkie trzy argumenty tego typu.

2.1.4 Niepoprawne dane - CantFindFranchiseException

Zbadano przypadki gdy w bazie nie ma ani jednego uniwersum oraz gdy zostanie wpisany argument franchiseName, który nie pokrywa się z danymi w bazie.

2.1.5 Niepoprawne dane - StringLongerThan45Exception

Dla wszystkich argumentów String, został przeprowadzony test, w którym dany argument był dłuższy niż 45 znaków.

2.1.6 Niepoprawne dane - NullPointerException

Zbadano przypadki dla wszystkich argumentów gdy zostaną wpisane z wartością null.

2.2 AddFranchiseMethodTester

Klasa testująca obsługę dodawania nowego uniwersum - addFranchise(String name, String genre)
Sprawdzane wyjątki:

- DuplicateEntryException - gdy chcemy dodać isniejące już uniwersum (name się powtarza)
- EmptyStringException - w argumentach są puste Stringi
- StringLongerThan45Exception - argument String większy niż 45 znaków (limit bazy danych)
- NullPointerException - argument został wpisany jako null

2.2.1 Poprawne dane

Stworzono nowe uniwersum z prawidłowymi danymi przy użyciu metody CosplayDatabaseAPI.addFranchise(name, genre) i porównano te dane z danymi, które trafiły do bazy, za pomocą metod z org.junit.Assert.

2.2.2 Niepoprawne dane - DuplicateEntryException

W tym teście została przeprowadzona próba użycia metody addFranchise dwukrotnie z tymi samymi argumentami.

2.2.3 Niepoprawne dane - EmptyStringException

W teście w polu argumentów metody addFranchise, które są typu String wpisano puste ciągi. Występuje jeden taki argument - name.

2.2.4 Niepoprawne dane - CantFindFranchiseException

Zbadano przypadki gdy w bazie nie ma ani jednego uniwersum oraz gdy zostanie wpisany argument franchiseName, który nie pokrywa się z danymi w bazie.

2.2.5 Niepoprawne dane - StringLongerThan45Exception

Dla argumentu name, który jest typu String, został przeprowadzony test, w którym dany argument był dłuższy niż 45 znaków.

2.2.6 Niepoprawne dane - NullPointerException

Zbadano przypadki dla obu argumentów metody addFranchise, którym przypisano wartość null.

2.3 AddUserMethodTester

AddUserMethodTester jest zbiorem testów dla funkcji:
AddUser(String nick, Integer age) oraz jej poprawności reagowania na wyjątki:

- DuplicateEntryException
- StringLongerThan45Exception
- EmptyStringException
- AgeLowerThanOneException

2.3.1 Poprawne dane

Test reagowania na wprowadzenie poprawnych danych został przeprowadzony poprzez wykreowanie nowego użytkownika, przy pomocy funcji:

- CosplayDatabaseAPI.addUser(nick, age)

przy założeniu iż jest ona zaimplementowana poprawnie. Następnie pobrano dane o użytkownikach. Z racji przeprowadzenia testu na pustej bazie przed dodaniem użytkownika lista użytkownika byłaby pusta, tak więc jeżeli testowana funkcja zadzałałaby poprawnie w efekcie otrzymanoby dokładnie jednego użytkownika z wpisanymi wcześniej na początku danymi. Ta część testu została przeprowadzona przy użyciu biblioteki Assert, funkcjami:

- Assert.assertEquals(users.size(), 1)
- Assert.assertEquals(nick,users.get(0).getNick())
- Assert.assertEquals(age,users.get(0).getAge())

2.3.2 Niepoprawne dane - DuplicateEntryException

DuplicateEntryException jest wyjątkiem zgłaszanym w przypadku znalezienia w bazie użytkownika o tych samych rekordach. Test został przeprowadzony poprzez próbę wstawienia kolejnego identycznego użytkownika do bazy

2.3.3 Niepoprawne dane - StringLongerThan45Exception

StringLongerThan45Exception jest wyjątkiem zgłaszanym w przypadku przekroczenia przez wprowadzany nick limitu jego wielkości. Test został przeprowadzony poprzez stworzenie zmiennej z domniemanym nickiem użytkownika o zadługiej nazwie.

2.3.4 Niepoprawne dane - EmptyStringException

EmptyStringException jest wyjątkiem zgłaszanym w przypadku gdy wprowadzany nick jest pusty. Test został przeprowadzony poprzez stworzenie zmiennej z pustym nickiem domniemanego użytkownika.

2.3.5 AgeLowerThenOneExcepted

AgeLowerThenOneExcepted jest wyjątkiem zgłaszanym w przypadku gdy wprowadzany wiek jest «=0". Test został rodzielony na dwie części - jedną gdy podawany jest age=0, oraz drugą gdy age podawany jest jako ujemny.

2.4 ChangeUserAgeMethodTester

ChangeUserAgeMethodTester jest zbiorem testów dla funkcji: ChangeUser(String nick, Integer newAge) oraz jej poprawności reagowania na wyjątki:

- CantFindTheUserException
- StringLongerThan45Exception
- EmptyStringException
- AgeLowerThanOneException

2.4.1 Poprawne dane

Test reagowania na wprowadzenie poprawnych danych został przeprowadzony poprzez wykreowanie nowego użytkownika, przy pomocy funkcji

- CosplayDatabaseAPI.addUser(nick, age)

przy założeniu iż jest ona zaimplementowana poprawnie. Następnie pobrano dane użytkownika sprawdzając czy jego wiek zgadza się z wpisaną wartością

- Assert.assertEquals(age, users.get(0).getAge())

następnie używana jest funkcja która ma zmienić pole wiek użytkownika na nowe, oraz sprawdzane jest czy pole wiek zostało zmienione na drugą daną newAge

- CosplayDatabaseAPI.changeUserAge(nick, newAge);
- Assert.assertEquals(age, users.get(0).getAge())

Dzięki temu porównaniu wiemy czy funkcja ta zadziałała poprawnie.

2.4.2 Niepoprawne dane - StringLongerThan45Exception

StringLongerThan45Exception jest wyjątkiem zgłaszanym w przypadku przekroczenia przez wprowadzany nick limitu jego wielkości. Test został przeprowadzony poprzez stworzenie zmiennej z domniemanym nickiem użytkownika o zadługiej nazwie.

2.4.3 Niepoprawne dane - EmptyStringException

EmptyStringException jest wyjątkiem zgłaszanym w przypadku gdy wprowadzany nick jest pusty. Test został przeprowadzony poprzez stworzenie zmiennej z pustym nickiem domniemanego użytkownika.

2.4.4 AgeLowerThenOneExcepted

AgeLowerThenOneExcepted jest wyjątkiem zgłaszanym w przypadku gdy wprowadzany wiek jest «=0". Test został rodzielony na dwie części - jedną gdy podawany jest age=0, oraz drugą gdy age podawany jest jako ujemny.

2.4.5 CantFindTheUserException

2.5 DeleteUserAndHisCosplayDataMethodTester

DeleteUserAndHisCosplayDataMethodTester jest zbiorem testów dla funkcji:
deleteUserAndHisCosplayData(String nick) oraz jej poprawności reagowania na wyjątki:

- CantFindTheUserException
- StringLongerThan45Exception
- EmptyStringException

2.5.1 Poprawne dane

Test reagowania na wprowadzenie poprawnych danych został przeprowadzony poprzez wykreowanie nowego użytkownika wraz z jego danymi cosplay'owymi, przy pomocy funcji:

- CosplayDatabaseAPI.addUser(nick, age)
- CosplayDatabaseAPI.addFranchise(name, genre)
- CosplayDatabaseAPI.addCosplay(data, isFavourite, characterName, name, nick)

przy założeniu iż są one zaimplementowane poprawnie. Następnie wywołano testowaną funkcję i przy pomocy funkcji: getCosplayList(), getUsersList(), utworzono listy cosplay oraz users. Z racji przeprowadzenia testu na pustej bazie przed dodaniem użytkownika obie listy byłyby puste, tak więc jeżeli testowana funkcja zadzałałaby poprawnie w efekcie otrzymanoby również puste listy. Ta część testu została przeprowadzona przy użyciu biblioteki Assert, funkcjami:

- Assert.assertEquals(users.isEmpty(), true)
- Assert.assertEquals(cosplay.isEmpty(), true)

Test przeszedł pozytywnie.

2.5.2 Niepoprawne dane - CantFindTheUserException

CantFindTheUserException jest wyjątkiem zgłaszanym w przypadku nie znalezienia w bazie użytkownika o szukanym nick'u. Test został przeprowadzony na pustej bazie (bez jakiegokolwiek użytkownika) poprzez stworzenie zmiennej z domniemanym nickiem użytkownika

String nick ="hakunamatata";

i wywołanie funkcji CosplayDatabaseAPI.deleteUserAndHisCosplayData(nick) , która nie mogła znaleść użytkownika o tym nick'u, co powinno zaowocować zwróceniem wyjątku.

Test przeszedł pomyślnie, wyjątek został zgłoszony.

2.5.3 Niepoprawne dane - StringLongerThan45Exception

StringLongerThan45Exception jest wyjątkiem zgłaszanym w przypadku przekroczenia przez wprowadzany nick limitu jego wielkości. Test został przeprowadzony poprzez stworzenie zmiennej z domniemanym nickiem użytkownika o zadługiej nazwie.

String nick ="testusereetestusereetestusereetestusereetestusereetestuseree";

i wywołanie funkcji CosplayDatabaseAPI.deleteUserAndHisCosplayData(nick) , która po sprawdzeniu długości nick'a odrazu zgłosiła wyjątek.

Test przeszedł pomyślnie, wyjątek został zgłoszony.

2.6 GetUserDataMethodTester

GetUserDataMethodTester jest zbiorem testów dla funkcji:
getUserData(String nick) oraz jej poprawności reagowania na wyjątki:

- CantFindTheUserException
- StringLongerThan45Exception
- EmptyStringException

2.6.1 Poprawne dane

Test reagowania na wprowadzenie poprawnych danych został przeprowadzony poprzez wykreowanie nowego użytkownika wraz z jego danymi cosplay'owymi, przy pomocy funcji:

- CosplayDatabaseAPI.addUser(nick, age)
- CosplayDatabaseAPI.addFranchise(name, genre)
- CosplayDatabaseAPI.addCosplay(data, isFavourite, characterName, name, nick)

przy założeniu iż są one zaimplementowane poprawnie. Następnie wywołano testowaną funkcję

CosplayDatabaseAPI.getUserData(nick) i utworzono user typu UsersEntity. Następnie przeprowazono sprawdzenie poprawności zwracanych przez nią danych przy użyciu biblioteki Assert, funkcjami sprawdzajacymi czy elementy zwróconego usera są zgodne z naszymi zmiennymi wykorzystywanymi do jego utworzenia:

- Assert.assertEquals(user.getAge(), age);
- Assert.assertEquals(user.getNick(), nick);

Test przeszedł pozytywnie.

2.6.2 Niepoprawne dane - CantFindTheUserException

CantFindTheUserException jest wyjątkiem zgłaszanym w przypadku nie znalezienia w bazie użytkownika o szukanym nick'u. Test został przeprowadzony na pustej bazie (bez jakiegokolwiek użytkownika) poprzez stworzenie zmiennej z domniemanym nickiem użytkownika

String nick ="hakunamatata";

i wywołanie funkcji CosplayDatabaseAPI.getUserData(nick) , która nie mogła znaleść użytkownika o tym nick'u, co powinno zaowocować zwróceniem wyjątku.

Test przeszedł pomyślnie, wyjątek został zgłoszony.

2.6.3 Niepoprawne dane - StringLongerThan45Exception

StringLongerThan45Exception jest wyjątkiem zgłaszanym w przypadku przekroczenia przez wprowadzany nick limitu jego wielkości. Test został przeprowadzony poprzez stworzenie zmiennej z domniemanym nickiem użytkownika o zadługiej nazwie.

String nick ="testusereetestusereetestusereetestusereetestusereetestuseree";

i wywołanie funkcji CosplayDatabaseAPI.getUserData(nick) , która po sprawdzeniu długości nick'a odrazu zgłosiła wyjątek.

Test przeszedł pomyślnie, wyjątek został zgłoszony.

2.6.4 Niepoprawne dane - EmptyStringException

EmptyStringException jest wyjątkiem zgłaszanym w przypadku gdy wprowadzany nick jest pusty. Test został przeprowadzony poprzez stworzenie zmiennej z pustym nickiem domniemanego użytkownika.

String nick ="";

i wywołanie funkcji CosplayDatabaseAPI.getUserData(nick) , która po sprawdzeniu czy nick jest pusty zgłosiła wyjątek.

Test przeszedł pomyślnie, wyjątek został zgłoszony.