CENTRO EDUCACIONAL TECNOLÓGICO DO AMAZONAS - CETAM
ESCOLA ESTADUAL DE TEMPO INTEGRAL GOVERNADOR MELO E PÓVOAS
CURSO TÉCNICO DE NÍVEL MÉDIO DE INFORMÁTICA

DESENVOLVIMENTO WEB E APLICAÇÕES

Estudo das etapas de criação, implementação de testes realizados

MANAUS - AM
2025

Francisco Mário Viana de Amorim
Ana Ester Nascimento Feitosa
Julita Angelina Lopes de Souza
Shymenne Siqueira Carvalho
João Arlesson Gomes de Farias
Vitória Rafaele Rocha de Andrade

Relatório técnico atividades práticas, apresentado como requisito parcial para a aprovação na unidade curricular/curso: linguagem de programação I - Técnico de nível médio em Informática.

Professora Esp. Priscila Gonçalves.

MANAUS - AM
2025

SUMÁRIO

1. Introdução

O avanço das tecnologias computacionais tem permitido o desenvolvimento de softwares cada vez mais robustos, seguros e eficientes. Nesse cenário, a Programação Orientada a Objetos (POO) se destaca como um dos paradigmas mais utilizados para modelar sistemas complexos, possibilitando representar elementos do mundo real através de classes e objetos. Este trabalho apresenta o desenvolvimento de um sistema bancário simples, implementado em linguagem C++, utilizando conceitos fundamentais da POO, como encapsulamento, herança, polimorfismo e abstração. O sistema simula operações básicas de um ambiente bancário, incluindo criação de contas, autenticação via senha, depósitos, saques e geração de diferentes tipos de contas (corrente, poupança e salário).

2. Objetivo Geral

Desenvolver um sistema bancário em C++ utilizando Programação Orientada a Objetos, modelado a partir de princípios estruturados de análise e design, contemplando criação de contas, autenticação, operações financeiras e interação através de um menu em console.

3. Objetivos Específicos

- Implementar classes e objetos que representem elementos essenciais do sistema bancário.
- Aplicar os conceitos de encapsulamento, herança e polimorfismo na construção das classes.
- Modelar o sistema utilizando diagramas UML para melhor organização e entendimento da estrutura do software.
- Criar um menu interativo no console para permitir a interação do usuário com o sistema.
- Desenvolver rotinas de autenticação para garantir a segurança das operações bancárias.
- Testar e validar o funcionamento do sistema através de exemplos práticos.
- Demonstrar a importância da POO no desenvolvimento de sistemas reais.

4. Desenvolvimento

O desenvolvimento do sistema seguiu uma abordagem modular, estruturada a partir de classes, para representar os diferentes elementos do contexto bancário.

4.1 Estrutura de Classes

O sistema foi constituído pelas seguintes classes principais:

- **Cliente**: Representa o titular da conta, armazenando nome e CPF.
- **Conta (classe base)**: Contém os atributos e métodos fundamentais, como número da conta, saldo, senha, depósito, saque e autenticação.
- **ContaCorrente**: Especialização da classe Conta, com implementação de limite.
- **ContaPoupanca**: Classe derivada contendo taxa de rendimento.
- **ContaSalario**: Classe derivada com limitação de saques mensais.
- **Banco**: Responsável por gerenciar todas as contas do sistema, realizar buscas e armazenar as operações.

4.2 Exemplos Práticos em POO

Criação de Objetos

```
Cliente c1("Maria Silva", "123.456.789-00");
ContaCorrente conta1(1001, c1, 500.0, "1234");
```

Autenticação de Senha

```
if (conta1.autenticar("1234")) {
    cout << "Acesso autorizado!" << endl;
}
```

Depósito

```
conta1.depositar(200.0);
```

Saque

```
if (conta1.sacar(100.0)) {
    cout << "Saque realizado!" << endl;
}
```

Polimorfismo

```
Conta* conta = new ContaPoupanca(2001, c1, 0.02, "9999");
conta->depositar(300);
conta->sacar(50);
```

Gerenciamento de Contas

```
Banco banco;
banco.criarContaCorrente(1001, c1, 500.0, "1234");
banco.listarContas();
```

5. Metodologia Utilizada

A metodologia adotada para o desenvolvimento deste projeto seguiu os seguintes passos:

1. **Análise e levantamento de requisitos**: Identificação das funcionalidades mínimas necessárias para o sistema bancário.

2. **Modelagem UML**: Criação dos diagramas de classes e estruturação das relações entre objetos utilizando notação padrão.

3. **Programação Orientada a Objetos em C++**: Implementação das classes utilizando encapsulamento, herança e polimorfismo.

4. **Implementação de mecanismos de autenticação**: Adição de rotinas de segurança através de senhas privadas.

5. **Testes práticos**: Execução de diversos cenários para validar as operações de criação de contas, saques, depósitos e autenticação.

6. **Documentação**: Organização de código-fonte, diagramas e explicações conceituais para fins educacionais.

6. Conclusão

O desenvolvimento do sistema bancário em C++ utilizando Programação Orientada a Objetos demonstrou a eficácia desse paradigma no tratamento de problemas reais. Através da criação de classes, reutilização de código por herança e aplicação de polimorfismo, foi possível construir um sistema organizado, seguro e funcional.

O exercício contribuiu para o fortalecimento dos conceitos fundamentais de POO e mostrou como técnicas de modelagem, como UML, auxiliam no planejamento e clareza do projeto. O sistema apresentado pode servir como base para evoluções futuras, incluindo conexão com banco de dados, interfaces gráficas e funcionalidades avançadas de segurança.

7. Referências Bibliográficas

- Deitel, H. M., & Deitel, P. J. **C++ Como Programar**. 10ª edição. Pearson, 2017.

- Lafore, R. **Object-Oriented Programming in C++**. 4th edition. Pearson, 2002.
- Booch, G., Rumbaugh, J., & Jacobson, I. **UML: Guia do Usuário**. 2ª edição. Campus, 2006.
- Stroustrup, B. **The C++ Programming Language**. 4th edition. Addison-Wesley, 2013.
- Sommerville, I. **Engenharia de Software**. 10ª edição. Pearson, 2019.