

**UNIVERSIDADE FEDERAL DO CEARÁ****Departamento de Computação**

Prof. Lincoln Souza Rocha

CK0442- Técnicas de Programação para Ciência de Dados - T01

TP06
2024.2

1. Implementando Exceções em Contas do SisBanco

Descrição: Considerando o módulo `sisbanco.py` disponível no ArquivosTP06.zip no SIGAA/SI3, levante as exceções `VIException`, `SIException` e `TJIException` disponíveis em `excecoes.py` (dentro de ArquivosTP06.zip) seguindo as regras estabelecidas abaixo.

1. `ContaAbstrata.creditar()`

- Caso 01: o método deve levantar a exceção `VIException` (exceção de valor inválido) caso o argumento `valor` possuir conteúdo `None` ou negativo.

2. `Conta.debitar()` e `ContaImposto.debitar()`

- Caso 01: o método deve levantar a exceção `VIException` (exceção de valor inválido) caso o argumento `valor` possuir conteúdo `None` ou negativo.
- Caso 02: o método deve levantar a exceção `SIException` (exceção de saldo insuficiente) caso o valor a ser debitado (considerando a taxa de imposto) for maior que o valor do saldo da conta.

3. `ContaPoupanca.render_juros()`

- Caso 01: o método deve levantar a exceção `TJIException` (exceção de taxa de juros inválida) caso o argumento `taxa` possuir conteúdo `None`, zero ou negativo.

2. Implementando Exceções no Banco do SisBanco

Descrição: Considerando o módulo `sisbanco.py` disponível no ArquivosTP06.zip no SIGAA/SI3, levante as exceções `CIException`, `CEException` e `VIException` disponíveis em `excecoes.py` (dentro de ArquivosTP06.zip) seguindo as regras estabelecidas abaixo.

1. `Banco.debitar()`

- Caso 01: o método deve levantar a exceção `CIException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `numero` seja `None`.

2. `Banco.creditar()`

- Caso 01: o método deve levantar a exceção `CIException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `numero` seja `None`.

3. `Banco.saldo()`

- Caso 01: o método deve levantar a exceção `CIException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `numero` seja `None`.

4. `Banco.render_juros()`

- Caso 01: o método deve levantar a exceção `CIException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `numero` seja `None`.

5. `Banco.render_bonus()`

- Caso 01: o método deve levantar a exceção `CIException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `numero` seja `None`.

6. `Banco.cadastrar()`

- Caso 01: o método deve levantar a exceção `VIException` (exceção de valor inválido) caso o argumento `conta` possuir conteúdo `None` ou tipo diferente de uma das classes do `SisBanco`.

- **Caso 02:** antes de efetivar o cadastro, verifique se já existe no sistema uma conta com o mesmo número. Caso exista, levante a exceção `CEException` (exceção de conta existente).

7. `Banco.transferir()`

- **Caso 01:** o método deve levantar a exceção `CException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `origem` seja `None`.
- **Caso 02:** o método deve levantar a exceção `CException` (exceção de conta inexistente) caso o retorno da busca pela conta usando o argumento `destino` seja `None`.

3. Tratando Exceções no Terminal do SisBanco

Descrição: Implemente as funcionalidades descritas no arquivo `terminal.py` (ArquivosTP06.zip) do SisBanco com o respectivo código de tratamento de exceção (i.e., `try-except-else-finally`). (Dica! Use o método `print_mensagem_erro()` das exceções para imprimir a mensagem de erro dentro das cláusulas `except`).