HAYSTAQ

Uitwerking Demo Workshop Test Banking

EXPERTS IN AUTOMATED SOFTWARE TESTING

Haystaq
Huize Muyserick
Maurick 1
5261 NA Vught

T 06 53 77 94 92
E info@haystaq.nl
W www.haystaq.nl

NL33 RABO 0315 5905 13
BTW NL.8571.45.769.B.01
KvK 67715478

Uitwerking 1 | Login

Unit Test Code login

1. Login_ShouldReturnFalseWithMessage_WhenTooManyLogins

Arrange
database.userexists returns true
database.attempts returns 10
Act
result = login(“edeibert”, “12345”, out message)
Assert
result equals false
message equals “too many login attempts please wait”
database.increaseAttempts not called

2. Login_ShouldReturnFalseAndAttemptsIncreased_WhenLoginIncorrect

Arrange
database.userexists returns true
database.attempts returns 1
database.validUsernamePassword returns false
Act
result = login(“edeibert”, “12345”, out message)
Assert
result equals false
message equals “invalid username or password”
database.increaseAttempts(“edeibert”) called

~~*3. Login_ShouldReturnTrue_WhenLoginCorrect*~~ *Test not required*

~~**Arrange**~~
~~database.userexists returns true~~
~~database.attempts returns 1~~
~~database.validUsernamePassword returns true~~
~~**Act**~~
~~result = login(“edeibert”, “12345”, out message)~~
~~**Assert**~~
~~result equals true~~
~~message equals null~~
~~database.increaseAttempts not called~~

Uitwerking 2 | Transfer money

Unit Test Code transferMoney

1. transferMoney_ShouldReturnFalseWithMessage_WhenReceivingAccountNotExists

Arrange
database.getBalance returns 100
database.accountExists returns false
Act
result = transferMoney(“1111”, 10, “2222”, out message)
Assert
result equals false
message equals “receiving account is unknown”
database.transferFunds not called

2. transferMoney_ShouldReturnTrue_WhenTransferSucceeds

Arrange
database.getBalance returns 100
database.accountExists returns true
Act
result = transferMoney(“1111”, 10, “2222”, out message)
Assert
result equals true
message equals null
database.transferFunds(10, “1111”, “2222”)

3. transferMoney_ShouldReturnFalse_WhenNotEnoughFunds

Arrange
database.getBalance returns 5
database.accountExists returns true
Act
result = transferMoney(“1111”, 10, “2222”, out message)
Assert
result equals false
message equals “insufficient funds”
database.transferFunds not called

Opdracht 3 | Database

Unit Test Code userExists

1. userExists_ShouldReturnTrue_WhenOneUserIsRetrieved

Arrange

Make sure database contains edeibert user once.

Act

result = userExists(“edeibert”)

Assert

result equals true

2. userExists_ShouldReturnFalse_WhenNoUserIsRetrieved

Arrange

Make sure database does not contain edeibert user.

Act

result = userExists(“edeibert”)

Assert

result equals false

3. userExists_ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = userExists(“edeibert”)

Assert

Exception thrown

Unit Test Code attempts

1. attempts_ShouldReturnAttempts_WhenCalled

Arrange

Make sure database contains edeibert with attempts 2.

Act

result = attempts(“edeibert”)

Assert

result equals 2

2. attempts_ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = attempts(“edeibert”)

Assert

Exception thrown

Unit Test Code validUsernamePassword

1. validUsernamePassword_ShouldReturnTrue_WhenValidUsernamePassword

Arrange

Make sure database contains edeibert with password 12345.

Act

result = validUsernamePassword(“edeibert”, “12345”)

Assert

result equals true

2. validUsernamePassword _ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = validUsernamePassword(“edeibert”, “12345”)

Assert

Exception thrown

Unit Test Code increaseAttempts

1. increaseAttempts_ShouldIncreaseAttemptsInDatabase_WhenCalled

Arrange

Make sure database contains edeibert with attempts = 1.

Act

increaseAttempts("edeibert")

Assert

attempts("edeibert") equals 2

2. increaseAttempts _ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = increaseAttempts ("edeibert")

Assert

Exception thrown

Unit Test Code cleanAttempts

1. cleanAttempts_ShouldCleanAttempts_WhenMoreThanThreeAttemptsOlderThan60Minutes

Arrange

Make sure database contains edeibert with attempts = 4 and older than 60minutes.

Act

cleanAttempts()

Assert

attempts("edeibert") equals 0

2. cleanAttempts _ShouldNotCleanAttempts_WhenMoreThanThreeAttemptsLessThan60Minutes

Arrange

Make sure database contains edeibert with attempts = 4 and 30minutes.

Act

cleanAttempts()

Assert

attempts("edeibert") equals 4

3. cleanAttempts _ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

cleanAttempts()

Assert

Exception thrown

Unit Test Code substractFunds

1. substractFunds_ShouldSubstractFundsFromSender_WhenCalled

Arrange

Make sure database contains account “1111” with balance 100.

Make sure database contains account “2222” with balance 100.

Act

result = substractFunds(10, “1111”, “2222”)

Assert

getBalance(“1111”, 90)

2. substractFunds _ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = *substractFunds* (10, “1111”, “2222”)

Assert

Exception thrown

Unit Test Code addFunds

1. addFunds_ShouldAddFundsToReceiver_WhenCalled

Arrange

Make sure database contains account “1111” with balance 100.

Make sure database contains account “2222” with balance 100.

Act

result = addFunds(10, “1111”, “2222”)

Assert

getBalance(“2222”, 110)

2. addFunds _ShouldThrowException_WhenDatabaseThrowsException

Arrange

Make sure database is not running.

Act

result = *addFunds* (10, “1111”, “2222”)

Assert

Exception thrown