PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Delano M. Beder

Roteiro 02b

Paradigma Orientado a Objetos - Conta Corrente (número e saldo)

Conta Corrente (número e saldo) e 4 operações:

- retirada de um valor em uma conta corrente (verificar se o saldo é suficiente para a retirada)
- depósito de um valor em uma conta corrente
- transferência de um valor entre 2 contas (retirada da 1a conta e depósito na 2a conta)
- imprime informações (número e saldo) de uma conta corrente

---

1. Crie um projeto (Aplicação C++) denominado **ContaCorrente-Cpp**
2. Implementação da classe **ContaCorrente**

   Crie o arquivo de cabeçalho C++ denominado **ContaCorrente.h**

```
#ifndef CONTACORRENTE_H
#define CONTACORRENTE_H

class ContaCorrente {
public:
    ContaCorrente(int numero, double saldo = 0); // construtor
    virtual ~ContaCorrente(); // destrutor
    // métodos da classe
    bool retirada(double valor);
    void deposito(double valor);
    bool transferencia(ContaCorrente &outra, double valor);
    void imprime() const;
private: // atributos da classe
    int numero;
    double saldo;
};

#endif /* CONTACORRENTE_H */
```

   Crie o arquivo de código-fonte C++ denominado **ContaCorrente.cpp**

```
#include <iostream>
#include "ContaCorrente.h"

using namespace std;

ContaCorrente::ContaCorrente(int numero, double saldo) {
    cout << "Construindo Conta " << numero << endl;
    this->numero = numero;
    this->saldo = saldo;
```

```
}

ContaCorrente::~ContaCorrente() {
    cout << "Destruindo Conta " << numero << endl;
}

bool ContaCorrente::retirada(double valor) {
    if (saldo - valor >= 0) {
        saldo -= valor;
        return true;
    }
    return false;
}

void ContaCorrente::deposito(double valor) {
    this->saldo += valor;
}

bool ContaCorrente::transferencia(ContaCorrente &outra, double valor) {
    bool ok = this->retirada(valor);
    if (ok) {
        outra.deposito(valor);
    }
    return ok;
}

void ContaCorrente::imprime() const {
    cout << "Numero: " << numero << ", saldo: " << saldo << endl;
}
```

3. Atualize o arquivo de código-fonte C++ denominado **main.cpp** (programa principal)

```
#include <iostream>
#include "ContaCorrente.h"

using namespace std;

int main() {

    ContaCorrente c1(1000, 1200.50);
    ContaCorrente c2(2000);

    cout << endl << "Antes da Transferência" << endl << endl;

    c1.imprime();
    c2.imprime();

    c1.transferencia(c2, 500);

    cout << endl << "Depois da Transferência" << endl << endl;

    c1.imprime();
    c2.imprime();

    cout << endl;
```

```
    return 0;
}
```

4. Crie o arquivo **Makefile** (opcional)

```
CXX          := g++
CXX_FLAGS    := -Wall -Wextra -std=c++17 -ggdb

LIBRARIES    :=
EXECUTABLE   := ContaCorrente-Cpp

all: run
    @rm -f $(EXECUTABLE)

run: $(EXECUTABLE)
    @./$(EXECUTABLE)

$(EXECUTABLE): *.cpp
    $(CXX) $(CXX_FLAGS) -I. $^ -o $@ $(LIBRARIES)
```

5. Compile, execute e verifique a saída impressa
6. Fim