TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

SENG453/SENG311
Software Quality Assurance

Assignment 1 Report

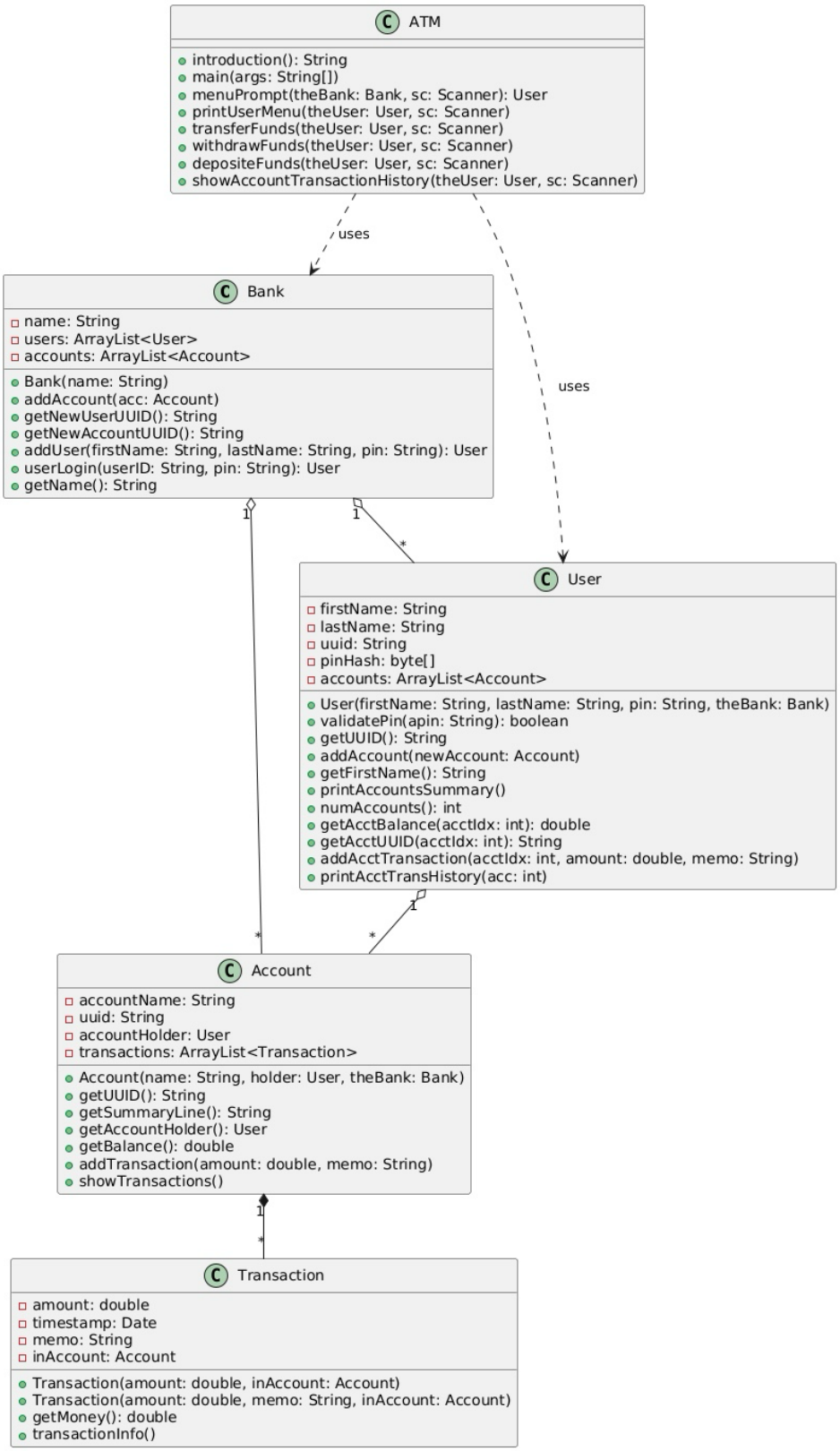
JUnit Test Programming

Hilal Yurtoğlu

17386054060

1. UML class diagram

Below is the UML diagram representing the project structure:



2. Assertion statements and outputs

JUnit Assertions is used to compare the results obtained at the end of the test scenario we have written with the expected results. If the expected result from a test scenario is the same as the result obtained, the test is considered successful.

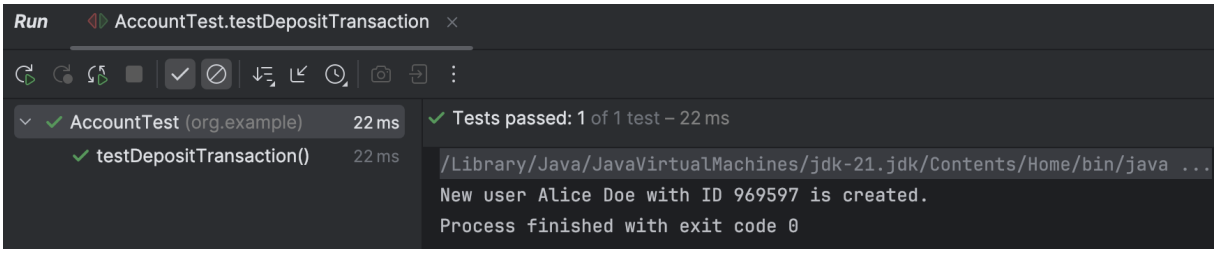
Below are some assertion test cases used in the project:

- assertEquals/assertNotEquals:

Checks if values are equal or not

```java
@Test
public void testDepositTransaction() {
    Bank bank = new Bank( name: "Test Bank");
    User user = new User( firstName: "Alice", lastName: "Doe", pin: "1234", bank);
    Account account = new Account( name: "Savings", user, bank);

    // 1000 units of money are deposited.
    account.addTransaction( amount: 1000, memo: "Initial deposit");
    // Balance should be 1000.
    assertEquals( expected: 1000, account.getBalance()); // assertEquals
    //Balance should not be 500
    assertNotEquals( unexpected: 500, account.getBalance());
}
```

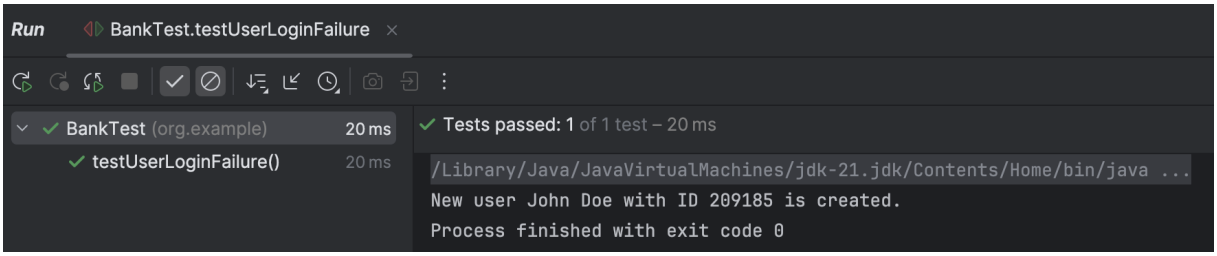


- assertNull:

Confirms if a value is null.

```java
@Test
public void testUserLoginFailure() {
    Bank bank = new Bank( name: "Test Bank");
    User user = bank.addUser( firstName: "John",  lastName: "Doe",  pin: "1234");

    // If you log in with a wrong user ID, null should be returned.
    assertNull(bank.userLogin( userID: "INVALID_ID",  pin: "1234"));
}
```



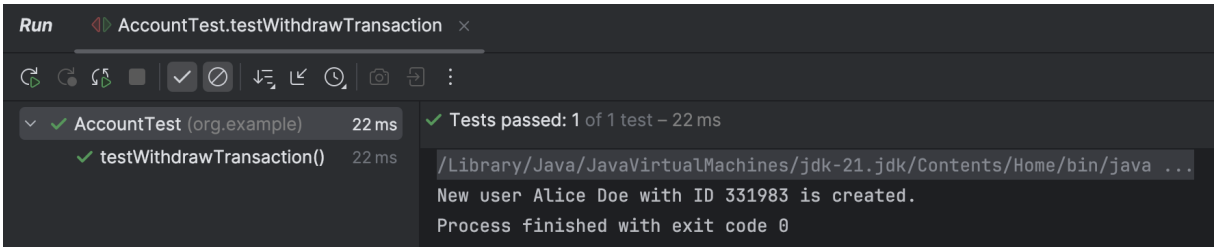
- assertTrue/assertFalse:

Ensures boolean expressions return expected results.

```java
@Test
public void testWithdrawTransaction() {
    Bank bank = new Bank( name: "Test Bank");
    User user = new User( firstName: "Alice",  lastName: "Doe",  pin: "1234", bank);
    Account account = new Account( name: "Savings", user, bank);

    // First 1000 units are deposited, then 500 are withdrawn.
    account.addTransaction( amount: 1000,  memo: "Deposit");
    account.addTransaction( amount: -500,  memo: "Withdraw");

    // Balance must be 500.
    assertTrue( condition: account.getBalance() > 0);  // assertTrue()
    assertFalse( condition: account.getBalance() < 0); // assertFalse()
}
```



- assertSame/assertNotSame:

Verifies if two references point to the same object.

```java
@Test
public void testTransactionCreation() {
    Bank bank = new Bank( name: "Test Bank");
    User user = new User( firstName: "Alice", lastName: "Doe", pin: "1234", bank);
    Account account = new Account( name: "Savings", user, bank);
    Transaction transaction1 = new Transaction( amount: 500, memo: "Deposit", account);
    Transaction transaction2 = transaction1;

    // Transaction amount must be 500.
    assertEquals( expected: 500, transaction1.getMoney());
    // Transaction amount must not be 600.
    assertNotEquals( unexpected: 600, transaction1.getMoney());
    // The two transaction objects must be the same.
    assertSame(transaction1, transaction2);

}
```

```
Run  TransactionTest.testTransactionCreation
TransactionTest (org.example) 23 ms
  testTransactionCreation() 23 ms
Tests passed: 1 of 1 test – 23 ms
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...
New user Alice Doe with ID 926211 is created.
Process finished with exit code 0
```

```java
@Test
public void testDifferentTransactions() {
    Bank bank = new Bank( name: "Test Bank");
    User user = new User( firstName: "Alice", lastName: "Doe", pin: "1234", bank);
    Account account = new Account( name: "Savings", user, bank);
    Transaction transaction1 = new Transaction( amount: 500, memo: "Deposit", account);
    Transaction transaction2 = new Transaction( amount: 1000, memo: "Deposit", account);

    // The two transaction objects must not be the same.
    assertNotSame(transaction1, transaction2);  // assertNotSame()
}
```

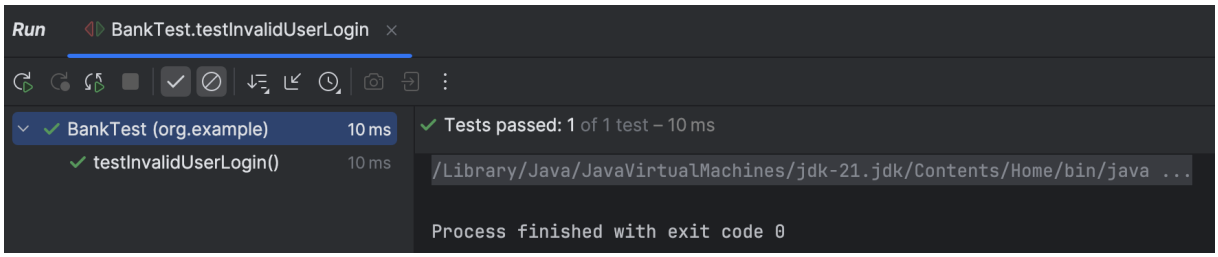
```
Run  TransactionTest.testDifferentTransactions
TransactionTest (org.example) 22 ms
  testDifferentTransactions() 22 ms
Tests passed: 1 of 1 test – 22 ms
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...
New user Alice Doe with ID 321796 is created.
Process finished with exit code 0
```

- assertThrows:

  Confirms if a method throws an expected exception.

```
@Test
public void testInvalidUserLogin() {
    Bank bank = new Bank( name: "Test Bank");

    // If you log in with the wrong UUID, an error should be thrown.
    assertThrows(NullPointerException.class, () -> {
        bank.userLogin( userID: "wrong_uuid",  pin: "1234").getFirstName(); // assertThrows()
    });
}
```



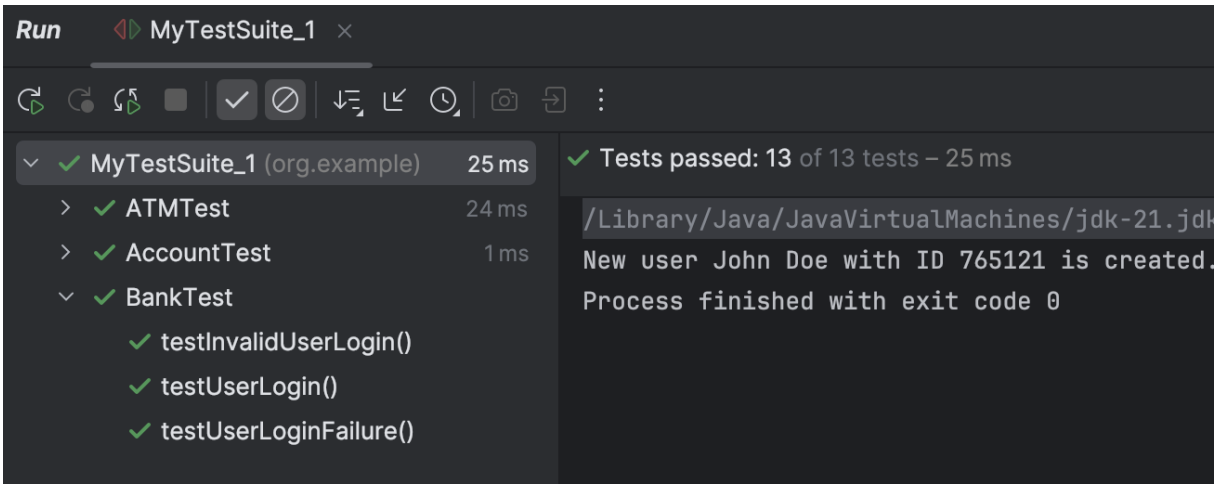
3. Test suits and outputs

Group multiple related test cases into suites for organized testing.

The project includes two test suites:

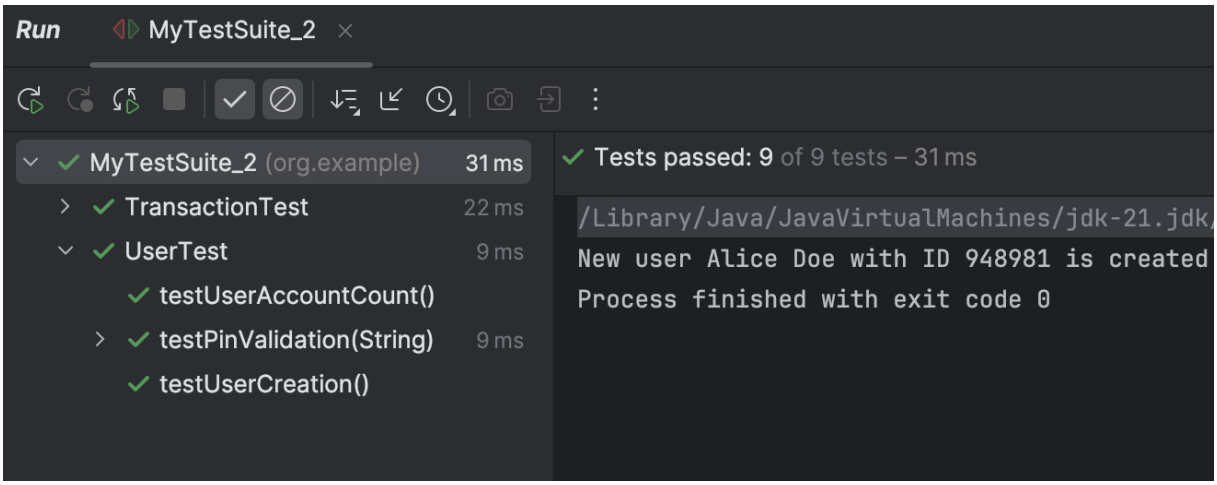
- MyTestSuite_1:

```
@Suite
@SelectClasses({
        ATMTest.class,
        AccountTest.class,
        BankTest.class
})
public class MyTestSuite_1 {
}
```



- MyTestSuite_2:

```
@Suite
@SelectClasses({
        TransactionTest.class,
        UserTest.class
})
public class MyTestSuite_2 {
}
```

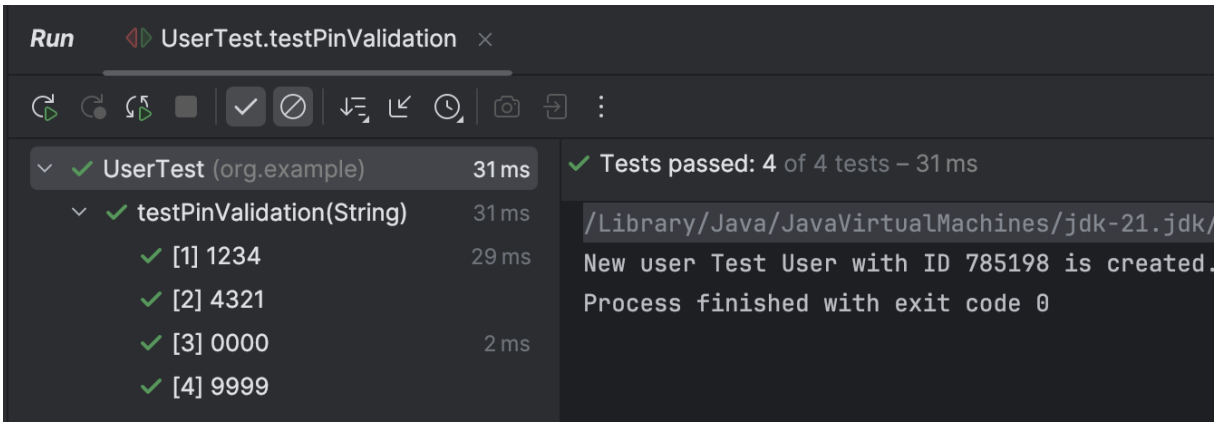


4. Parameterized test and outputs

- A test that runs multiple times with different input values.
- Ensures methods work for various cases without writing separate tests.

```java
@ParameterizedTest
@ValueSource(strings = {"1234", "4321", "0000", "9999"})
public void testPinValidation(String pin) {
    Bank bank = new Bank( name: "Test Bank");
    User user = new User( firstName: "Test",  lastName: "User", pin, bank);

    // User must be able to log in with the correct PIN.
    assertTrue(user.validatePin(pin)); // assertTrue()
    assertFalse(user.validatePin( apin: "WRONG_PIN"));
}
```

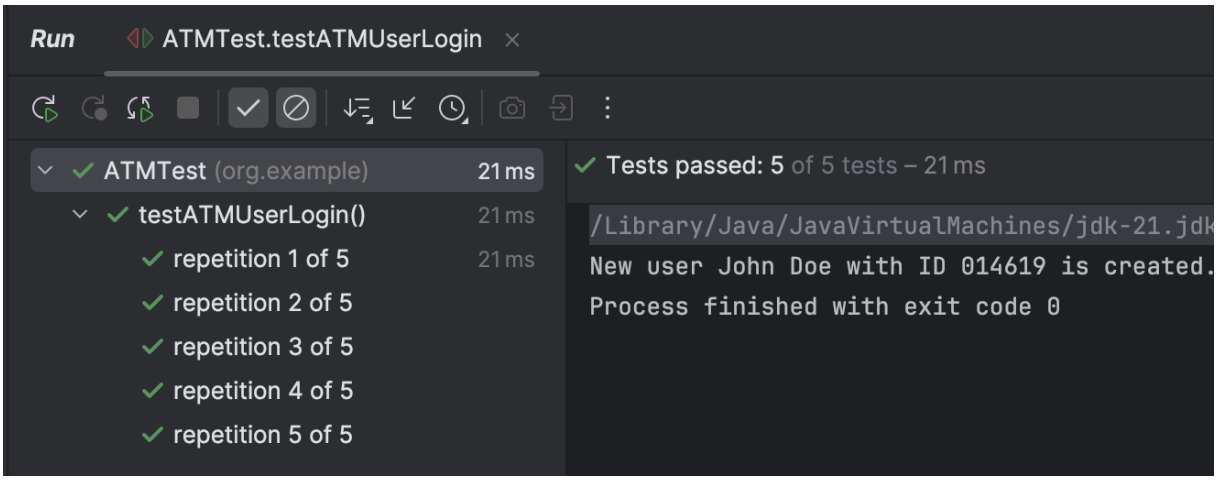


5. Repeatable test and outputs

- A test that always gives the same output when executed under the same conditions.
- Ensures consistency and reliability in testing.

```java
@RepeatedTest(5)  // Repeatable test (5 kez çalıştırılır)
public void testATMUserLogin() {
    Bank bank = new Bank( name: "Test Bank");
    User user = bank.addUser( firstName: "John",  lastName: "Doe",  pin: "1234");

    // Kullanıcı doğru bilgilerle giriş yapabilmeli.
    assertNotNull(bank.userLogin(user.getUUID(),  pin: "1234")); // assertNotNull()
}
```



Conclusion

This report presents the **UML class diagram**, **test cases** and **test results** of the project. It includes **assertion statements, parameterized tests, repeatable tests, and test suite executions**. The testing process confirms that all components of the ATM system function correctly according to the specified requirements.