JUnit tesztek eredménye

A java alapú backend a JUnit 4 egységteszt-keretrendszer használatával lett tesztelve. A Felhasználó és a játék entitás validáló metódusainak működése, a megfelelő kivétel osztályok alkalmazása és a kivétel pontos szövege.

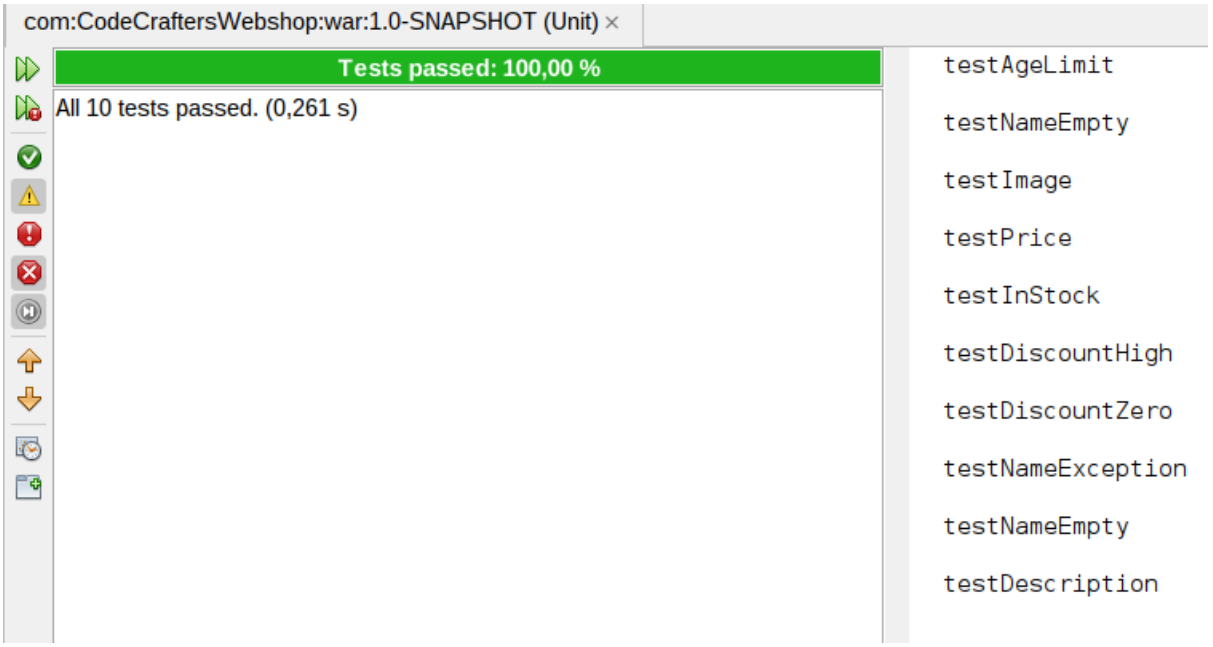
A következő metódusok a Játék entitáshoz tartoznak, amik a játékkal kapcsolatos elemeket tesztelik.

```java
@Test
public void testPrice() throws GameException {
    System.out.println(x: "testPrice");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék ára nem lehet kisebb, mint 0!");
    int price = -1;
    boolean result = Game.checkPrice(price);

    assertTrue(condition:result);
}

@Test
public void testDescription() throws GameException {
    System.out.println(x: "testDescription");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék leírása nem lehet üres!");
    String description = "";
    boolean result = Game.checkDescription(description);

    assertTrue(condition:result);
}

@Test
public void testImage() throws GameException {
    System.out.println(x: "testImage");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék képe nem lehet üres!");
    String image = "";
    boolean result = Game.checkImage(image);

    assertTrue(condition:result);
}

@Test
public void testAgeLimit() throws GameException {
    System.out.println(x: "testAgeLimit");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék korhatára nem térhet el a PEGI számoktól!");
    int age = 13;
    boolean result = Game.checkAgeLimit(age);

    assertTrue(condition:result);
}
```

```java
@Test(expected = GameException.class)
public void testNameException() throws GameException {
    System.out.println(x: "testNameException");

    String name = "";
    boolean result = Game.checkName(name);

    assertTrue(condition: result);
}

@Rule
public ExpectedException thrown = ExpectedException.none();

@Test
public void testNameEmpty() throws GameException {
    System.out.println(x: "testNameEmpty");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring: "A játék neve nem lehet üres!");
    String name = "";
    boolean result = Game.checkName(name);

    assertTrue(condition: result);
}

@Test
public void testNameLong() throws GameException {
    System.out.println(x: "testNameEmpty");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring: "A játék neve nem lehet 100 karakternél hosszabb!");
    String name = "Forzakkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"
            + "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk";
    boolean result = Game.checkName(name);

    assertTrue(condition: result);
}
```

```java
@Test
public void testDiscountZero() throws GameException {
    System.out.println(x: "testDiscountZero");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék akciója nem lehet kisebb, mint 0!");
    int discount = -1;
    boolean result = Game.checkDiscount(discount);

    assertTrue(condition:result);
}

@Test
public void testDiscountHigh() throws GameException {
    System.out.println(x: "testDiscountHigh");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék akciója nem lehet nagyobb, mint 100!");
    int discount = 101;
    boolean result = Game.checkDiscount(discount);

    assertTrue(condition:result);
}

@Test
public void testInStock() throws GameException {
    System.out.println(x: "testInStock");

    thrown.expect(type:GameException.class);
    thrown.expectMessage(substring:"A játék raktáron lévő mennyisége nem lehet kisebb, mint 0!");
    int price = -1;
    boolean result = Game.checkInStock(inStock: price);

    assertTrue(condition:result);
}
```

A Játék tesztosztály eredménye hibakeresés módban.



A következő metódusok a Felhasználó entitáshoz tartoznak, amik a felhasználóval kapcsolatos elemeket tesztelik.

```java
@Test
public void testFirstNameSpecial() throws UserException {
    System.out.println(x: "testFirstNameSpecial");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó keresztneve nem tartalmazhat speciális karaktert!");

    String firstName = "Elek?";
    boolean result = User.checkFirstName(firstName);

    assertTrue(condition:result);
}

@Test
public void testEmailEmpty() throws UserException {
    System.out.println(x: "testEmailEmpty");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó emailje nem lehet üres!");

    String email = "";
    boolean result = User.checkEmail(email);

    assertTrue(condition:result);
}

@Test
public void testEmailLong() throws UserException {
    System.out.println(x: "testEmailLong");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó emailje nem lehet 100 karakternél hosszabb!");

    String email = "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"
            + "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk@gmail.com";
    boolean result = User.checkEmail(email);

    assertTrue(condition:result);
}
```

```java
@Test
public void testLastNameSpecial() throws UserException {
    System.out.println(x: "testLastNameSpecial");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó vezetékneve nem tartalmazhat speciális karaktert!");

    String lastName = "Elek?";
    boolean result = User.checkLastName(lastName);

    assertTrue(condition:result);
}

@Test
public void testFirstNameEmpty() throws UserException {
    System.out.println(x: "testFirstNameEmpty");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó keresztneve nem lehet üres!");

    String firstName = "";
    boolean result = User.checkFirstName(firstName);

    assertTrue(condition:result);
}

@Test
public void testFirstNameLong() throws UserException {
    System.out.println(x: "testFirstNameLong");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó keresztneve nem lehet 100 karakternél hosszabb!");

    String firstName = "Elekkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"
            + "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk";
    boolean result = User.checkFirstName(firstName);

    assertTrue(condition:result);
}
```

```java
@Test
public void testPasswordShort() throws UserException {
    System.out.println(x: "testPasswordShort");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó jelszava nem lehet 8 karakternél rövidebb!");

    String password = "asd123";
    boolean result = User.checkPassword(password);

    assertTrue(condition:result);
}

@Test
public void testPasswordNumber() throws UserException {
    System.out.println(x: "testPasswordNumber");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó jelszavának tartalmaznia kell számot!");

    String password = "asdASDqwe*";
    boolean result = User.checkPassword(password);

    assertTrue(condition:result);
}

@Test
public void testPasswordLetter() throws UserException {
    System.out.println(x: "testPasswordLetter");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó jelszavának tartalmaznia kell betűt!");

    String password = "123789456*";
    boolean result = User.checkPassword(password);

    assertTrue(condition:result);
}
```

```java
@Test
public void testFirstNameSpecial() throws UserException {
    System.out.println(x: "testFirstNameSpecial");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó keresztneve nem tartalmazhat speciális karaktert!");

    String firstName = "Elek?";
    boolean result = User.checkFirstName(firstName);

    assertTrue(condition:result);
}

@Test
public void testEmailEmpty() throws UserException {
    System.out.println(x: "testEmailEmpty");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó emailje nem lehet üres!");

    String email = "";
    boolean result = User.checkEmail(email);

    assertTrue(condition:result);
}

@Test
public void testEmailLong() throws UserException {
    System.out.println(x: "testEmailLong");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó emailje nem lehet 100 karakternél hosszabb!");

    String email = "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"
            + "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk@gmail.com";
    boolean result = User.checkEmail(email);

    assertTrue(condition:result);
}
```

```java
@Test
public void testEmailValid() throws UserException {
    System.out.println(x: "testEmailValid");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó emailjének tartalmaznia kell a @ karaktert!");

    String email = "kkgmail.com";
    boolean result = User.checkEmail(email);

    assertTrue(condition:result);
}

@Test
public void testPasswordEmpty() throws UserException {
    System.out.println(x: "testPasswordEmpty");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó jelszava nem lehet üres!");

    String password = "";
    boolean result = User.checkPassword(password);

    assertTrue(condition:result);
}

@Test
public void testPasswordLong() throws UserException {
    System.out.println(x: "testPasswordLong");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring:"A felhasználó jelszava nem lehet 100 karakternél hosszabb!");

    String password = "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"
            + "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk123456789*";
    boolean result = User.checkPassword(password);

    assertTrue(condition:result);
}
```

```java
@Test
public void testPasswordSpecial() throws UserException {
    System.out.println(x: "testPasswordSpecial");

    thrown.expect(type:UserException.class);
    thrown.expectMessage(substring: "A felhasználó jelszavának tartalmaznia kell speciális karaktert!");

    String password = "asdASD123q";
    boolean result = User.checkPassword(password);

    assertTrue(condition: result);
}
```

A Felhasználó tesztosztály eredménye hibakeresés módban.