

**Universidade de São Paulo**  
**Escola de Artes, Ciências e Humanidades**  
**Bacharelado em Sistemas de Informação**  
**ACH2028 - Qualidade de Software**

**Atividade 03 - Teste Estrutural ou Caixa-Branca**

Karina Duran Munhos - 11295911

Vitor Caetano da Silva - 9276999

**1)** Implementar as funções `classificaCliente(pontos)` e `calculaTaxaDesconto(tipoCliente, valorCompra)` para as quais você criou casos de teste na atividade anterior. Vocês podem escolher a linguagem de programação.

```
def classificaCliente(pontos):
    if(pontos < 0):
        return "erro"
    elif(pontos <= 10000):
        return "bronze"
    elif(pontos <= 20000):
        return "prata"
    elif(pontos <= 40000):
        return "ouro"
    else:
        return "platinum"

def calculaTaxaDesconto(tipoCliente, valorCompra):
    taxa = 0
    if (valorCompra >= 500 or tipoCliente == "ouro"):
        taxa = 15
    else:
        if (tipoCliente == "prata" or valorCompra >= 400):
            taxa = 10
        else:
            if (valorCompra >= 200 or tipoCliente == "bronze"):
                taxa = 5
    return taxa
```

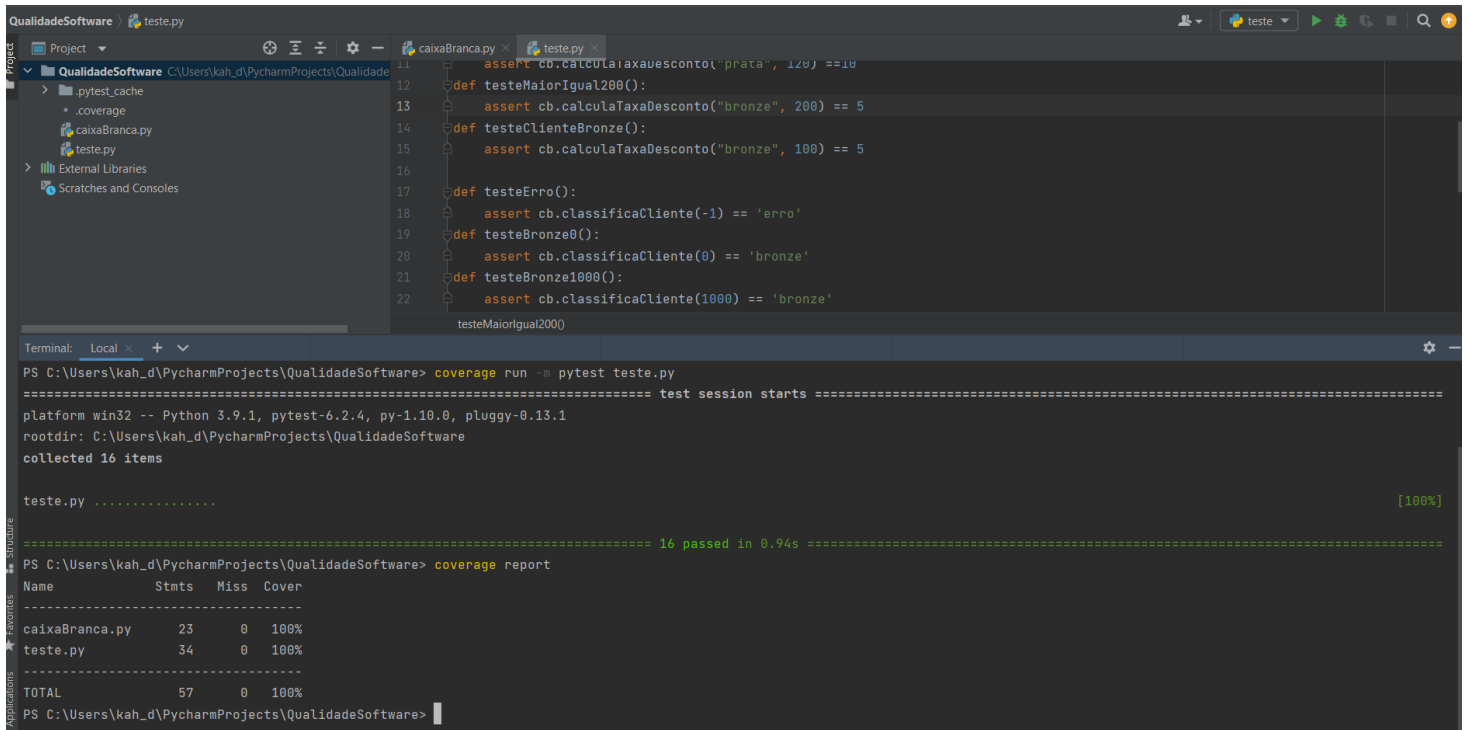
## 2) Escrever casos de teste automatizados considerando os casos de testes criados na atividade anterior.

```
import pytest
import caixaBranca as cb

def testeMaiorIgual500():
    assert cb.calculaTaxaDesconto("bronze", 500) == 15
def testeClienteOuro():
    assert cb.calculaTaxaDesconto("ouro", 160) == 15
def testeMaiorIgual400():
    assert cb.calculaTaxaDesconto("bronze", 400) == 10
def testeClientePrata():
    assert cb.calculaTaxaDesconto("prata", 120) == 10
def testeMaiorIgual200():
    assert cb.calculaTaxaDesconto("bronze", 200) == 5
def testeClienteBronze():
    assert cb.calculaTaxaDesconto("bronze", 100) == 5

def testeErro():
    assert cb.classificaCliente(-1) == 'erro'
def testeBronze0():
    assert cb.classificaCliente(0) == 'bronze'
def testeBronze1000():
    assert cb.classificaCliente(1000) == 'bronze'
def testeBronze10000():
    assert cb.classificaCliente(10000) == 'bronze'
def testePrata10001():
    assert cb.classificaCliente(10001) == 'prata'
def testePrata20000():
    assert cb.classificaCliente(20000) == 'prata'
def testeOuro20001():
    assert cb.classificaCliente(20001) == 'ouro'
def testeOuro40000():
    assert cb.classificaCliente(40000) == 'ouro'
def testePlatinum40001():
    assert cb.classificaCliente(40001) == 'platinum'
def testePlatinum():
    assert cb.classificaCliente(50000) == 'platinum'
```

3) Executar os casos de teste e apresentar os resultados em termos de cobertura de código (instruções, linhas, decisões/branches, etc)



The screenshot shows the PyCharm IDE interface. The top pane displays the `teste.py` file with several test functions. The bottom pane shows the terminal output of the test execution and coverage report.

```
def testeMaiorIgual200():
    assert cb.calculaTaxaDesconto('prata', 120) == 10
def testeMaiorIgual200():
    assert cb.calculaTaxaDesconto("bronze", 200) == 5
def testeClienteBronze():
    assert cb.calculaTaxaDesconto("bronze", 100) == 5
def testeErro():
    assert cb.classificaCliente(-1) == 'erro'
def testeBronze0():
    assert cb.classificaCliente(0) == 'bronze'
def testeBronze1000():
    assert cb.classificaCliente(1000) == 'bronze'
```

```
PS C:\Users\kah_d\PycharmProjects\QualidadeSoftware> coverage run -m pytest teste.py
===== test session starts =====
platform win32 -- Python 3.9.1, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\kah_d\PycharmProjects\QualidadeSoftware
collected 16 items

teste.py ..... [100%]

===== 16 passed in 0.94s =====
PS C:\Users\kah_d\PycharmProjects\QualidadeSoftware> coverage report
Name      Stmts   Miss  Cover
-----
caixaBranca.py    23     0   100%
teste.py         34     0   100%
-----
TOTAL            57     0   100%
```