

**Documento de Testes de Software**

**Realizado por:**

Abílio Castro – 8170054

Ricardo Cardoso – 8170278

Vitor Santos – 8170312

Índice

[**1.** **Fase de Planeamento** 3](#_Toc530431556)

[1.1. Generalização do Projeto 3](#_Toc530431557)

[1.2. Recursos Utilizados 3](#_Toc530431558)

[1.3. Calendário de Atividades 3](#_Toc530431559)

[**2.** **Planeamento de Tarefas** 4](#_Toc530431560)

[2.1. Generalização do Plano de Testes 4](#_Toc530431561)

[2.2. Áreas do Software a Abordar 4](#_Toc530431562)

[2.3. Recursos a Utilizar para Execução de Teste 4](#_Toc530431563)

[**3.** **Determinar Funcionalidades a Serem Testadas** 5](#_Toc530431564)

[3.1. Seleção e informação das funcionalidades a serem testadas 5](#_Toc530431565)

[3.2. Análises, Procedimentos , Testes e Conclusões Acerca dos Métodos Selecionados 6](#_Toc530431566)

[3.2.1. getBicycle() 6](#_Toc530431567)

[3.2.2. returnBikycle() 8](#_Toc530431568)

[3.2.3. bicycleRentalFee() 13](#_Toc530431569)

[3.2.4. addCredit() 19](#_Toc530431570)

[3.2.5. registerUser() 23](#_Toc530431571)

[3.2.6. verifyCredit() 27](#_Toc530431572)

[**4.** **Repositório GitHub** 30](#_Toc530431573)

[**5.** **Bibliografia** 30](#_Toc530431574)

# **Fase de Planeamento**

## Generalização do Projeto

Neste projeto realizado no âmbito da unidade curricular de **Engenharia de Software II**, temos como objetivo avaliar e testar um software de aluguer de bicicletas.

## Recursos Utilizados

**Numa fase inicial serão utilizados:**

Microsoft Office Word – Elaboração do Documento de Testes de Software;

Microsoft Office Excel / Google Sheets – Desenvolvimento de tabelas ECP, BVA e Test Cases;

JetBrains IntelliJ / Gradle / JUnit 5 – Execução de testes;

## Calendário de Atividades

**12/11/2018** – Início do planeamento de atividades, avaliação dos recursos necessários, criação do repositório no GitHub e configuração do Gradle;

**14/11/2018** – Distribuição de tarefas;

**16/11/2018** – Elaboração de tabelas ECP;

**17/11/2018** – Finalização de tabelas ECP, elaboração de tabelas BVA e Test Cases;

**18/11/2018** – Finalização de tabelas BVA e Test Cases, início dos testes de caixa preta e início da elaboração do Documento de Testes de Software;

**19/11/2018** – Finalização dos testes de caixa preta e finalização do Documento de Testes de Software (#1 Milestone);

# **Planeamento de Tarefas**

## Generalização do Plano de Testes

Numa fase inicial elaboramos tabelas de **Equivalence Class Partitioning**, **Boundary Value Analysis** e de **Test Cases** para os métodos selecionados;

De seguida, tendo em conta os métodos selecionados, executaremos **Testes de Caixa Preta** tendo em conta outputs esperados para certos inputs. Sendo que se corresponderem ao esperado serão aprovados, caso contrário serão reprovados;

## Áreas do Software a Abordar

Certos **métodos** do software serão submetidos a vários **testes**, sendo os quais:

getBicycle();

returnBikycle();

bikeRentalFee();

addCredit();

registerUser();

verifyCredit();

## Recursos a Utilizar para Execução de Teste

**Para execução de testes serão utilizados:**

JetBrains IntelliJ 2018.2.5. (Ultimate Edition);

Gradle Build Tool;

JUnit 5 Test Tool;

# **Determinar Funcionalidades a Serem Testadas**

## Seleção e informação das funcionalidades a serem testadas

Tendo em conta os requisitos funcionais do software, serão submetidos a testes, métodos específicos com o objetivo de verificar a coerência, qualidade e fiabilidade de certas funcionalidades. Para isso foram selecionados os seguintes:

**3.2.1. getBicycle()** – Método que permite alugar uma bicicleta a um utilizador;

**3.2.2. returnBikycle()** – Método que permite devolver uma bicicleta alugada por um utilizador;

**3.2.3. bicycleRentalFee()** – Método que calcula a quantia a pagar pelo aluguer de uma bicicleta;

**3.2.4. addCredit()** – Método que permite adicionar fundos à carteira de um utilizador;

**3.2.5. registerUser()** – Método que permite registar um utilizador;

**3.2.6. verifycredit()** – Método que verifica se um utilizador tem os fundos mínimos suficientes para alugar uma bicicleta;

## Análises, Procedimentos , Testes e Conclusões Acerca dos Métodos Selecionados

### getBicycle()

Método que permite alugar uma bicicleta a um utilizador;

#### ECP e BVA

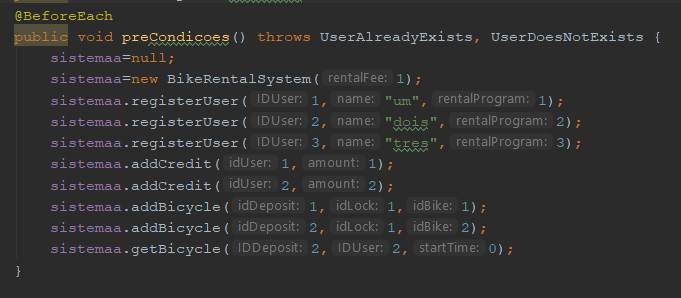
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING getBicycle()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 3 | !=3 |
| **INPUT TYPES** | IDDeposit: Inteiro | IDDeposit: != Inteiro |
| IDUser: Inteiro | IDUser: != Inteiro |
| starttime: Inteiro | starttime: != Inteiro |
| **SPECIFIC X VALUE** | IDDeposit: >=0 | IDDeposit: <0 |
| IDUser: >0 | IDUser: <=0 |
| starttime: >=0 | starttime: <0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS getBicycle()** | | | | | | | | |
|  | **LOWER BOUNDRY** | | | | **UPPER BOUNDRY** | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** |  | **VALID**  **PARTITION BY BELOW THE BOUNDRY** |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| IDDeposit | <=0 |  | >0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |
| IDUser | <=0 |  | >0 |  | Num. of Users |  | NO MAX AMOUNT |  |
| endtime | <=0 |  | >0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |

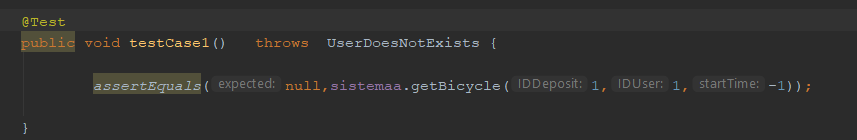
#### Test Cases

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES getBicycle()** | | | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | | | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| IDDeposit | IDUser | starttime |
| 1 | starttime inválido | -Utilizador,id=1,crédito=1,  semBike -Utilizador,id=2,credito=1,  comBike -Utilizador,id=3,crédito=0,  comBike -Deposito,id=1,comBike -Deposito,id=2,semBike | 1 | 1 | -1 | Não entra no método | 1 | Failed |
| 2 | starttime válido | 1 | 1 | 0 | Diz tempo inicial e return 1 | 1 | Passed |
| 3 | starttime válido | 1 | 1 | 1 | Diz tempo inicial e return 1 | 1 | Passed |
| 4 | IDUser inválido | 1 | 0 | 0 | Não entra no método | Exceptions UserDoesNotExists | Failed |
| 5 | IDUser válido | 1 | 1 | 0 | Diz tempo inicial e return 1 | 1 | Passed |
| 6 | IDUser válido já tem aluguer ativo | 1 | 2 | 0 | return -1 (já tem bike) | 1 | Failed |
| 7 | IDUser válido não tem crédito | 1 | 3 | 0 | return -1 (não tem crédito) | -1 | Passed |
| 8 | IDUser válido não existente | 1 | 4 | 0 | exceção user nao existe | Exceptions UserDoesNotExists | Passed |
| 9 | IDDeposit inválido | 0 | 1 | 0 | Não entra no método | -1 | Failed |
| 10 | IDDeposit válido | 1 | 1 | 0 | Diz tempo inicial e return 1 | 1 | Passed |
| 11 | IDDeposit válido não tem Bike | 2 | 1 | 0 | return -1 (deposito sem bike) | -1 | Passed |
| 12 | IDDeposit válido não existente | 3 | 1 | 0 | return -1 (deposito nao existe) | -1 | Passed |

Testes

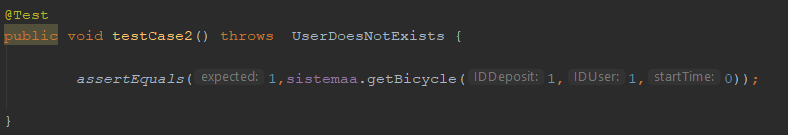


Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=1, depois é adicionado um utilizador com id=1,name = “um” e rentalProgram = 1, um utilizador com id=2,name=”dois”,rentalProgram=2,um utilizador com id=3,name=”tres”,rentalProgram=3, é adicionado crédito ao idUser = 1 e ao idUser=2, é adicionado duas bicicletas, uma no idDeposit=1 com lock =1, outra com idDeposit=2 e lock=1, fazemos o getBicycle com idDeposit=2 para o idUser=2, para o utilizador já ter uma bicicleta.



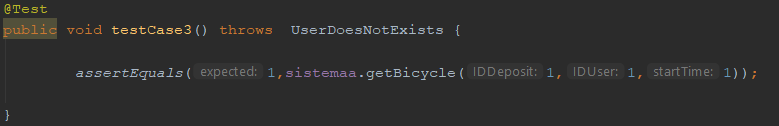
TestCase1: neste teste tentamos enviar para o método getBicycle um startTime inválido, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor 1.

Test Status: Failed



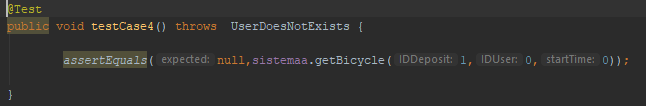
TestCase2: neste teste enviamos tudo correto para o método getBicycle, o resultado esperado é que o metodo retorne 1, e é o resultado obtido.

Test Status: Passed



TestCase3: neste teste enviamos tudo correto para o método getBicycle, o resultado esperado é que o metodo retorne 1, e é o resultado obtido.

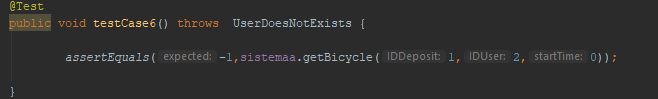
Test Status: Passed



TestCase4: neste teste tentamos enviar para o método getBicycle um idUser inválido, o resultado esperado é que o método não seja executado mas na verdade o resultado obtido é uma exceção UserDoesNotExists

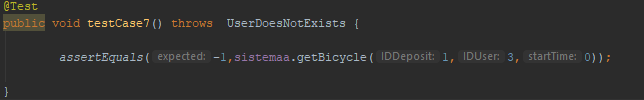
Test Status: Failed

TestCase5 igual ao TestCase2.



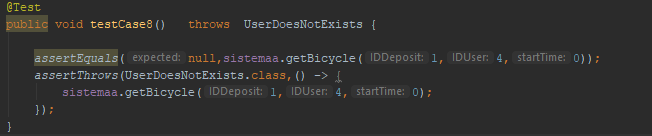
TestCase6: neste teste tentamos enviar para o método getBicycle um idUser válido mas com aluguer ativo, o resultado esperado é que retorne -1 mas na verdade o resultado obtido é 1.

Test Status: Failed



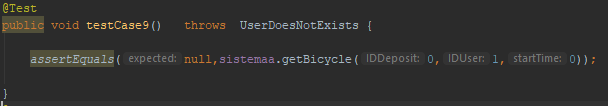
TestCase7: neste teste tentamos enviar para o método getBicycle um idUser válido mas sem crédito o resultado esperado é que retorne -1 e o resultado obtido é -1.

Test Status: Passed



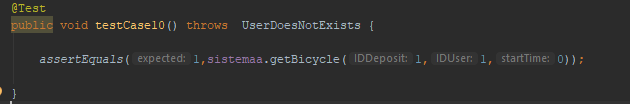
TestCase8: neste teste tentamos enviar para o método getBicycle um idUser válido mas que não existe o resultado esperado é que exceção UserDoesNotExiste e o resultado obtido é exceção UserDoesNotExiste

Test Status: Passed



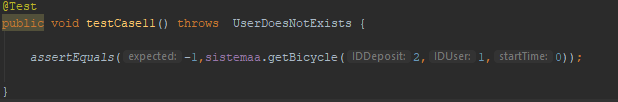
TestCase9: neste teste tentamos enviar para o método getBicycle um idDeposit inválido resultado esperado é que exceção não entre no método e o resultado obtido é o retorno -1

Test Status: Failed



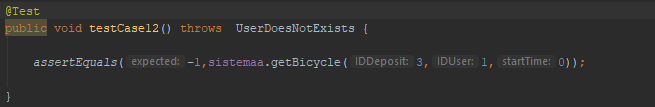
TestCase10: neste teste tentamos enviar para o método getBicycle um idDeposit válido resultado esperado é 1 não entre no método e o resultado obtido é o retorno 1

Test Status: Passed



TestCase11: neste teste tentamos enviar para o método getBicycle um idDeposit válido mas sem espaço resultado esperado é -1 não entre no método e o resultado obtido é o retorno -1

Test Status: Passed



TestCase11: neste teste tentamos enviar para o método getBicycle um idDeposit válido mas não existente resultado esperado é -1 não entre no método e o resultado obtido é o retorno -1

Test Status: Passed

### returnBikycle()

Método que permite devolver uma bicicleta alugada por um utilizador;

#### ECP e BVA

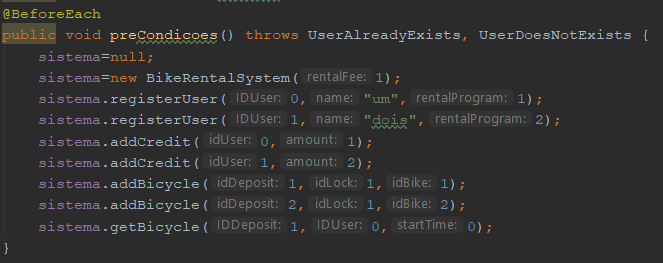
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING returnBikycle()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 3 | !=3 |
| **INPUT TYPES** | IDDeposit: Inteiro | IDDeposit: != Inteiro |
| IDUser: Inteiro | IDUser: != Inteiro |
| endtime: Inteiro | endtime: != Inteiro |
| **SPECIFIC X VALUE** | IDDeposit: >=0 | IDDeposit: <0 |
| IDUser: >0 | IDUser: <=0 |
| endtime: >=0 | endtime: <0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS returnBikycle()** | | | | | | | | |
|  | **LOWER BOUNDRY** | | | | **UPPER BOUNDRY** | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** |  | **VALID**  **PARTITION BY BELOW THE BOUNDRY** |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| rentalProgram | <=0 |  | >0 |  | <3 |  | >=3 |  |
| initTime | <=0 |  | >0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |
| endtime | <=0 |  | >0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |
| nRentals | <0 |  | >=0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES returnBikycle()** | | | | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | | | | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| rentalProgram | nRentals | initTime | endTime |
| 1 | endTime inválido | -RentalFee=3 | 1 | 0 | 0 | -1 | Não entra no método | -3 | Failed |
| 2 | endTime válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 3 | endTime válido | 1 | 0 | 0 | 1 | (1 - 0) \* 3 = 3 | 3 | Passed |
| 4 | initTime inválido | 1 | 0 | -1 | 0 | Não entra no método | 3 | Failed |
| 5 | initTime válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 6 | initTime válido mas maior que endTime | 1 | 0 | 1 | 0 | Não entra no método | -3 | Failed |
| 7 | nRentals inválido | 1 | -1 | 0 | 0 | Não entra no método | 0 | Failed |
| 8 | nRentals válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 9 | nRentals válido | 1 | 1 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 10 | rentalProgram inválido | 0 | 0 | 0 | 0 | Não entra no método | 0 | Failed |
| 11 | rentalProgram válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 12 | rentalProgram válido mas sem nRentals | 2 | 0 | 0 | 0 | 0 | 0 | Passed |
| 13 | rentalProgram válido | 2 | 1 | 0 | 0 | 3 \* (0 - 0) = 0 | 0 | Passed |
| 14 | rentalProgram válido | 2 | 1 | 0 | 11 | 10 \* 3 + ((11-0) - 10) \* 3/2 = 31 | 31 | Passed |
| 15 | rentalProgram inválido | 3 | 0 | 0 | 0 | 0 | 0 | Passed |

#### Test Cases

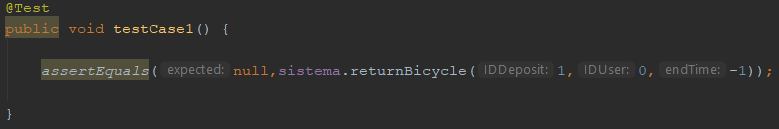
#### Testes



Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=3, depois é adicionado um utilizador com id=0,name = “um” e rentalProgram = 1 e outro com id=1, name=”dois” e rentalProgram =2.

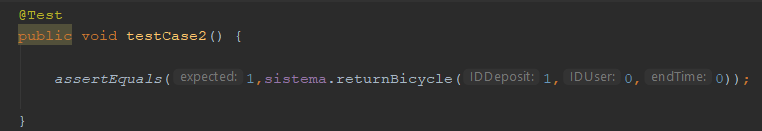
Ao utilzador com id = 1 são adicionados créditos na conta com amount=2 e ao utilizador com id = 0 são adicionados créditos na conta com amount = 1.

De seguida são adicionadas duas bicicletas em depositos diferentes e é feito o get da bicicleta guardada no depósito 1 para o o utilizador com id=0.



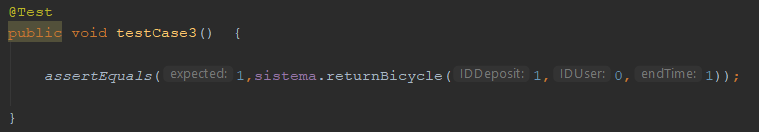
TestCase1: neste teste enviamos para o método returnBicycle() um endTime inválido = -1, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor 1.

Test Status: Failed



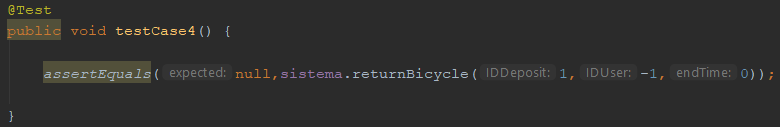
TestCase2: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 1, IDUser = 0 ,endTime = 0), o resultado esperado é o valor 1.

Test Status: Passed



TestCase3: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 1, IDUser = 0 ,endTime = 1), o resultado esperado é o valor 1.

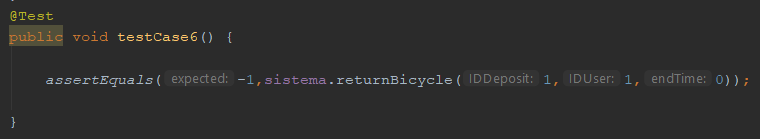
Test Status: Passed



TestCase4: neste teste enviamos para o método returnBicycle() um IDUser inválido = -1, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor -1.

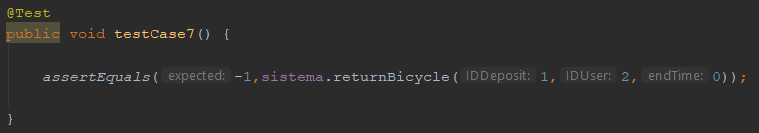
Test Status: Failed

TestCase5: Semelhante ao TestCase2



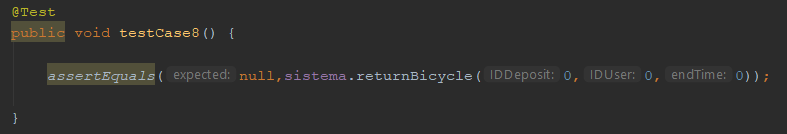
TestCase6: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 1, IDUser = 1 ,endTime = 0), o utilizador neste caso não tem uma bicicleta associada, o resultado esperado é o valor -1.

Test Status: Passed



TestCase7: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 1, IDUser = 2 ,endTime = 0) mas o utilizador não está registado no sistema, o resultado esperado é valor -1.

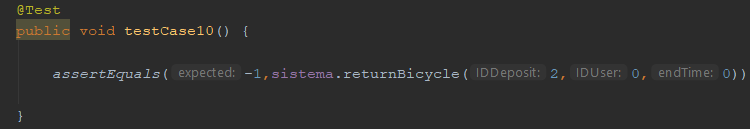
Test Status: Passed



TestCase8: neste teste enviamos para o método returnBicycle() um IDDeposit inválido = 0, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor -1.

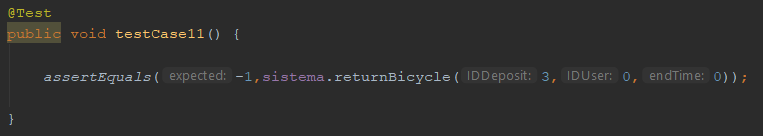
Test Status: Failed

TestCase9: Semelhante ao TestCase2.



TestCase10: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 2, IDUser = 0 ,endTime = 0), mas o depósito não tem espaço, o resultado esperado é o valor -1.

Test Status: Passed



TestCase11: neste teste todas as variáveis enviadas para o método são válidas(IDDeposit= 3, IDUser = 0 ,endTime = 0), mas o depósito não, o resultado esperado é o valor -1.

Test Status: Passed

### bicycleRentalFee()

Método que calcula a quantia a pagar pelo aluguer de uma bicicleta;

#### ECP e BVA

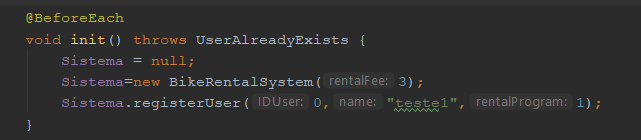
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING bicycleRentalFee()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 4 | !=4 |
| **INPUT TYPES** | rentalProgram: Inteiro | rentalProgram: !=Inteiro |
| initTime: Inteiro | initTime: !=Inteiro |
| endTime: Inteiro | endTime: !=Inteiro |
| nRentals: Inteiro | nRentals: !=Inteiro |
| **SPECIFIC X VALUE** | rentalProgram: (1 ou 2) | rentalProgram: != (1 ou 2) |
| initTime: >=0 | initTime: <0 |
| endTime: >=0 | endTime: <0 |
| nRentals: >=0 | nRentals: <0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS bicycleRentalFee()** | | | | | | | | | |
|  | **LOWER BOUNDRY** | | | | | **UPPER BOUNDRY** | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** |  | **VALID PARTITION BY BELOW THE BOUNDRY** | |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| rentalProgram | <=0 |  | >0 |  | <3 | |  | >=3 |  |
| initTime | <=0 |  | >0 |  | NO MAX AMOUNT | |  | NO MAX AMOUNT |  |
| endtime | <=0 |  | >0 |  | NO MAX AMOUNT | |  | NO MAX AMOUNT |  |
| nRentals | <0 |  | >=0 |  | NO MAX AMOUNT | |  | NO MAX AMOUNT |  |

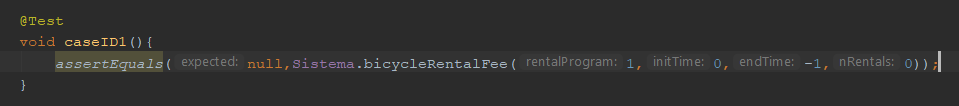
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES bicycleRentalFee()** | | | | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | | | | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| rentalProgram | nRentals | initTime | endTime |
| 1 | endTime inválido | -RentalFee=3 | 1 | 0 | 0 | -1 | Não entra no método | -3 | Failed |
| 2 | endTime válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 3 | endTime válido | 1 | 0 | 0 | 1 | (1 - 0) \* 3 = 3 | 3 | Passed |
| 4 | initTime inválido | 1 | 0 | -1 | 0 | Não entra no método | 3 | Failed |
| 5 | initTime válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 6 | initTime válido mas maior que endTime | 1 | 0 | 1 | 0 | Não entra no método | -3 | Failed |
| 7 | nRentals inválido | 1 | -1 | 0 | 0 | Não entra no método | 0 | Failed |
| 8 | nRentals válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 9 | nRentals válido | 1 | 1 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 10 | rentalProgram inválido | 0 | 0 | 0 | 0 | Não entra no método | 0 | Failed |
| 11 | rentalProgram válido | 1 | 0 | 0 | 0 | (0 - 0) \* 3 = 0 | 0 | Passed |
| 12 | rentalProgram válido mas sem nRentals | 2 | 0 | 0 | 0 | 0 | 0 | Passed |
| 13 | rentalProgram válido | 2 | 1 | 0 | 0 | 3 \* (0 - 0) = 0 | 0 | Passed |
| 14 | rentalProgram válido | 2 | 1 | 0 | 11 | 10 \* 3 + ((11-0) - 10) \* 3/2 = 31 | 31 | Passed |
| 15 | rentalProgram inválido | 3 | 0 | 0 | 0 | 0 | 0 | Passed |

#### Test Cases

#### Testes

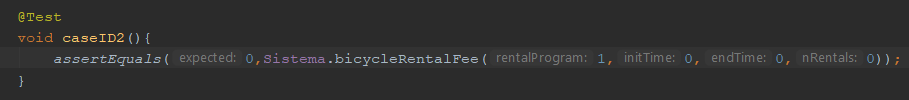


Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=3, depois é adicionado um utilizador com id=0,name = “teste1” e rentalProgram = 1.



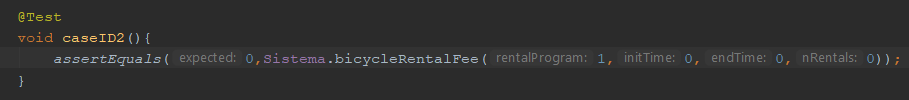
TestCase1: neste teste tentamos enviar para o método bicycleRentalFee() um endTime inválido, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor -3.

Test Status: Failed



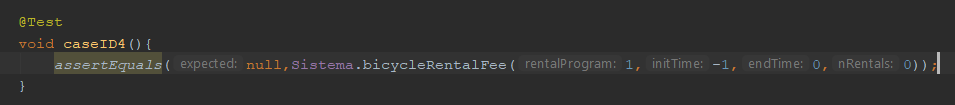
TestCase2: neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 1, initTime = 0, endTime = 0, nRentals = 0), o resultado esperado é o valor 0.

Test Status: Passed



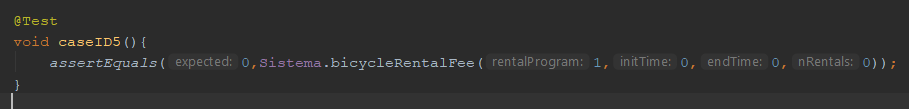
TestCase3: neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 1, initTime = 0, endTime = 1, nRentals = 0), o resultado esperado é o valor 3.

Test Status: Passed

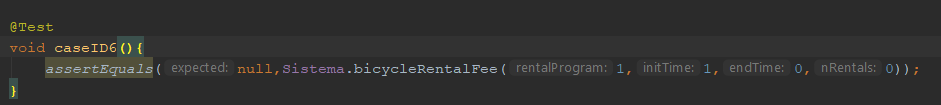


TestCase4: neste teste enviamos para o método bicycleRentalFee() um initTime inválido, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor 3.

Test Status: Failed

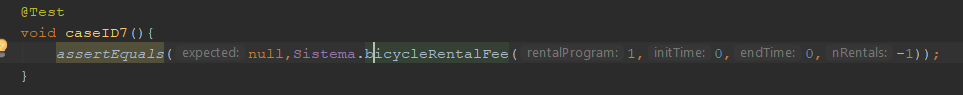


TestCase5: semelhante ao TestCase2.



TestCase6: neste teste é enviado para o método bicycleRentalFee() um initTime superior ao endTime (condição inválida), o resultado esperado é que o método não seja executado mas na verdade é retornado o valor -3.

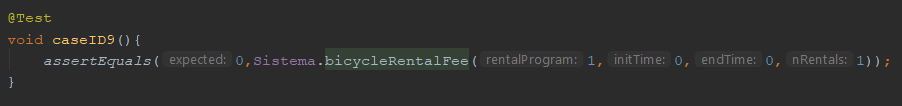
Test Status: Failed



TestCase7: neste teste é enviado para o método bicycleRentalFee() um nRentals inválido = -1, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor 0;

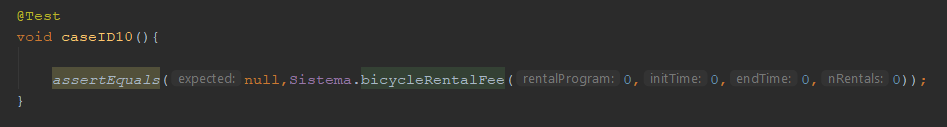
Test Status: Failed

TestCase8: semelhante ao TestCase2.



TestCase9: neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 1, initTime = 0, endTime = 0, nRentals = 1), o resultado esperado é o valor 0.

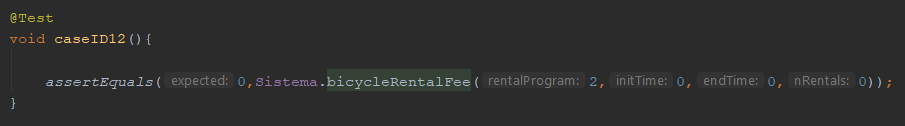
Test Status: Passed



TestCase10: neste teste é enviado para o método bicycleRentalFee() um rentalProgram inválido = 0, o resultado esperado é que o método não seja executado mas na verdade é retornado o valor 0;

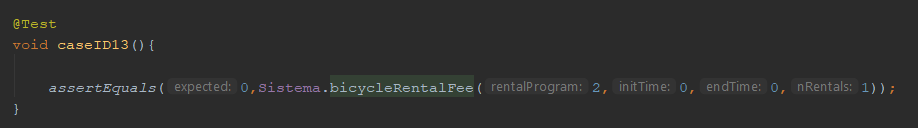
Test Status: Failed

TesteCase11: semelhante ao TestCase2.



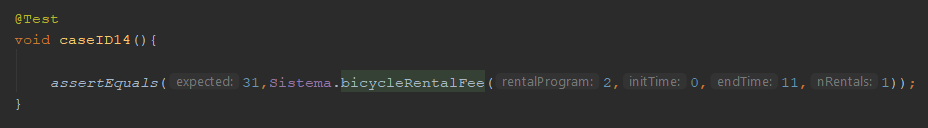
TestCase12:neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 2, initTime = 0, endTime = 0, nRentals = 0), o resultado esperado é o valor 0.

Test Status: Passed



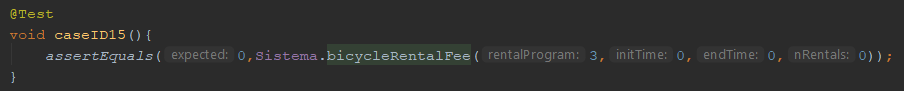
TestCase13: neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 2, initTime = 0, endTime = 0, nRentals = 1), o resultado esperado é o valor 0.

Test Status: Passed



TestCase14: neste teste todas as variáveis enviadas para o método são válidas( rentalProgram = 2, initTime = 0, endTime = 11, nRentals = 1), o resultado esperado é o valor 31.

Test Status: Passed



TestCase15: neste teste é enviado para o método bicycleRentalFee() um rentalProgram que é válido mas não existe e o resultado esperado é o valor 0.

Test Status: Passed

### addCredit()

Método que permite adicionar fundos à carteira de um utilizador;

#### ECP e BVA

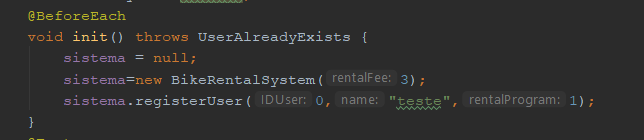
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING addCredit()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 2 | !=2 |
| **INPUT TYPES** | IDUser: Inteiro | IDUser: != Inteiro |
| amount: Inteiro | amount: != Inteiro |
| **SPECIFIC X VALUE** | IDUser: >=0 | IDUser: <0 |
| amount: >0 | amount: <1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS addCredit()** | | | | | | | | |
|  | **LOWER BOUNDRY** | | | | **UPPER BOUNDRY** | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** |  | **VALID PARTITION BY BELOW THE BOUNDRY** |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| IDUser | <=0 |  | >0 |  | Num. of Users |  | NO MAX AMOUNT |  |
| amount | <=0 |  | >0 |  | NO MAX AMOUNT |  | NO MAX AMOUNT |  |

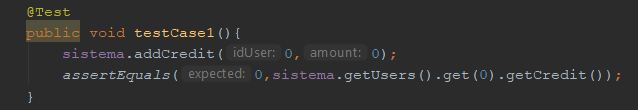
#### Test Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES addCredit()** | | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| IDUser | amount |
| 1 | amount não válido | Apenas um User registado com  id = 0 | 0 | 0 | Não entra no método |  |  |
| 2 | amount válido | 0 | 1 | O crédito foi adicionado = 1 | 1 | Passed |
| 3 | amount válido | 0 | 2 | O crédito foi adicionado = 2 | 2 | Passed |
| 4 | IDUser não válido | -1 | 1 | Não entra no método |  |  |
| 5 | IDUser válido | 0 | 1 | O crédito foi adicionado = 1 | 1 | Passed |
| 6 | IDUser válido | 1 | 1 | IndexOutOfBoundsException | IndexOutOfBoundsException | Passed |

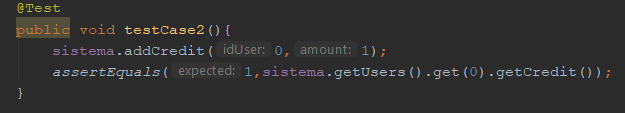
#### Testes



Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=3, depois é adicionado um utilizador com id=0,name = “teste” e rentalProgram = 1.

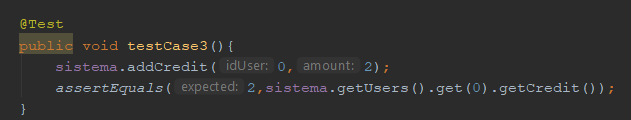


TestCase1: neste teste é adicionado a um utilizador já existente um amount inválido, os resultados esperados são que o método não seja executado e não seja adicionado créditos á conta do utilizador



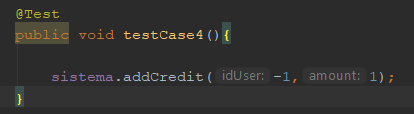
TestCase2: neste teste é adicionado a um utilizador já existente um amount válido = 1, os resultados esperados é que o crédito seja adicionado e que o balanço da conta do utilizador seja 1.

Test Status: Passed



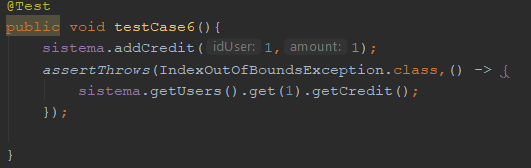
TestCase3: neste teste é adicionado a um utilizador já existente um amount válido = 2, os resultados esperados é que o crédito seja adicionado e que o balanço da conta do utilizador seja 2.

Test Status: Passed



TestCase4: neste teste é enviado para o método addCredit() um idUser inválido, o resultado esperado é que o método não seja executado.

TestCase5: semelhante ao TestCase 2.



TestCase6: neste teste tentamos adicionar a crédito a um utilizador não existente, o resultado esperado é uma exceção do tipo “IndexOutOfBoundsException”.

Test Status: Passed

### registerUser()

Método que permite registar um utilizador;

#### ECP e BVA

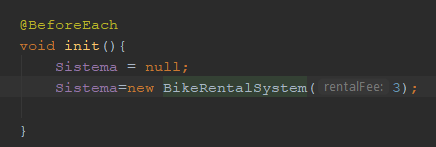
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING registerUser()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 3 | !=3 |
| **INPUT TYPES** | IDUser: Inteiro | IDUser: != Inteiro |
| name: String | name: != String |
| rentalProgram: Inteiro | rentalProgram: != Inteiro |
| **SPECIFIC X VALUE** | IDUser: >=0 | IDUser: <0 |
| name: "teste" | name: null |
| rentalProgram: 1 ou 2 | 4 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS registerUser()** | | | | | | | | | |
|  | **LOWER BOUNDRY** | | | **UPPER BOUNDRY** | | | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** | |  | **VALID PARTITION BY BELOW THE BOUNDRY** |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| IDUser | <=0 |  | >0 | |  | Num. of Users |  | NO MAX AMOUNT |  |
| name |  |  |  | |  |  |  |  |  |
| rentalProgram | <=0 |  | >0 | |  | <3 |  | >=3 |  |

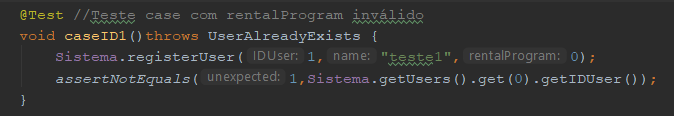
#### Test Cases

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES registerUser()** | | | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | | | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| IDUser | name | rentalProgram |
| 1 | rentalProgram não válido | Apenas um User registado com ID = 0 | 1 | "Utilizador1" | 0 | Não entra no metodo | O utilizador é adicionado com rentalProgram inválido | Failed |
| 2 | rentalProgram válido | 1 | "Utilzador1" | 1 | O utilizador é adicionado | O utilizador é adicionado | Passed |
| 3 | rentalProgram Válido | 1 | "Utilizador1" | 2 | O utilizador é adicionado | O utilizador é adicionado | Passed |
| 4 | rentalProgram não válido | 1 | "Utilizador1" | 3 | Não entra no metodo | O utilizador é adicionado com rentalProgram inválido | Failed |
| 5 | name não válido | 1 | null | 1 | Não entra no metodo | O utilizador é adicionado com name inválido | Failed |
| 6 | name válido | 1 | "Utilizador1" | 1 | O utilizador é adicionado | O utilizador é adicionado | Passed |
| 7 | IDUser não válido | -1 | "Utilizador1" | 1 | não entra no metodo | O utilizador é adicionado com IDUser inválido | Failed |
| 8 | IDUser válido registado | 0 | "Utilizador1" | 1 | Exception UserAlreadyExists | Exception UserAlreadyExists | Passed |

#### Testes

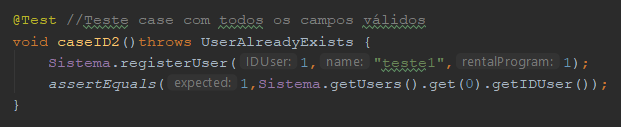


Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=3



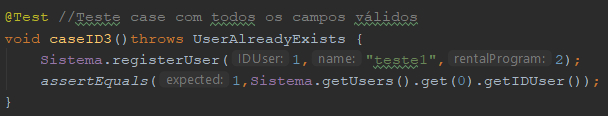
TestCase1: Neste teste é enviado para o método registerUser() um rentalProgram inválido = 0, resultado esperado é que o método não seja executado mas na verdade o utilizador é adicionado ao sistema com um rentalProgram inválido

Test Status: Failed



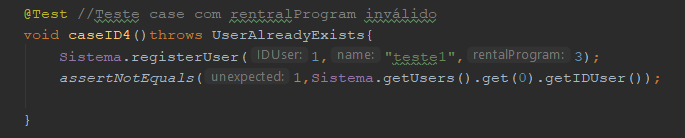
TestCase2: neste teste todas as variáveis enviadas para o método são válidas( IDUser = 1, name=”teste1”,rentalProgam=1), o resultado esperado é que o utilizador seja adicionado ao sistema com sucesso.

Test Results: Passed



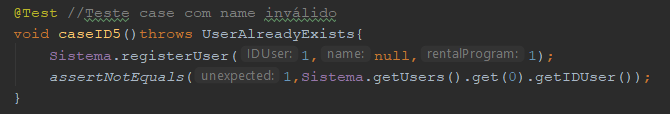
TestCase3: neste teste todas as variáveis enviadas para o método são válidas( IDUser = 1, name=”teste1”,rentalProgam=2), o resultado esperado é que o utilizador seja adicionado ao sistema com sucesso.

Test Results: Passed



TestCase4: Neste teste é enviado para o método registerUser() um rentalProgram inválido = 3,o resultado esperado é que o método não seja executado mas na verdade o utilizador é adicionado ao sistema com um rentalProgram inválido.

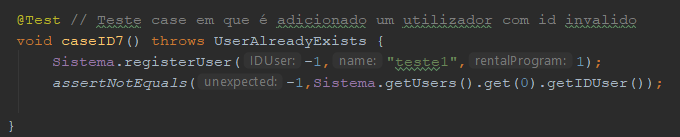
Test Status: Failed



TestCase5: Neste teste é enviado para o método registerUser() um name inválido = null,o resultado esperado é que o método não seja executado mas na verdade o utilizador é adicionado ao sistema com um name inválido.

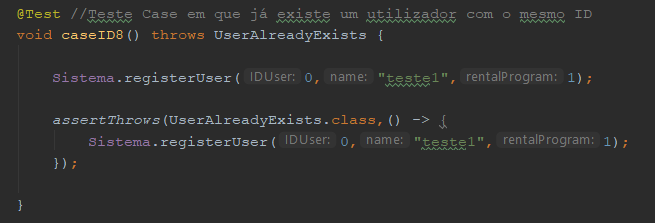
Test Results: Failed

TestCase6: Semelhante ao TestCase2



TestCase7: Neste teste é enviado para o método registerUser() um IDUser inválido = -1,o resultado esperado é que o método não seja executado mas na verdade o utilizador é adicionado ao sistema com um IDUser inválido.

Test Status: Failed



TestCase8: Neste teste tentamos adicionar um utilizador com um IDUser igual ao de um utilizador já registado no sistema, o resultado esperado é um exceção do tipo “UserAlreadyExists”.

Test Status: Passed

### verifyCredit()

Método que verifica se um utilizador tem os fundos mínimos suficientes para alugar uma bicicleta;

#### ECP e BVA

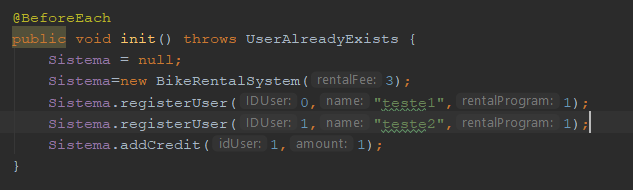
|  |  |  |
| --- | --- | --- |
| **EQUIVALENCE CLASS PARTITIONING verifyCredit()** | | |
| **REQUIREMENTS** | **VALID CLASS** | **INVALID CLASS** |
| **Nº INPUTS** | 1 | !=1 |
| **INPUT TYPES** | IDUser: Inteiro | IDUser: != Inteiro |
| **SPECIFIC X VALUE** | IDUser: >=0 | IDUser: <0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BOUNDARY VALUE ANALYSIS** | | | | | | | | | |
|  | **LOWER BOUNDRY** | | | **UPPER BOUNDRY** | | | | | |
| **VARIABLES** | **INVALID PARTITION BY BELOW THE BOUNDRY** |  | **VALID PARTITION BY ABOVE THE BOUNDRY** | |  | **VALID PARTITION BY BELOW THE BOUNDRY** |  | **INVALID PARTITION BY ABOVE THE BOUNDRY** |  |
|
| IDUser | <=0 |  | >0 | |  | Num. of Users |  | NO MAX AMOUNT |  |
| name |  |  |  | |  |  |  |  |  |
| rentalProgram | <=0 |  | >0 | |  | <3 |  | >=3 |  |

#### Test Cases

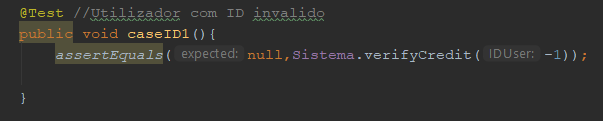
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST CASES** | | | | | | |
| **TEST CASE ID** | **TEST CASE** | **PRE-CONDITIONS** | **VARIABLES** | **EXPECTED RESULTS** | **ACTUAL RESULTS** | **Test Status** |
| IDUser |
| 1 | IDUser não válido | -utilizador,id=0,credit=0 -utilizador,id=1,credit=1 | -1 | Não entra no método | Return falso | Failed |
| 2 | IDUser válido registado | 0 | Retun falso | Return falso | Passed |
| 3 | IDUser válido registado | 1 | Return verdadeiro | Return verdadeiro | Passed |
| 4 | IDUser válido não registado | 2 | Return falso | Returb falso | Passed |

#### Testes



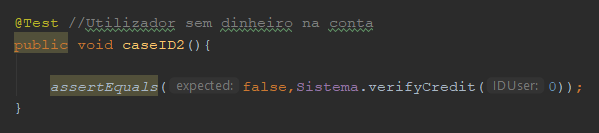
Este bloco representa as intruções executadas antes de cada teste, é instanciada uma variavél do tipo sistema com o rentalFee=3, depois é adicionado um utilizador com id=0,name = “teste1” e rentalProgram = 1 e outro com id=1, name=”teste2” e rentalProgram =1.

Ao utilzador com id = 1 são adicionados créditos na conta enquando que o utilizador com id = 0 tem a conta com 0 créditos.



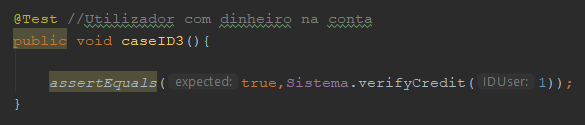
TestCase1: neste teste é enviado para o método verifyCredit() um IDUser inválido = -1, o resultado esperado é que o método não seja executado mas na verdade é retornado False.

Test Status: Failed



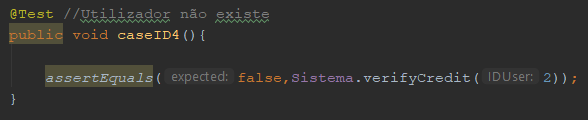
TestCase2: neste teste é enviado para o método verifyCredit() um IDUser de um utilizador que não tem créditos na conta, o resultado esperado é um false.

Test Status: Passed



TestCase3: neste teste é enviado para o método verifyCredit() um IDUser de um utilizador que tem créditos na conta, o resultado esperado é um true.

Test Status: Passed



TestCase4: neste teste é enviado para o método verifyCredit() um IDUser de um utilizador que não existe, o resultado esperado é um false.

Test Status: Passed

# **Repositório GitHub**

**https://github.com/abiliomg/Cykel**

# **Bibliografia**

<https://moodle.estg.ipp.pt/course/view.php?id=2408>

<https://www.guru99.com/software-testing.html>

<https://moodle.estg.ipp.pt/pluginfile.php/88661/mod_resource/content/1/1008-1987%20%20-%20%20IEEE%20Standard%20for%20Software%20Unit%20Testing.pdf>