**Отчёт о найденных ошибках в классе AccountManager**

Мавлетова Карина Радиковна, БПИ216

**Метод init**

Данный метод просто инициализирует объекты. Соответственно нужно проверить, что после вызова метода данные объекты не равны null.

1. **Код до исправления**

public void init(IServer s, IPasswordEncoder encoder) {  
 server = s;  
}

1. **Данные, на которых наблюдает некорректное поведение**

Передача любых объектов IServer и IPasswordEncoder выдаёт проблемы.

1. **Полученные и ожидаемые значения**

Тут я пошла от противного и решила проверить выдаст ли метод callLogin после инициализации значений NullPointerException.

Поэтому ожидаемое и полученное значения совпали.

1. **Код после исправления**

public void init(IServer s, IPasswordEncoder encoder) {  
 server = s;  
 passEncoder = encoder;  
}

**Метод callLogin (первая проблема)**

1. **Код до исправления (только нужная часть)**

case ServerResponse.*SUCCESS*:{  
 Object answ = ret.response;  
 if(answ instanceof Long)  
 return new AccountManagerResponse(AccountManagerResponse.*SUCCEED*, answ);  
 break;   
}

1. **Данные, на которых наблюдает некорректное поведение**

Если пользователь залогинился, то его нужно добавить в список авторизованных на текущей сессии. Поэтому при первом вызове callLogin мы должны получить success, при этом при повторном вызове должны получить результат already\_logged.

1. **Полученные и ожидаемые значения**

Полученное: код SUCCEED

Ожидаемое: код ALREADY\_LOGGED

1. **Код после исправления**

case ServerResponse.*SUCCESS*:{  
 Object answ = ret.response;  
 if(answ instanceof Long) {  
 activeAccounts.put(login, (Long) answ);  
 return new AccountManagerResponse(AccountManagerResponse.*SUCCEED*, answ);  
 }  
 break;   
}

**Метод callLogin (вторая проблема)**

1. **Код до исправления (только нужная часть)**

case ServerResponse.*ALREADY\_LOGGED*:  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;

1. **Данные, на которых наблюдает некорректное поведение**

Если пользователь залогинился, а потом разлогинился, то у него нет возможности снова попасть в список авторизованных пользователей.

1. **Полученные и ожидаемые значения**

Проверяем, что при повторном вызове callLogin вернётся ответ already\_logged.

Полученное: код ALREADY\_LOGGED

Ожидаемое: код ALREADY\_LOGGED

Но тут есть примечание: требования так устроены, что мы получаем одинаковый ответ в случае если пользователь есть в списке текущей сессии и если он в целом есть в бд. Это в свою очередь не даёт возможности корректно протестировать, что будучи уже в бд, его добавляют в список авторизованных пользователей.

Как делала я: замокала ответ, что человек уже есть в бд. По сути тут его должны добавить и уже при повторном вызове данного метода я бы ожидала получить, что вот он уже авторизован. Да, я его получу, только это будет ответ из части кода с case, т.е что он был в бд. В общем тут эти два случая тестами не обойти из-за одинаковых ответов. Но код я всё равно решила поправить☺

1. **Код после исправления**

case ServerResponse.*ALREADY\_LOGGED*:  
 activeAccounts.put(login, (Long) ret.response);  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;

Теперь если пользователя нет в списке авторизованных, но при этом он залогинен и снова пытается авторизоваться, то всё будет ок.

**Метод callLogin (третья проблема)**

Не очень понятно, как обработать ситуацию с ошибкой кодирования пароля. Метод callLogin может возвращать ENCODING\_ERROR. При этом ниже сказано, что если мы передаём null или же происходят проблемы с шифрованием, то нужно выкинуть исключение NullPointerException. И вот тут не очень понятно два момента. Во-первых, как обработать случай, что пароль закодировался неверно? Так как для мока мы передаём уже готовую строку. В целом можно передать что-то кривое, сравнить с тем, что должны бы были получить, а если не то вышло, то кинуть исключение. И судя по всему в catch вернуть из метода ENCODING\_ERROR. Но не уверена, насколько это не костыльный вариант. Именно с проверкой ошибки шифрования.

**Метод callLogout (первая проблема)**

1. **Код до исправления**

public AccountManagerResponse callLogout(String user, long session) {  
 Long rem = activeAccounts.remove(user);  
 if(rem == null)  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 ServerResponse resp = server.logout(session);  
 switch(resp.code){  
 case ServerResponse.*NOT\_LOGGED*:  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:  
 return AccountManagerResponse.*SUCCEED\_RESPONSE*;  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, resp);  
}

1. **Данные, на которых наблюдает некорректное поведение**

Если пользователь зарегистрировался/авторизовался с одним кодом сессии, а потом по тому же имени в метод выхода из системы передаётся другой код сессии, то метод срабатывает некорректно.

1. **Полученные и ожидаемые значения**

*Полученное:* объект типа IServer оказался равным null. Перед вызовом метода callLogout у администратора я не ставила мок на объект IServer. По причине, что если сессия некорректная – запрос на сервер вообще не должен был уйти. А он ушёл, а мока нет, поэтому null

*Ожидаемое:* INCORRECT\_SESSION

1. **Код после исправления**

public AccountManagerResponse callLogout(String user, long session) {  
 Long storedSession = activeAccounts.remove(user);  
 if(storedSession == null)  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 if(storedSession!=session){  
 return AccountManagerResponse.*INCORRECT\_SESSION\_RESPONSE*;  
 }  
 ServerResponse resp = server.logout(session);  
 switch(resp.code){  
 case ServerResponse.*NOT\_LOGGED*:  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:  
 return AccountManagerResponse.*SUCCEED\_RESPONSE*;  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, resp);  
}

**Метод callLogout (вторая проблема)**

1. **Код до исправления**

public AccountManagerResponse callLogout(String user, long session) {  
 Long storedSession = activeAccounts.remove(user);  
 if(storedSession == null)  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 if(storedSession!=session){  
 return AccountManagerResponse.*INCORRECT\_SESSION\_RESPONSE*;  
 }  
 ServerResponse resp = server.logout(session);  
 switch(resp.code){  
 case ServerResponse.*NOT\_LOGGED*:  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:  
 return AccountManagerResponse.*SUCCEED\_RESPONSE*;  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, resp);  
}

1. **Данные, на которых наблюдает некорректное поведение**

Юзера не должно удалять из списка авторизованных в текущей сессии, если передан некорректный номер сессии.

1. **Полученные и ожидаемые значения**

Расскажу, как я тестировала это. Для начала я залогинила юзера, далее попыталась из удалить из списка авторизованных, передав некорректный номер сессии. А потом снова попыталась его убрать из списка авторизованных.

Полученное: NOT\_LOGGED

Ожидаемое: INCORRECT\_SESSION

Т.е так вышло, что после первого вызова метода с некорректным номером сессии пользователя уже удалили, хотя не должны были.

1. **Код после исправления**

public AccountManagerResponse callLogout(String user, long session) {  
 Long storedSession = activeAccounts.get(user);  
 if(storedSession == null)  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 if(storedSession!=session){  
 return AccountManagerResponse.*INCORRECT\_SESSION\_RESPONSE*;  
 }  
 ServerResponse resp = server.logout(session);  
 switch(resp.code){  
 case ServerResponse.*NOT\_LOGGED*:

activeAccounts.remove(user);  
 return AccountManagerResponse.*NOT\_LOGGED\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:

activeAccounts.remove(user);  
 return AccountManagerResponse.*SUCCEED\_RESPONSE*;  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, resp);  
}

Тут одновременно учитывается случай, что пользователя не удалят, если будет INCORRECT\_RESPONSE.

**Метод deposit**

В самой реализации рассматривается случай с NO\_MONEY, но в требованиях про этот вариант ничего не сказано

**Проблема с UNDEFINED\_ERROR**

Я решила это вынести в отдельный блок, чтобы для каждого метода по отдельности не писать. Ни один метод не обрабатывает корректно ответ от сервера UNDEFINED\_ERROR. Данная ошибка засчитывается, как INCORRECT\_RESPONSE.

Исправляем вот так (покажу на примере одного метода):

**Код до исправления**

public AccountManagerResponse callLogin(String login, String password) {  
 Long session = activeAccounts.get(login);  
 if(session!=null)  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;  
 ServerResponse ret = server.login(login, passEncoder.makeSecure(password));  
 switch(ret.code) {  
 case ServerResponse.*ALREADY\_LOGGED*:  
 activeAccounts.put(login, (Long) ret.response);  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;  
 case ServerResponse.*NO\_USER\_INCORRECT\_PASSWORD*:  
 return AccountManagerResponse.*NO\_USER\_INCORRECT\_PASSWORD\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:{  
 Object answ = ret.response;  
 if(answ instanceof Long) {  
 activeAccounts.put(login, (Long) answ);  
 return new AccountManagerResponse(AccountManagerResponse.*SUCCEED*, answ);  
 }  
 break;   
 }  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, ret);  
}

**Код после исправления**

public AccountManagerResponse callLogin(String login, String password) {  
 Long session = activeAccounts.get(login);  
 if(session!=null)  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;  
 ServerResponse ret = server.login(login, passEncoder.makeSecure(password));  
 switch(ret.code) {  
 case ServerResponse.*ALREADY\_LOGGED*:  
 activeAccounts.put(login, (Long) ret.response);  
 return AccountManagerResponse.*ACCOUNT\_MANAGER\_RESPONSE*;  
 case ServerResponse.*NO\_USER\_INCORRECT\_PASSWORD*:  
 return AccountManagerResponse.*NO\_USER\_INCORRECT\_PASSWORD\_RESPONSE*;  
 case ServerResponse.*UNDEFINED\_ERROR*:  
 return AccountManagerResponse.*UNDEFINED\_ERROR\_RESPONSE*;  
 case ServerResponse.*SUCCESS*:{  
 Object answ = ret.response;  
 if(answ instanceof Long) {  
 activeAccounts.put(login, (Long) answ);  
 return new AccountManagerResponse(AccountManagerResponse.*SUCCEED*, answ);  
 }  
 break;   
 }  
 }  
 return new AccountManagerResponse(AccountManagerResponse.*INCORRECT\_RESPONSE*, ret);  
}