Relatório Part.1

Engenharia de Software II

Gabriel Pereira 8150115

Nuno Josefino 8150189

Rafael Vieira 8150448



# Índice

[Índice 2](#_Toc530439333)

[Introdução 5](#_Toc530439334)

[Metódo getBicycle 6](#_Toc530439335)

[Tabela ECP 6](#_Toc530439336)

[Tabela BVA 6](#_Toc530439337)

[Casos de teste 7](#_Toc530439338)

[Caso de teste 0 (testGetBicycleUSERMENOSUM) 7](#_Toc530439339)

[Caso de teste 1/3/6 (testGetBicycleEXISTEBICICLETA) 7](#_Toc530439340)

[Caso de teste 4 (testGetBicucleSEMBICICLETASDISP) 8](#_Toc530439341)

[Caso de teste 4 (testGetBicycleNAOEXISTEDEPOSITO) 8](#_Toc530439342)

[Caso de teste 5 (testGetBicycleSTARTTIMEMENOSUM) 9](#_Toc530439343)

[Caso de teste 7 (testGetBicycleSTARTUM) 9](#_Toc530439344)

[Metódo returnBikycle 10](#_Toc530439345)

[Tabela ECP 10](#_Toc530439346)

[Tabela BVA 10](#_Toc530439347)

[Caso de teste 1 (testReturnBicycleRETORNASALDOUSERUM) 11](#_Toc530439348)

[Caso de teste 2 (testReturnBicycleRETORNASALDOUSEROZERO) 11](#_Toc530439349)

[Caso de teste 3 (testReturnBicycleUSERNAOEXISTE) 11](#_Toc530439350)

[Caso de teste 1/4/6 (testReturnBicycleDEPOSITEXISTE) 11](#_Toc530439352)

[Caso de teste 5 (testReturnBicycleDEPOSITNAOEXISTE) 12](#_Toc530439354)

[Caso de teste 8 (testReturnBicycleSEMLUGARESLIVRES) 12](#_Toc530439356)

[Metódo bicycleRentalFee 12](#_Toc530439358)

[Tabela ECP 12](#_Toc530439359)

[Tabela BVA 13](#_Toc530439360)

[Caso de teste 14](#_Toc530439361)

[Metódo verifyCredit 15](#_Toc530439362)

[Tabela ECP 15](#_Toc530439363)

[Tabela BVA 15](#_Toc530439364)

[Caso de teste 1 (testVerifyCreditUSERMENOSUM) 15](#_Toc530439365)

[Caso de teste 2 (testVerifyCreditUSERZERO) 16](#_Toc530439366)

[Caso de teste 3 (testVerifyCreditUSERUM) 16](#_Toc530439367)

[Metódo addCredit 17](#_Toc530439368)

[Tabela ECP 17](#_Toc530439369)

[Tabela BVA 17](#_Toc530439370)

[Caso de teste 1/4 testAddCredit1 17](#_Toc530439371)

[Caso de teste 2 (testAddCredit2) 18](#_Toc530439372)

[Caso de teste 3 (testAddCreditMenosUM) 18](#_Toc530439373)

[Caso de teste 5 (testAddCreditAmountZero) 18](#_Toc530439374)

[Metódo registerUser 19](#_Toc530439375)

[Tabela ECP 19](#_Toc530439376)

[Tabela BVA 19](#_Toc530439377)

[Caso de teste 20](#_Toc530439378)

[Caso de teste 1 ( testRegisterUserUSERJAEXISTE) 20](#_Toc530439379)

[Caso de teste 2 (testRegisterUserIDINVALIDO) 20](#_Toc530439380)

[Caso de teste 3 (testRegisterUserExist) 20](#_Toc530439381)

[Caso de teste 4 (testRegisterUserJAEXISTECOMZERO) 21](#_Toc530439382)

[Caso de teste 5 (testRegisterUserRENTALZERO) 21](#_Toc530439383)

[Caso de teste 6/7 (testRegisterUserValido) 21](#_Toc530439384)

[Caso de teste 8 (testRegisterUserRENTALRES) 22](#_Toc530439385)

[Conclusão 23](#_Toc530439386)

# 

# Introdução

O presente trabalho surge no âmbito da Unidade Curricular de Engenharia de Software II e em específico da parte prática da mesma.

Considere-se a existência de um sistema público de aluguer de bicicletas.

O sistema é composto por vários depósitos de bicicletas espalhados pela cidade. Cada depósito contém cerca de uma dúzia de bicicletas numa área para o efeito. O depósito de bicicletas contém um determinado número de lugares devidamente numerados, sendo um lugar para cada bicicleta, onde as bicicletas estão inicialmente colocadas com um sistema “*lock/unlock*”.

Este sistema encontra-se conectado ao sistema principal e funciona da seguinte forma:

* Se um utilizador está registado e tem crédito na sua conta, pode alugar uma bicicleta, o sistema “liberta” a bicicleta e o utilizador pode retirar uma bicicleta, tornando-se responsável por ela até que a devolva.;
* Quando o utilizador pretende devolver a bicicleta, escolhe um lugar disponível no depósito de bicicletas para o estacionamento da mesma. O sistema *“lock/unlock”* deteta a bicicleta automaticamente e “prende” a bicicleta. A partir desse momento o utilizador deixa de ser responsável pela bicicleta.
* Para que uma pessoa possa alugar bicicletas terá de se registar, fornecendo um conjunto de dados entre os quais o seu nome e a informação do cartão de crédito, recebendo um login (ID). Se, de seguida, o utilizador desejar alugar uma bicicleta, deve dirigir-se a um depósito e introduz, através de uma interface para o efeito, como um monitor tátil, o seu login (ID). Após as respetivas validações, o sistema seleciona uma bicicleta entre as disponíveis e “liberta-a”. No momento da entrega, não há qualquer interação com qualquer interface.

Para além das funcionalidades descritas, o sistema deve acompanhar o estado de todas as bicicletas e alugueres., o utilizador deverá pagar uma determinada quantia pelo aluguer, dependendo da duração do respetivo aluguer e os utilizadores podem escolher uma bicicleta de um determinado depósito e entrega-la num outro qualquer à escolha. O sistema deverá igualmente monitorizar a manutenção das bicicletas (uma bicicleta que nunca foi alugada pode provavelmente estar avariada). Esta monitorização deverá ser realizada à distribuição das bicicletas pelos depósitos.

Os requisitos do problema apresentado encontram-se implementados na biblioteca *BikeRentalSystem*.

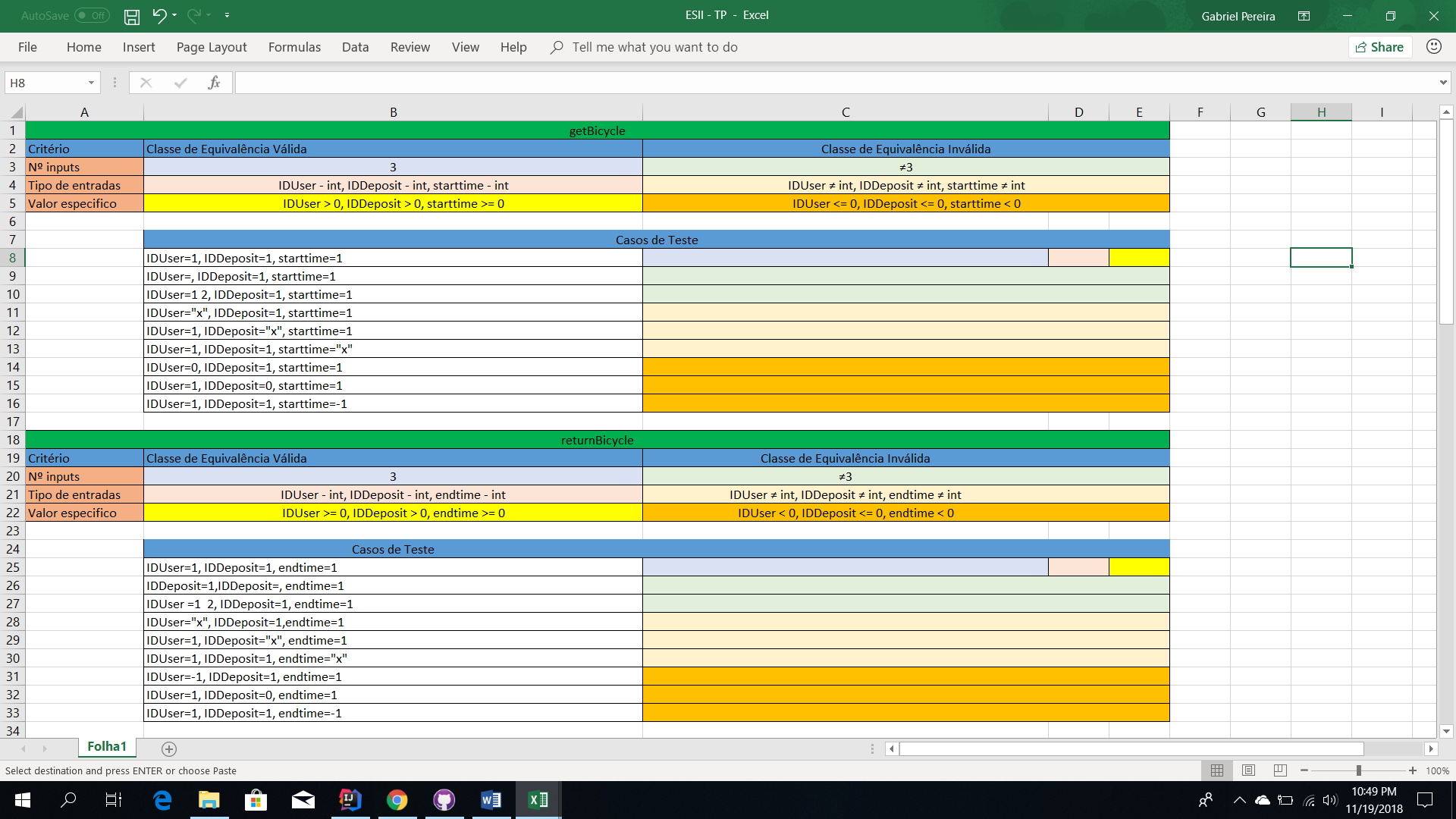
Este trabalho tem como objetivo a especificação de casos de teste utilizando as técnicas *Equivalence Class Partitioning* e *Boundary Value Analysis* no nível de testes unitários especificando casos de teste para cada método. Deverá ser descrito igualmente *test inputs*, *execution conditions* e *expected outputs*, assegurando que os casos de teste cobrem *Valid equivalence classes* e *Invalid equivalence classes*.

Todos estes testes deverão ser implementados e executados utilizando a *framework*: *JUnit* 5.1.

Como último objetivo define-se a elaboração de um relatório de testes, registo dos resultados dos testes especificando o *test case ID*, objeto testado, descrição, entradas, saídas esperadas e comentários relevantes. De igual forma, a implementação de cada um dos testes deverá devidamente documentado de forma a identificar cada caso de teste.

# Metódo getBicycle

## Tabela ECP

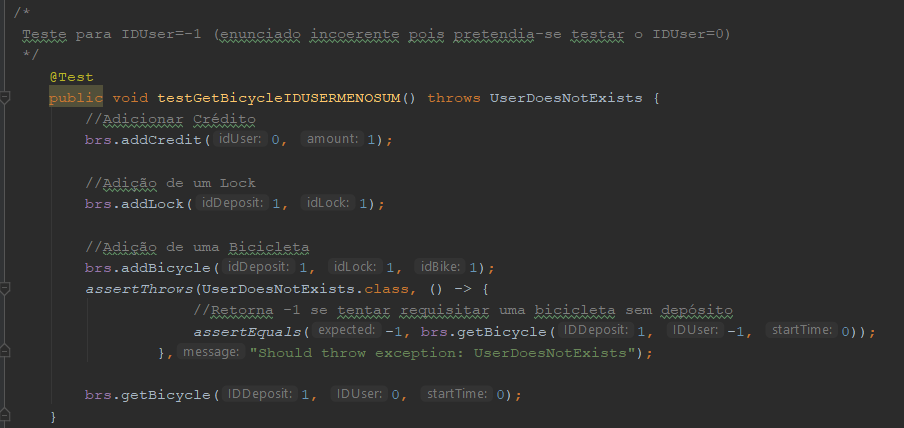


## Tabela BVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | getBicycle | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | IDUser>0, IDDeposit >0, starttime>=0 | | | | |
| 2 | IDUser=<0, IDDeposit >0, , starttime>=0 | | | | |
| 3 | IDUser>0, IDDeposit=<0, , starttime>=0 | | | | |
| 4 | IDUser>0, IDDeposit>0, , starttime<0 | | | | |
|  |  |  |  |  |  |
| Casos de Teste | Input | | |  | Output |
|  | **IDUser** | **IDDeposit** | **starttime** |  |  |
| 1 | 1 | 1 | 0 |  | identificador da bicicleta que será utilizada |
| 2 | 0 | 1 | 0 |  | -1 |
| 3 | 1 | 1 | 0 |  | identificador da bicicleta que será utilizada |
| 4 | 1 | 0 | 0 |  | -1 |
| 5 | 1 | 1 | -1 |  | -1 |
| 6 | 1 | 1 | 0 |  | identificador da bicicleta que será utilizada |
| 7 | 1 | 1 | 1 |  | identificador da bicicleta que será utilizada |

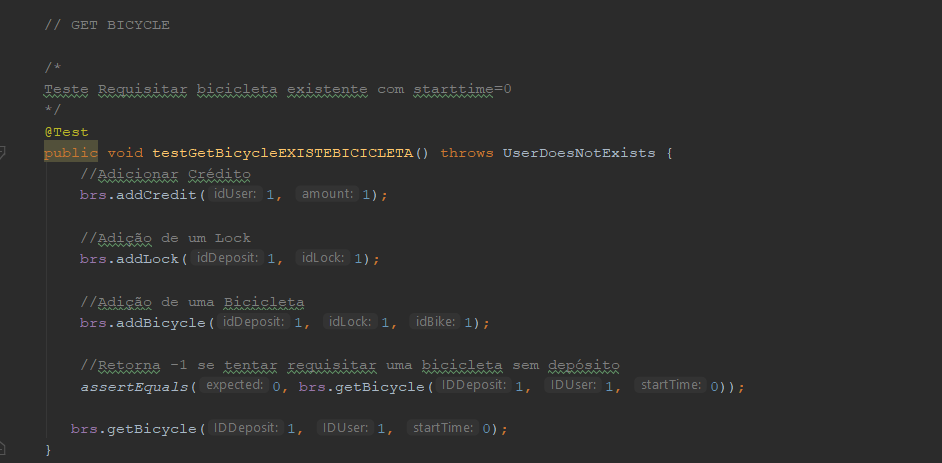
## Casos de teste

## Caso de teste 0 (testGetBicycleUSERMENOSUM)

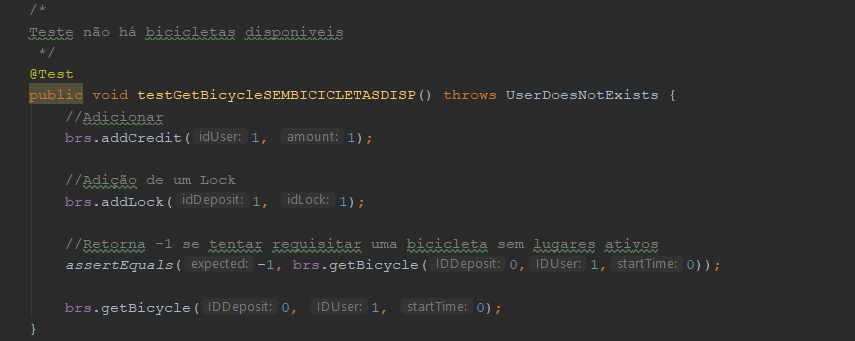


Caso de teste extra criado devido á falta de validações no IDUser que permite que este seja 0 (o que não deveria acontecer visto ser suposto que IDUser > 0) e para que sejam testadas todas as fronteiras foi necessário adicionar o teste para IDUser = -1.

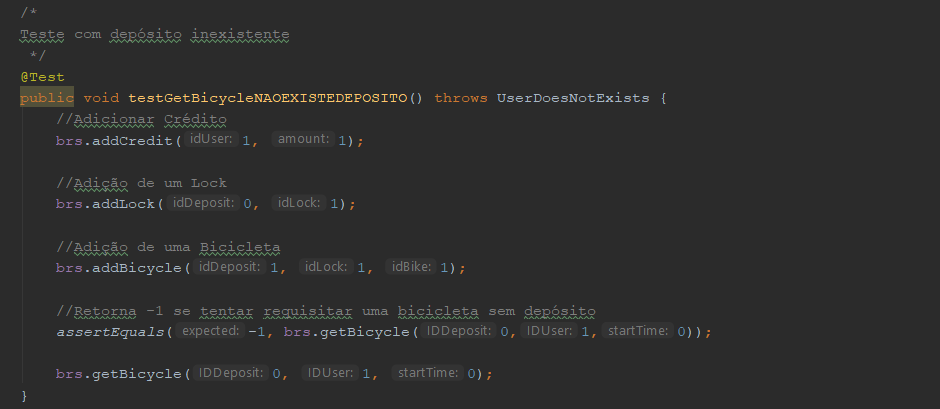
## Caso de teste 1/3/6 (testGetBicycleEXISTEBICICLETA)



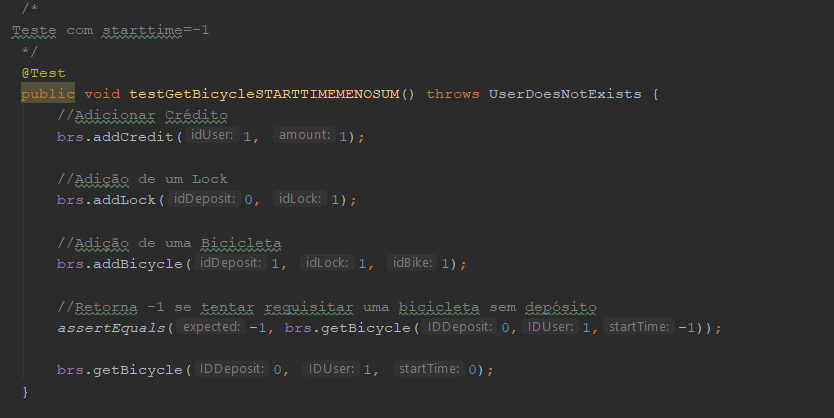
## Caso de teste 4 (testGetBicucleSEMBICICLETASDISP)



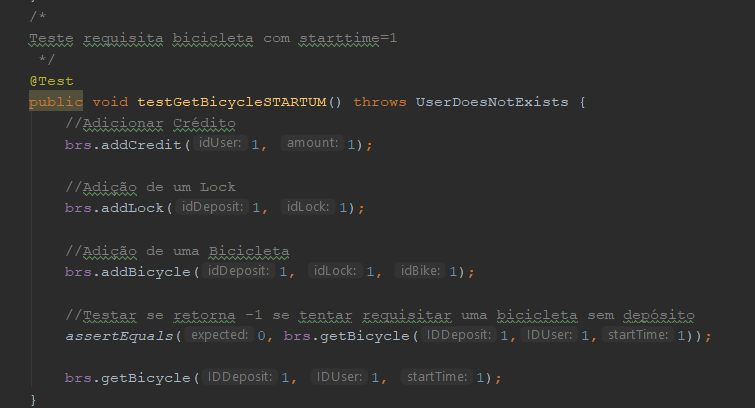
## Caso de teste 4 (testGetBicycleNAOEXISTEDEPOSITO)



## Caso de teste 5 (testGetBicycleSTARTTIMEMENOSUM)

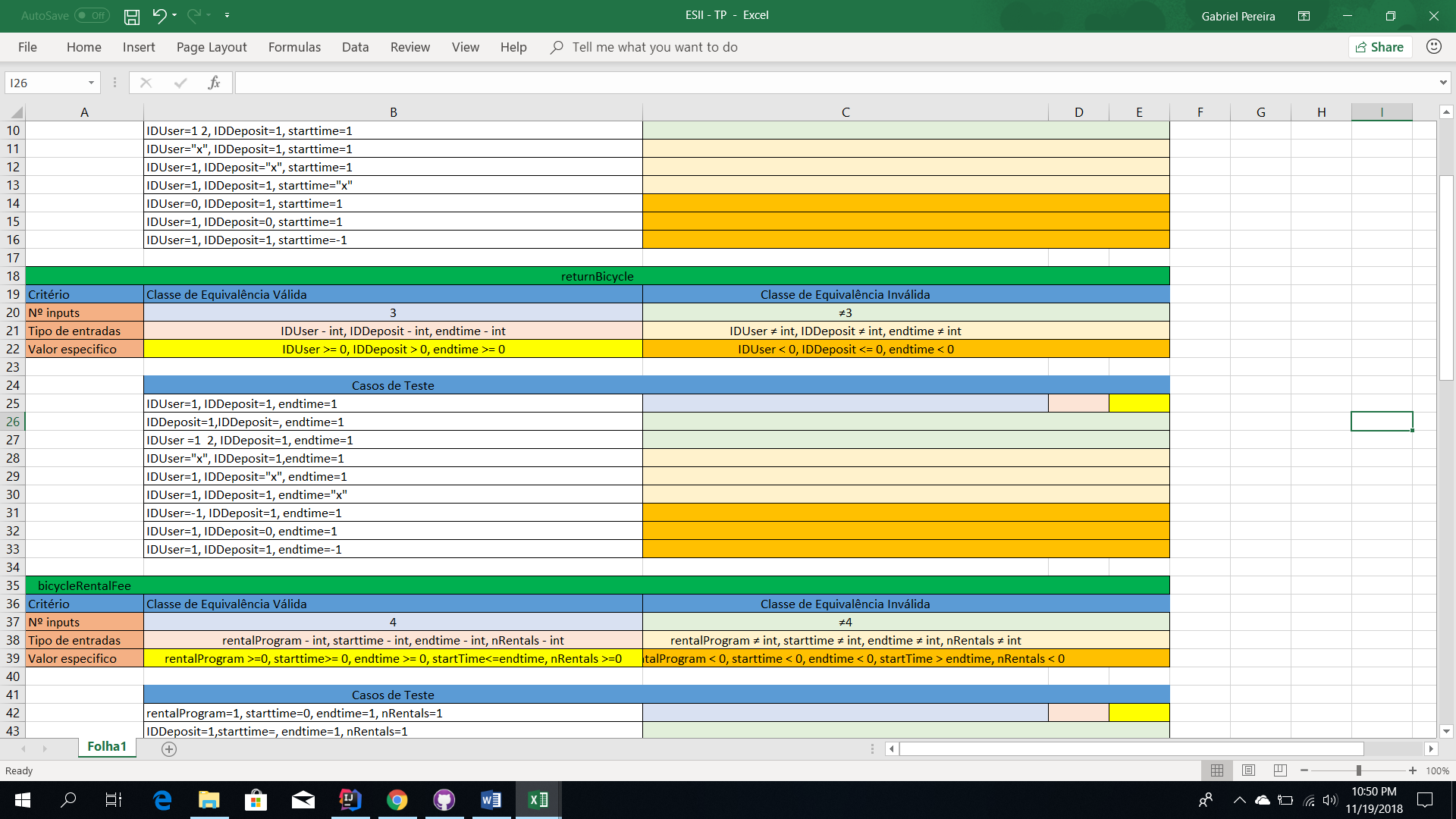


## Caso de teste 7 (testGetBicycleSTARTUM)



# Metódo returnBikycle

## Tabela ECP



## Tabela BVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | returnBikycle | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | IDUser>=0, IDDeposit >0, endtime>=0 | | | | |
| 2 | IDUser<0, IDDeposit >0, , endtime>=0 | | | | |
| 3 | IDUser>0, IDDeposit=<0, , endtime>=0 | | | | |
| 4 | IDUser>0, IDDeposit>0, , endtime<0 | | | | |
|  |  |  |  |  |  |
| Casos de Teste | Input | | |  | Output |
|  | **IDUser** | **IDDeposit** | **endtime** |  |  |
| 1 | 1 | 1 | 1 |  | Saldo atual |
| 2 | 0 | 1 | 1 |  | Saldo atual |
| 3 | -1 | 1 | 1 |  | -1 |
| 4 | 1 | 1 | 1 |  | Saldo atual |
| 5 | 1 | 0 | 1 |  | -1 |
| 6 | 1 | 1 | 1 |  | Saldo atual |
| 7 | 1 | 1 | 0 |  | Saldo atual |
| 8 | 1 | 1 | -1 |  | -1 |

## Caso de teste 1 (testReturnBicycleRETORNASALDOUSERUM)

## Caso de teste 2 (testReturnBicycleRETORNASALDOUSEROZERO)

## Caso de teste 3 (testReturnBicycleUSERNAOEXISTE)

## 

## Caso de teste 1/4/6 (testReturnBicycleDEPOSITEXISTE)

## 

## Caso de teste 5 (testReturnBicycleDEPOSITNAOEXISTE)

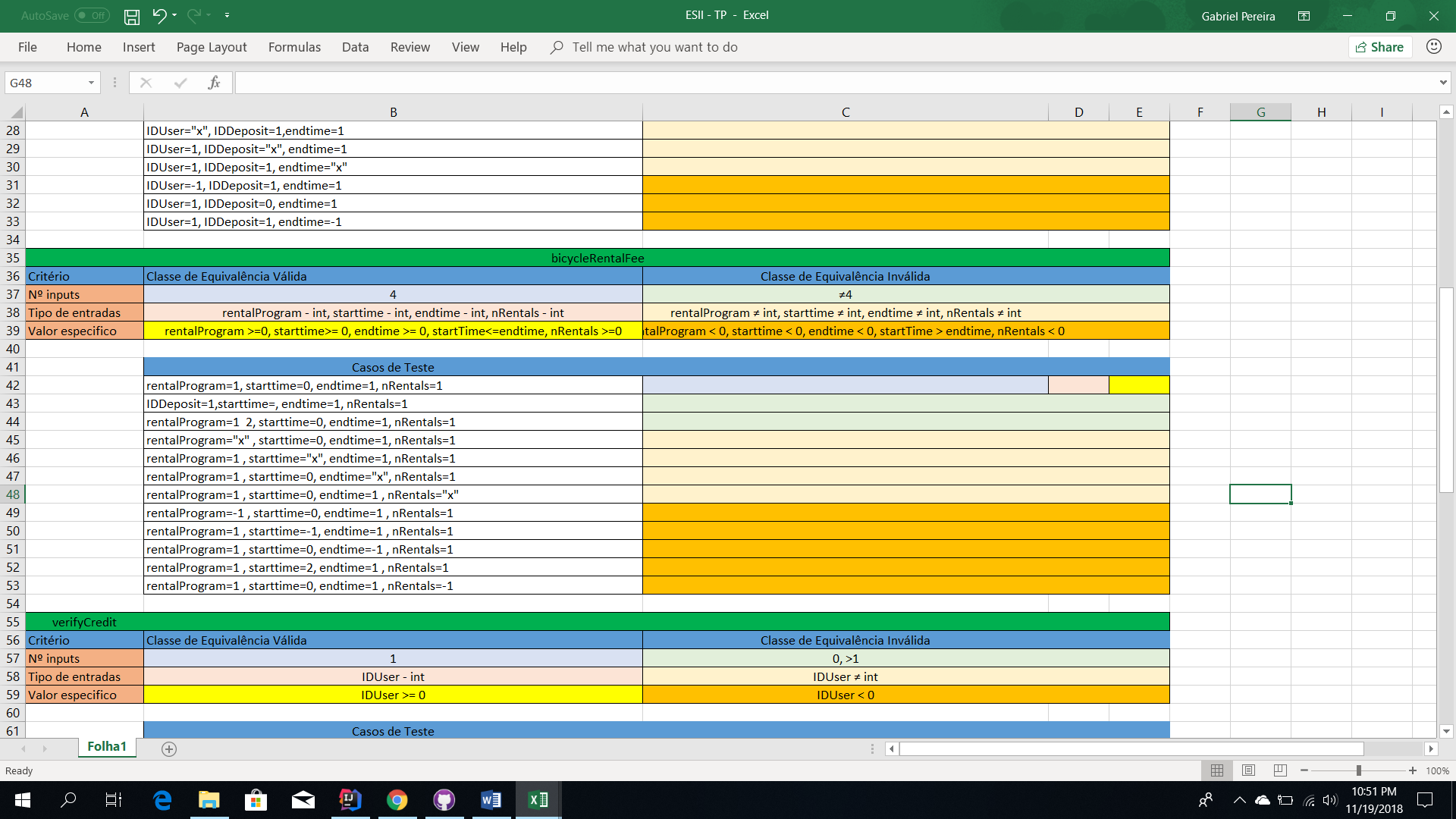
## 

## Caso de teste 8 (testReturnBicycleSEMLUGARESLIVRES)

## 

# Metódo bicycleRentalFee

## Tabela ECP

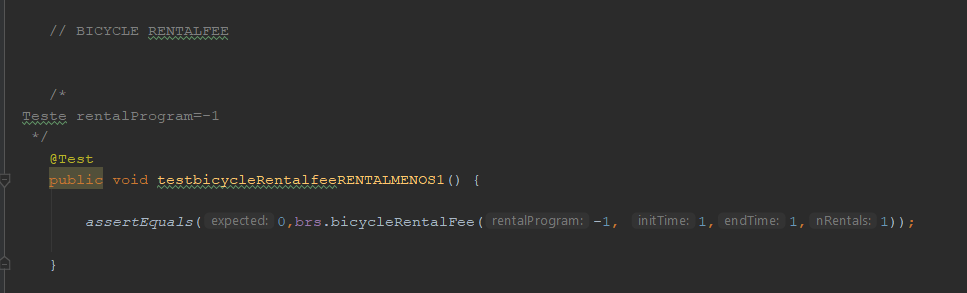


## Tabela BVA

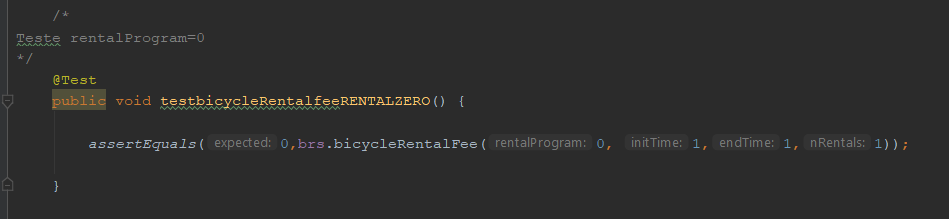
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | bicycleRentalFee | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | rentalProgram >=0, starttime>=0, endtime>=0, starttime <=endtime, nRentals>=0 | | | | |
| 2 | rentalProgram <0, starttime>=0, endtime>=0, starttime <=endtime, nRentals>=0 | | | | |
| 3 | rentalProgram >=0, starttime<0, endtime>=0, starttime<=endtime, nRentals>=0 | | | | |
| 4 | rentalProgram >=0, starttime>=0, endtime<0, starttime <=endtime, nRentals>=0 | | | | |
| 5 | rentalProgram >=0, starttime<0, endtime>=0, starttime>endtime, nRentals>=0 | | | | |
| 6 | rentalProgram >=0, starttime>=0, endtime<0, starttime <=endtime, nRentals<0 | | | | |
|  |  |  |  |  |  |
| Casos de Teste | Input | | | | Output |
|  | **rentalProgram** | **starttime** | **endtime** | **nRentals** |  |
| 1 | -1 | 1 | 2 | 1 | 0 |
| 2 | 0 | 1 | 2 | 1 | 0 |
| 3 | 1 | 1 | 2 | 1 | valor referente ao programa de aluguer |
| 4 | 2 | 1 | 2 | 1 | valor referente ao programa de aluguer |
| 5 | 3 | 1 | 2 | 1 | 0 |
| 6 | 1 | -1 | 1 | 1 | 0 |
| 7 | 1 | 0 | 1 | 1 | valor referente ao programa de aluguer |
| 8 | 1 | 1 | 2 | 1 | valor referente ao programa de aluguer |
| 9 | 1 | 1 | -1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 | 0 |
| 11 | 1 | 1 | 1 | 1 | valor referente ao programa de aluguer |
| 12 | 1 | 1 | 2 | -1 | 0 |
| 13 | 1 | 1 | 2 | 0 | valor referente ao programa de aluguer |
| 14 | 1 | 1 | 2 | 1 | valor referente ao programa de aluguer |

## Caso de teste

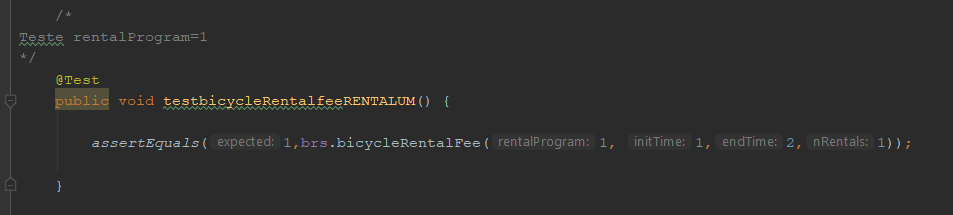
## Caso de teste 1 – testbicycleRentalfeeRENTALMENOS1



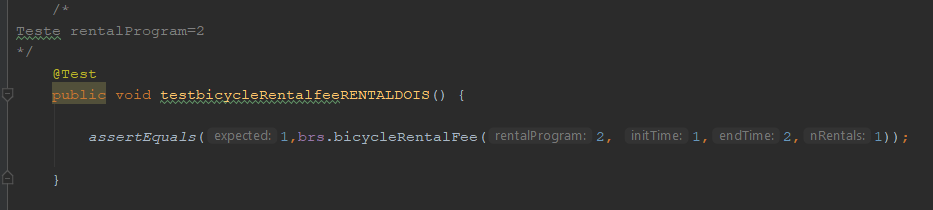
## Caso de teste 2 – testbicycleRentalfeeRENTALZERO



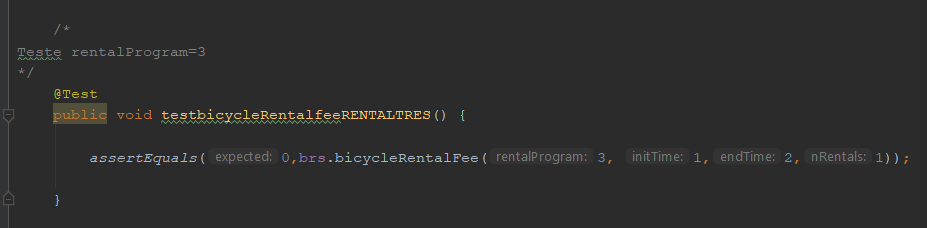
## Caso teste 3/8/14 – testbicycleRentalfeeRENTALUM



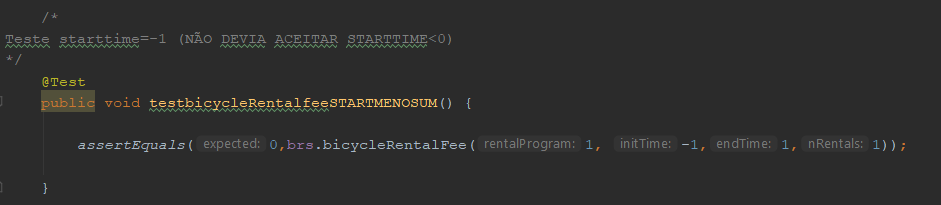
## Caso teste 4 – testbicycleRentalfeeRENTALDOIS



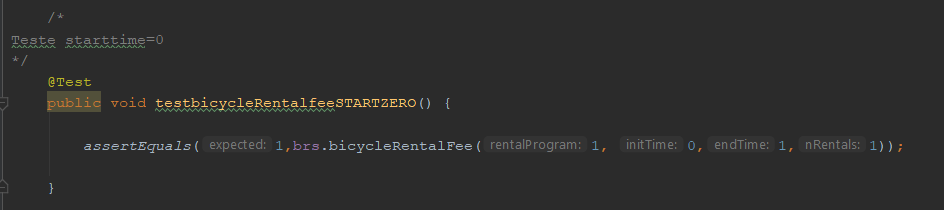
## Caso teste 5 – testbicycleRentalfeeRENTALTRES

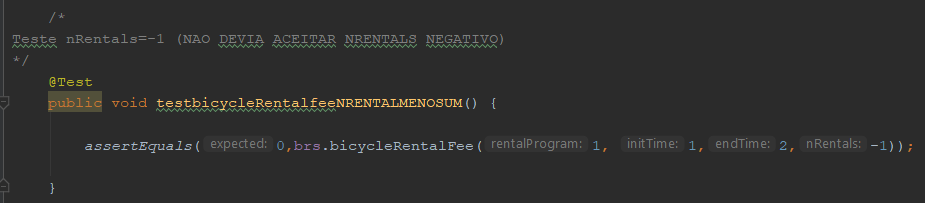


## Caso de teste 6 – testbicycleRentalfeeSTARTMENOSUM

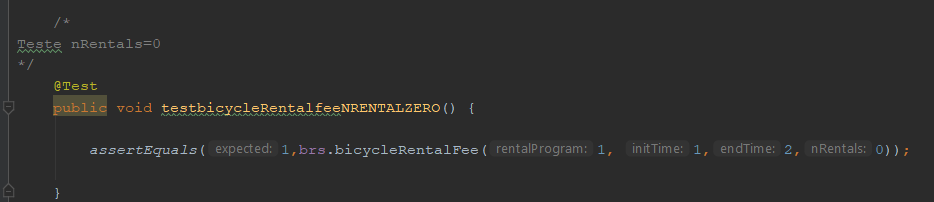


## Caso de teste 7 – testbicycleRentalfeeSTARTZERO

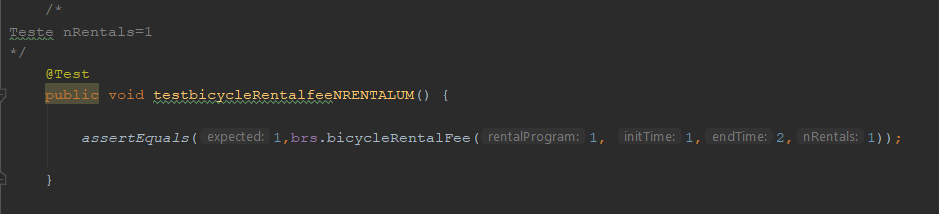


Caso de teste 9 – testbicycleRentalfeeNRENTALMENOSUM 

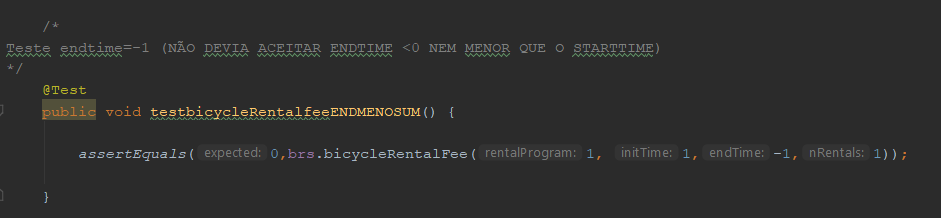
## Caso de teste 10 – testbicycleRentalfeeNRENTALZERO



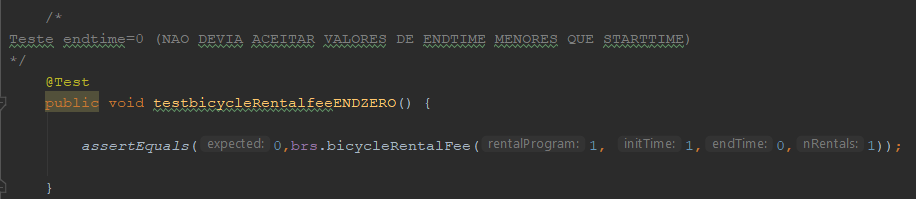
## Caso de teste 11 – testbicycleRentalfeeNRENTALUM



## Caso de teste 12 – testbicycleRentalfeeENDMENOSUM

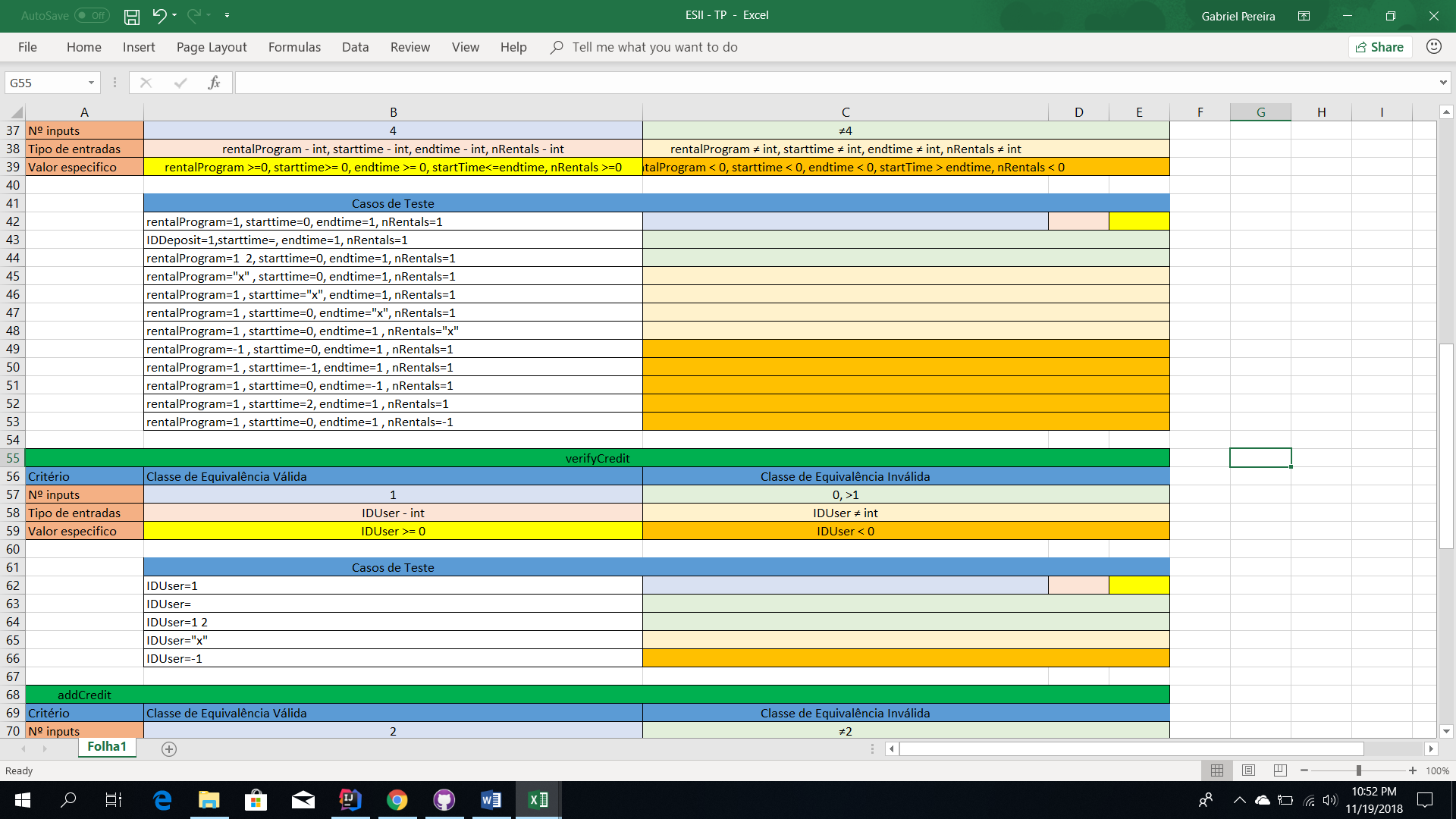


## Caso de teste 13 – testbicycleRentalfeeENDZERO



## Metódo verifyCredit

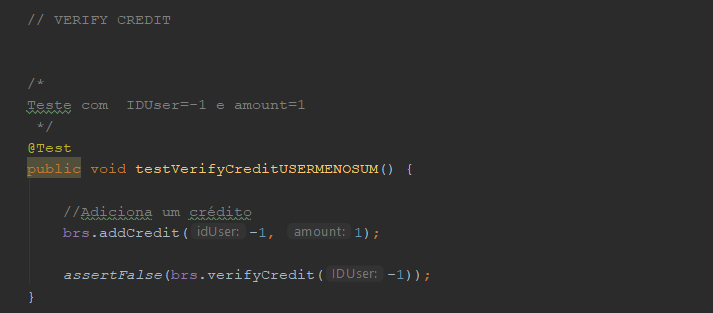
## Tabela ECP



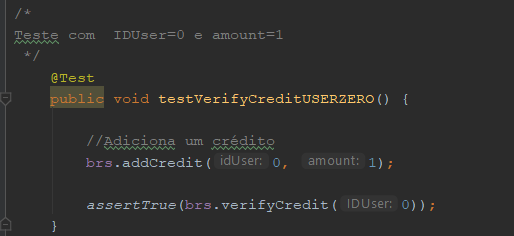
## Tabela BVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | verifyCredit | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | O IDUser >=0 | | | | |
| 2 | O IDUser <0 | | | | |
|  |  |  |  |  |  |
| Casos de Teste | Input | | |  | Output |
|  | **IDUser** | | |  |  |
| 1 | -1 | | |  | FALSE |
| 2 | 0 | | |  | TRUE |
| 3 | 1 | | |  | TRUE |

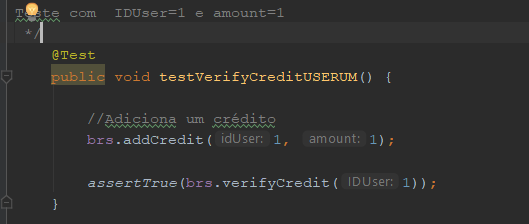
## Caso de teste 1 (testVerifyCreditUSERMENOSUM)



## Caso de teste 2 (testVerifyCreditUSERZERO)

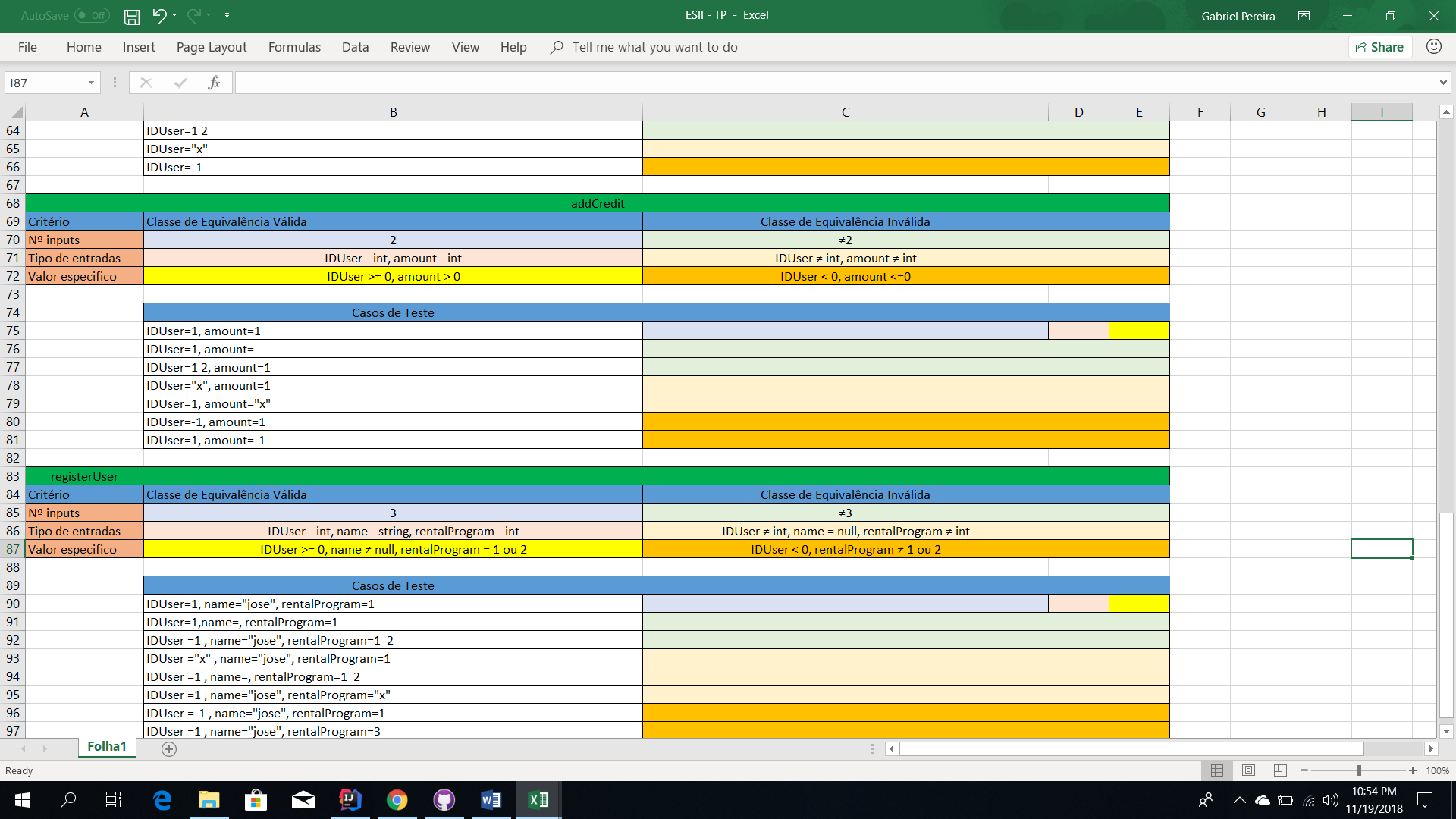


## Caso de teste 3 (testVerifyCreditUSERUM)



# Metódo addCredit

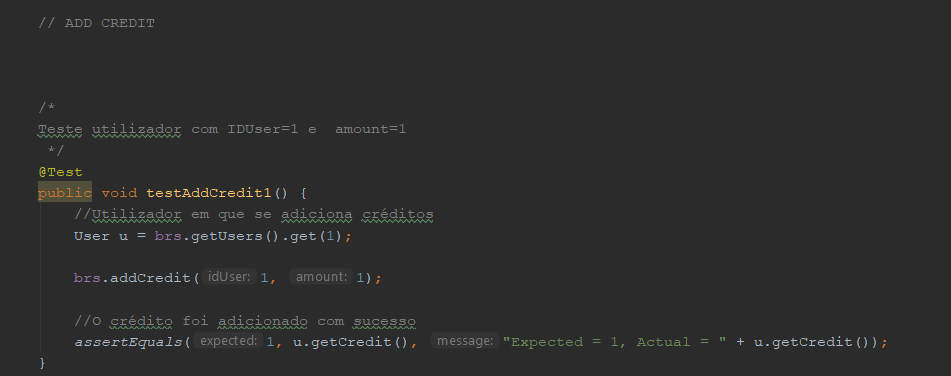
## Tabela ECP



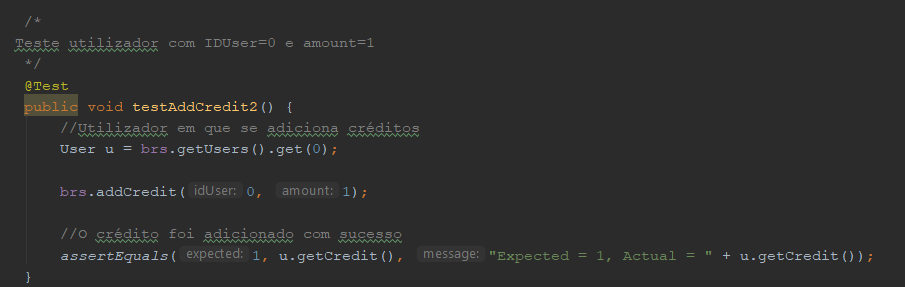
## Tabela BVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | addCredit | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | IDUser>=0, amount > 0 | | | | |
| 2 | IDUser<0, amount > 0 | | | | |
| 3 | IDUser>=0, amount<=0 | | | | |
|  |  |  |  |  |  |
| Casos de Teste | Input | |  |  | Output |
|  | **IDUser** | **amount** |  |  |  |
| 1 | 1 | 1 |  |  | adicionar valor amount ao credito |
| 2 | 0 | 1 |  |  | adicionar valor amount ao credito |
| 3 | -1 | 1 |  |  | null |
| 4 | 1 | 1 |  |  | adicionar valor amount ao credito |
| 5 | 1 | 0 |  |  | 0 |

## Caso de teste 1/4 testAddCredit1



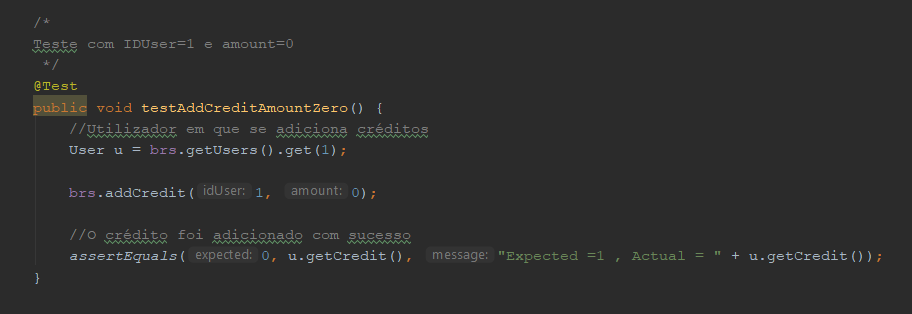
## Caso de teste 2 (testAddCredit2)



## Caso de teste 3 (testAddCreditMenosUM)

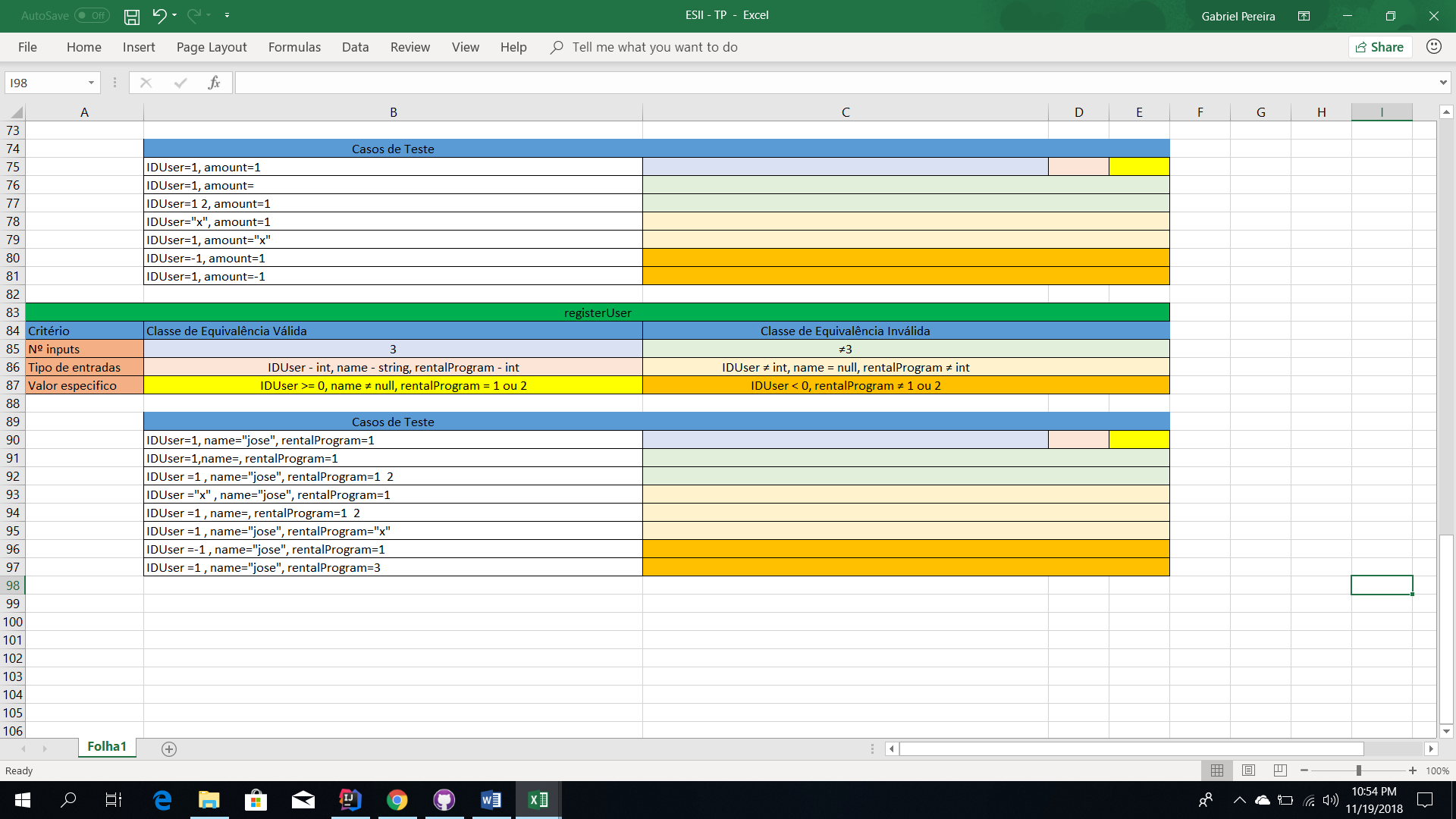


## Caso de teste 5 (testAddCreditAmountZero)



# Metódo registerUser

## Tabela ECP

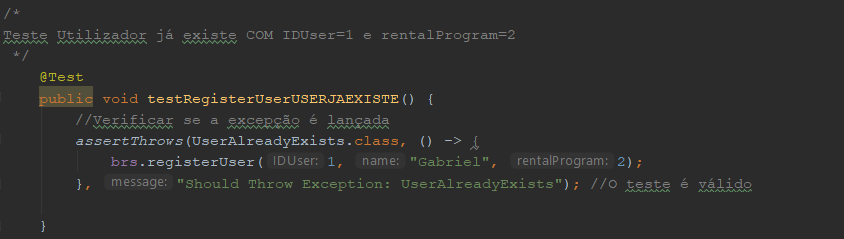


## Tabela BVA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | registerUser | | | | |
|  |  |  |  |  |  |
| Cenários |  |  |  |  |  |
| 1 | IdUser>=0 , name !=null , rentalProgram=1 | | | | |
| 2 | IdUser<0 , name !=null , rentalProgram=1 | | | | |
| 3 | IdUser>=0 , name =null , rentalProgram=1 | | | | |
| 4 | IdUser>=0 , name !=null , rentalProgram=0 | | | | |
| 5 | IdUser>=0 , name !=null , rentalProgram=1 | | | | |
| 6 | IdUser>=0 , name !=null , rentalProgram=2 | | | | |
| 7 | IdUser>=0 , name !=null , rentalProgram=3 | | | | |
| 8 |  |  |  |  |  |
|  |  |  |  |  |  |
| Casos de Teste | Input | | |  | Output |
|  | **IDUser** | **name** | **rentalProgram** |  |  |
| 1 | 1 | !null | 1 |  | Cria utilizador |
| 2 | -1 | !null | 1 |  | Exception |
| 3 | 0 | !null | 1 |  | Cria utilizador |
| 4 | 1 | null | 1 |  | Exception |
| 5 | 1 | !null | 0 |  | Exception |
| 6 | 1 | !null | 1 |  | Cria utilizador |
| 7 | 1 | !null | 2 |  | Cria utilizador |
| 8 | 1 | !null | 3 |  | Exception |

## Caso de teste

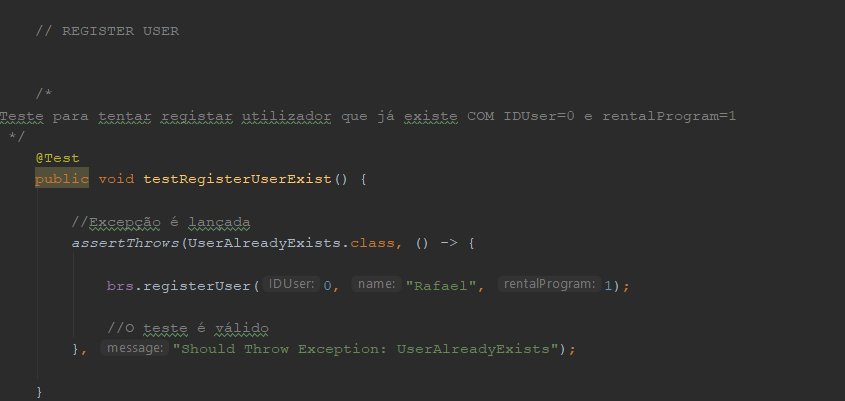
## Caso de teste 1 ( testRegisterUserUSERJAEXISTE)



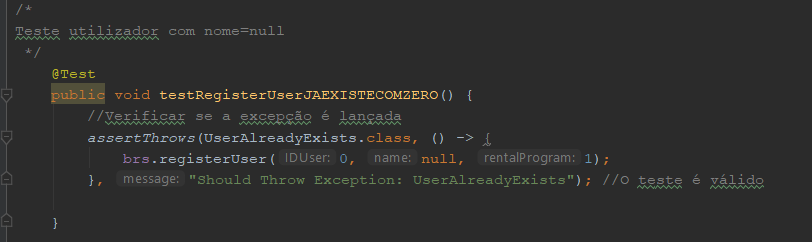
## Caso de teste 2 (testRegisterUserIDINVALIDO)



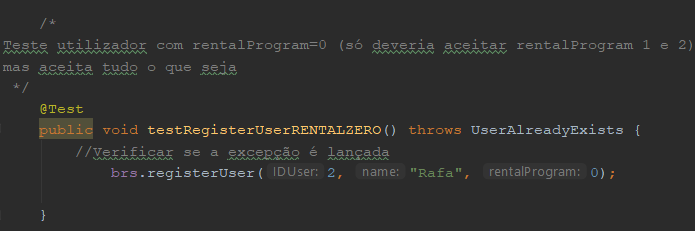
## Caso de teste 3 (testRegisterUserExist)



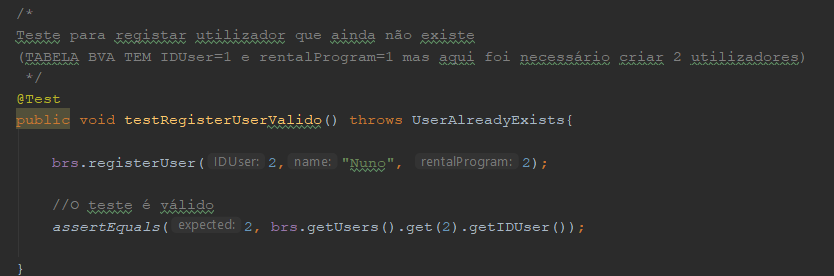
## Caso de teste 4 (testRegisterUserJAEXISTECOMZERO)



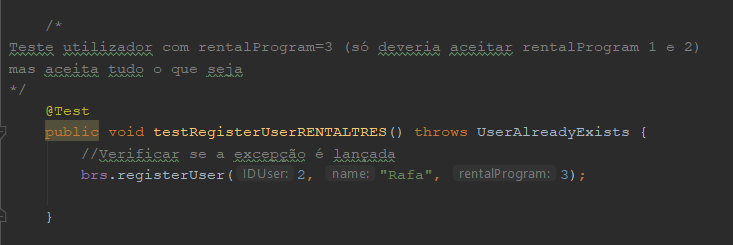
## Caso de teste 5 (testRegisterUserRENTALZERO)



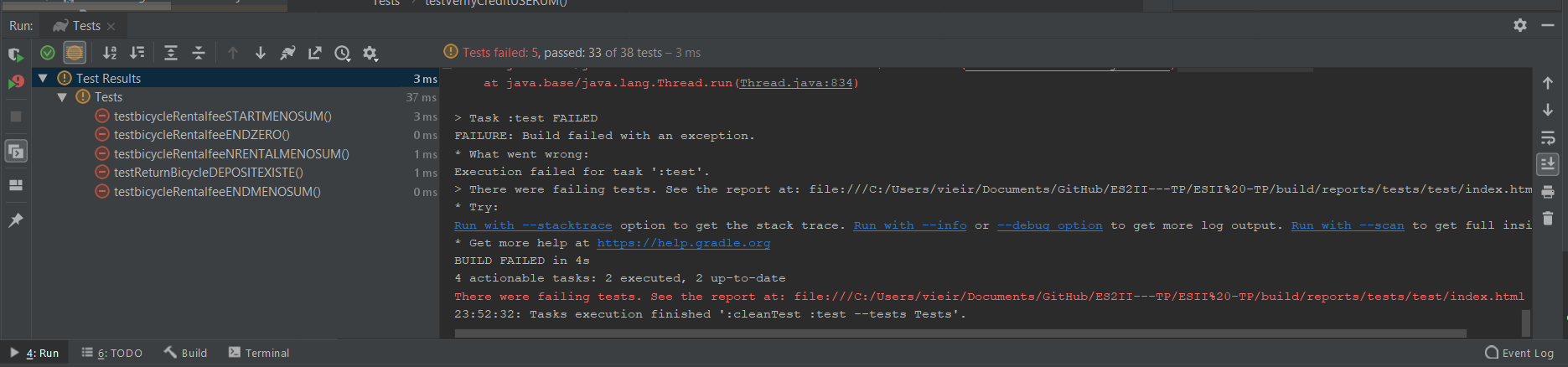
## Caso de teste 6/7 (testRegisterUserValido)



## Caso de teste 8 (testRegisterUserRENTALRES)



# Teste Gerais



# Conclusão

Este projeto vem dar resposta a uma solicitação de especificação de casos de testes referente a um sistema de aluguer de bicicletas usando as técnicas *Equivalence Class Partitioning* e *Boundary Value Analysis,* descrição de *test inputs, execution conditions* e *expected outputs,* assegurando que os casos de teste cobrem *Valid equivalence classes* e *Invalid equivalence classes.*

Apesar das dificuldades iniciais na interpretação do enunciado, na configuração da *framework*: *JUnit* 5.1 e no uso das técnicas de *ECP* e de *BVA*, todos os requisitos do projeto foram possíveis de concluir por completo e com sucesso.