Зорица Коцева 185043 КИ
Андреј Петрушев 181033 СИИС

Софтверски квалитет и тестирање

- Проект -

Логичко покривање на код (Logic Coverage)

```
@Override
public User register(String username, String password, String repeatPassword, String name, String surname, Role userRole) {
    if (username==null || username.isEmpty()  || password==null || password.isEmpty())
        throw new InvalidUsernameOrPasswordException();
    if (!password.equals(repeatPassword))
        throw new PasswordsDoNotMatchException();
    if(this.userRepository.findByUsername(username).isPresent())
        throw new UsernameAlreadyExistsException(username);
    User user = new User(username,passwordEncoder.encode(password),name,surname,userRole);
    return userRepository.save(user);
}
```

1) Предикатот кој произлегува од дадениот код е:

(username==null || username.isEmpty() || password==null || password.isEmpty())

За да се одредат табелите на вистинитост за овој предикат, во кој што се прави споредба на корисничкото име и на лозинката кои се внесуваат, дали истите се празни или пак е внесена некоја null вредност, се формираат соодветно 4 клаузули и истите се заменуваат со 4 променливи, се со цел подобро користење на логичките оператори и точно испишување во табелите.

Предикатот го добива следниот облик:

=> **(firstClause | secondClause | thirdClause | fourthClause)**

каде што, променливата firstClause го заменува изразот username==null, променливата secondClause го заменува изразот username.isEmpty(), променливата thirdClause го заменува изразот password==null и променливата fourthClause го заменува изразот password.isEmpty().

Зорица Коцева 185043 КИ
Андреј Петрушев 181033 СИИС

Truth Table:

Row#	firstClause	secondClause	thirdClause	fourthClause	P	PfirstClause	PsecondClause	PthirdClause	PfourthClause
1	T	T	T	T	T				
2	T	T	T		T				
3	T	T		T	T				
4	T	T			T				
5	T		T	T	T				
6	T		T		T				
7	T			T	T				
8	T				T	T			
9		T	T	T	T				
10		T	T		T				
11		T		T	T				
12		T			T		T		
13			T	T	T				
14			T		T			T	
15				T	T				T
16						T	T	T	T

Табела на вистинитост за наведениот предикат

Test requirements (**TR**) за следните покривања (CACC и RACC):

The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
firstClause	(8,16)
secondClause	(12,16)
thirdClause	(14,16)
fourthClause	(15,16)

The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
firstClause	(8,16)
secondClause	(12,16)
thirdClause	(14,16)
fourthClause	(15,16)

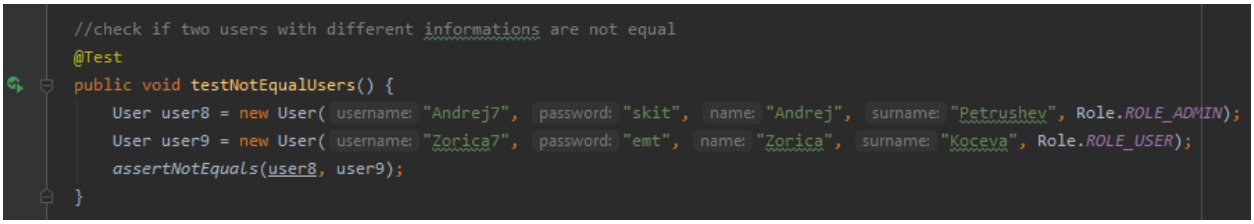
Зорица Коцева 185043 КИ
Андреј Петрушев 181033 СИИС

```java
import org.springframework.security.crypto.password.PasswordEncoder;

public class UserServiceImplTestsRACC {
    UserServiceImpl userService;
    public UserRepository userRepository;
    public PasswordEncoder passwordEncoder;

    public UserServiceImplTestsRACC() {}

//    public UserServiceImplTestsRACC(UserRepository userRepository, PasswordEncoder passwordEncoder) {
//        this.userRepository = userRepository;
//        this.passwordEncoder = passwordEncoder;
//    }

    @Before
    public void setUp() {
        userService = new UserServiceImpl(userRepository, passwordEncoder);
    }
```

```java
// Tests for table where predicate is (username==null || username.isEmpty()  || password==null || password.isEmpty())

// firstClause: T  secondClause: F  thirdClause: F  fourthClause: F  => P: T
@Test
public void testUsernameNull() {
    User user1 = new User( username: null,  password: "finki",  name: "Petko",  surname: "Petkovski", Role.ROLE_USER);
    if(user1.getUsername()==null) {
        assertTrue( message: "InvalidUsername - null",  condition: true);
    }
}

// firstClause: F  secondClause: T  thirdClause: F  fourthClause: F  => P: T
@Test
public void testInvalidEmptyUsername() {

    User user2 = new User( username: "",  password: "finki",  name: "Bojana",  surname: "Koteska", Role.ROLE_ADMIN);
    if(user2.getUsername().isEmpty()) {
        assertTrue( message: "InvalidUsername - empty",  condition: true);
    }
}
```

```java
// firstClause: F  secondClause: F  thirdClause: T  fourthClause: F  => P: T
@Test
public void testPasswordNull() {
    //logicCoverageClass.cal(0,15,10,9,2012);
    User user3 = new User( username: "skit99",  password: null,  name: "Softverski kvalitet",  surname: "Testiranje", Role.ROLE_USER);
    if(user3.getPassword()==null) {
        assertTrue( message: "InvalidPassword - null",  condition: true);
    }
}

// firstClause: F  secondClause: F  thirdClause: F  fourthClause: T  => P: T
@Test
public void testInvalidEmptyPassword() {

    User user4 = new User( username: "zocche",  password: "",  name: "Zorica",  surname: "Koceva", Role.ROLE_USER);
    if(user4.getPassword().isEmpty()){
        assertTrue( message: "InvalidPassword - empty",  condition: true);
    }
}
```

```
@Test
public void testCorrectUser() {

    User user5 = new User( username: "Marko7",  password: "youtube",  name: "Marko",  surname: "Markovikj", Role.ROLE_ADMIN);
    if (!(user5.getUsername()==null || user5.getUsername().isEmpty() || user5.getPassword()==null || user5.getPassword().isEmpty())){
        assertFalse( message: "User with correct and valid info",  condition: false);
    }
}

//check if two users with same informations are equal
@Test
public void testEqualUsers() {
    User user6 = new User( username: "Andrej7",  password: "skit",  name: "Andrej",  surname: "Petrushev", Role.ROLE_ADMIN);
    User user7 = new User( username: "Andrej7",  password: "skit",  name: "Andrej",  surname: "Petrushev", Role.ROLE_ADMIN);
    assertEquals(user6, user7);
}
```



```
//check if two users with different informations are not equal
@Test
public void testNotEqualUsers() {
    User user8 = new User( username: "Andrej7",  password: "skit",  name: "Andrej",  surname: "Petrushev", Role.ROLE_ADMIN);
    User user9 = new User( username: "Zorica7",  password: "emt",  name: "Zorica",  surname: "Koceva", Role.ROLE_USER);
    assertNotEquals(user8, user9);
}
```

RACC (Restricted Active Clause Coverage) – за секој предикат p во множеството на предикати P и за секоја главна класузула Ci во множеството на клаузули Cp, се одбираат помали или споредни клаузули Cj, така што j!=i, а притоа клаузулата Ci го детерминира p. За секоја главна клаузула Ci има две тест побарувања, а тоа е дека треба да се евалуира во точно и во неточно. Вредностите на малите клаузули како составен дел на главната мора да бидат исти во ситуација кога главната клаузула се евалуира во точно, како и кога се евалуира во неточно.

CACC (Correlated Active Clause Coverage) – за секој предикат p во множеството на предикати P и за секоја главна класузула Ci во множеството на клаузули Cp, се одбираат помали или споредни клаузули Cj, така што j!=i, а притоа клаузулата Ci го детерминира p. За секоја главна клаузула Ci има две тест побарувања, а тоа е дека треба да се евалуира во точно и во неточно. Вредностите на малите клаузули како составен дел на главната мора да предизвикаат предикатот p да биде точен за една вредност на главната клаузула и неточен за друга вредност на главната клаузула.