Disciplina: Programação Orientada a Objetos
Professor: Antonio Henrique Pinto Selvatici

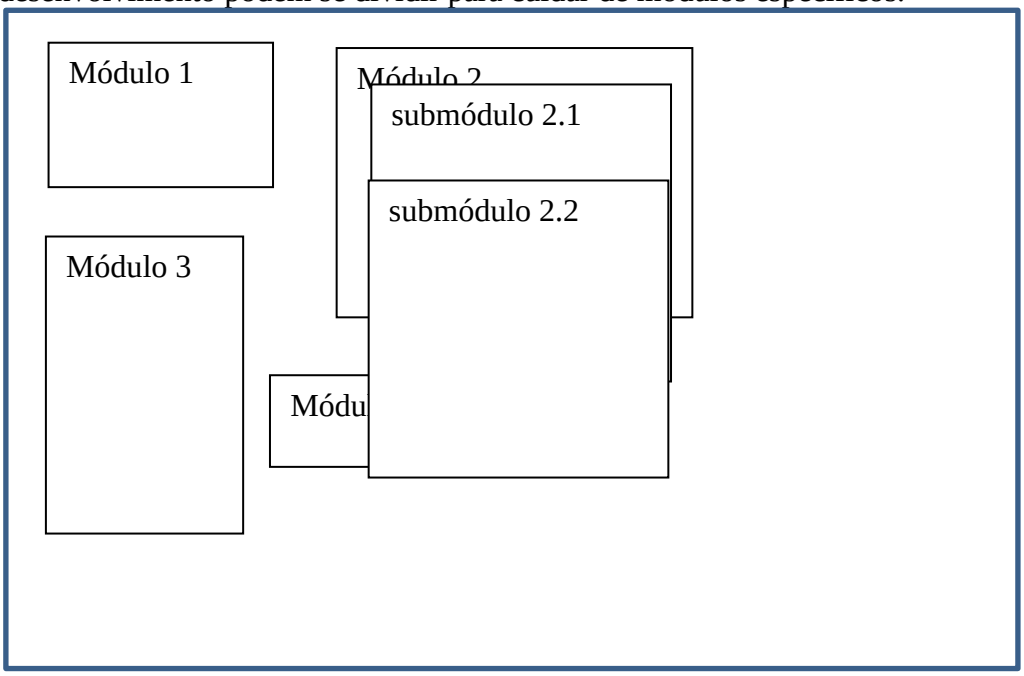
AULA 1 – Introdução à programação orientada a objetos

ROTEIRO:

- Apresentação
- Objetivos da disciplina
- Sistema de avaliação
- Plano do curso
- Bibliografia
- Introdução a POO:
    a. Pano de fundo
    b. Resolução de problema em grupo
    c. Introdução a POO
    d. Modelagem como objetos, representação UML

1. Apresentação

Importância da disciplina: capacidade de desenvolver sistemas de forma sistemática e com escalabilidade, pois permite escrever o software com base em elementos modulares e com funções específicas. Assim, os times de desenvolvimento podem se dividir para cuidar de módulos específicos.



A vantagem desses módulos sobre as funções da programação estruturada é que eles são vistos como uma representação de uma parte do mundo que está sendo representado no sistema de software, e não apenas a decomposição da lógica necessária para executar uma tarefa.

2. Objetivos da disciplina

Apresenta os conceitos e as principais características das técnicas de programação orientada a objetos, assim como terminologia e tecnologia correlatas. Exercita a programação orientada a objetos, utilizando as etapas do desenvolvimento de software orientado a objetos documentado com UML.

3. Sistema de avaliação

O curso será dividido em 3 módulos. A cada módulo serão avaliados: (1) capacidade de traduzir uma solução em termos da modelagem com diagramas UML, (2) capacidade de implementar e documentar a solução em Java.
Ao final de cada módulo, os alunos farão uma prova escrita ou um projeto prático em grupo, valendo 80% da nota. Os outros 20% resultarão da apresentação dos exercícios de laboratório, avaliados em sala de aula. Esses exercícios abordarão a modelagem UML, com diagramas de classes e comunicação, bem como implementação e documentação do código, perfazendo 20% da nota semestral.

4. Plano do curso
    - Módulo 1: Introdução à Programação Orientada a Objetos
        - Introdução a Orientação a Objetos e à modelagem UML
        - Conceitos de atributos e métodos.
        - Documentação em diagramas de classe e comunicação
        - Conceitos de encapsulamento, herança e polimorfismo
    - Módulo 2: Elementos avançados de POO em Java
        - Conceituação de abstração e interface
        - Conceituação e tratamento de exceções em Java: modelagem e implementação
        - Uso de coleções em Java: modelagem e implementação
    - Módulo 3: Criação de interfaces gráficas usando Java Swing
        - Java Swing: introdução e elementos básicos
        - Manipulação de layout e elementos avançados
        - Documentação de software com UML: diagramas de caso de uso e de sequência
        - Projeto, modelagem e documentação de software com UML
        - Implementação e documentação do projeto

5. Bibliografia

BARNES, David J.; KOLLING, Michael. Programação Orientada a Objetos com Java: uma introdução prática usando o BlueJ. São Paulo: Pearson, 2010.
FOWLER, Martin. UML essencial: um breve guia para a Linguagem-Padrão de Modelagem de Objetos. Porto Alegre: Bookman, 2005.
SCHILDT, Herbert; SKRIEN, Dale. Programação com Java: uma introdução abrangente. Porto Alegre: Bookman McGraw Hill, 2013

6. Introdução à Programação Orientada a Objetos

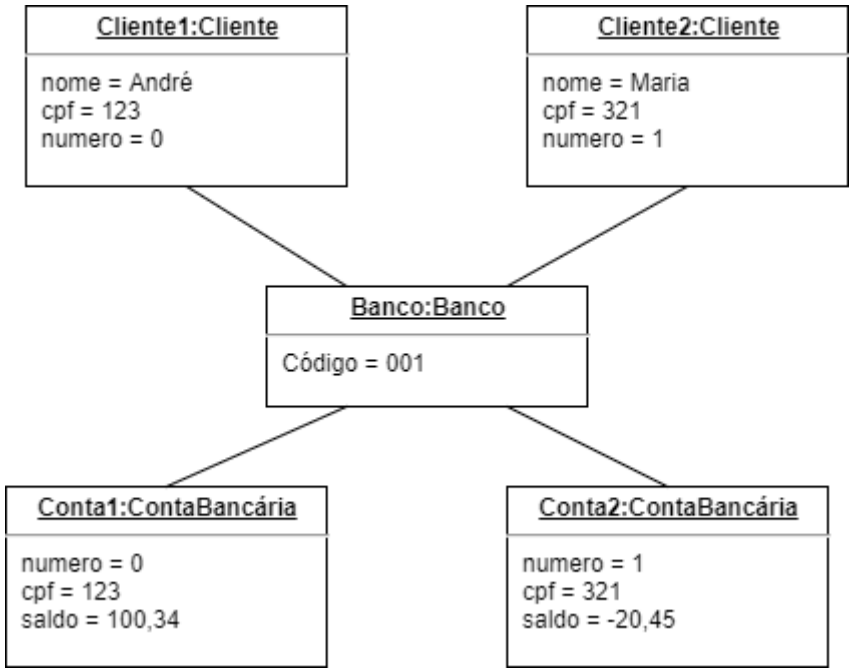
6.1. Pano de fundo:
Ao criar um programa de computador em uma linguagem orientada a objetos, você criará, em seu computador, um modelo de alguma parte do mundo. As Partes das quais o modelo é construído são os objetos que aparecem no domínio do problema. Esses objetos devem ser representados no modelo de computador que estiver sendo criado.

6.2. Dinâmica de grupo

Vamos nos dividir em grupos de 3 alunos para escrever um programa que monitora a conta bancária dos clientes de um banco. As informações que precisam ser monitoradas são: nome e CPF dos clientes e saldo e número da conta bancária. As operações necessárias ao banco são: cadastro de um novo cliente, com saldo zero, registro de saques e depósitos, e transferência de valores entre os clientes. Os dados não precisam ser salvos em arquivo (podem ficar apenas na memória do computador). Após escrever o programa, teste criando 2 clientes.
Agora vamos modificar o programa para cadastrar também o endereço dos clientes, e também o limite de crédito de cada conta. Como vocês dividiriam o programa para agilizar o desenvolvimento e a manutenção?

6.3.Introdução a POO: objetos e classes

Os objetos são formas de descrever componentes específicos de um sistema. Tudo que pode ser considerado como uma unidade, pode também ser um objeto. Assim, no nosso sistema, que objetos podemos ter?
Os objetos podem ser categorizados, ou seja, há diferentes categorias de objetos. No nosso exemplo, temos objetos que representam contas bancárias, e objetos que representam clientes. Uma classe descreve — de maneira abstrata — todos os objetos de um tipo particular. Assim, no exemplo acima podemos criar a classe "Conta Bancária" e a classe "Cliente". Podemos também ter a classe Banco, que conteria as definições relativas ao objeto que representa o banco. E, cada objeto que representa um cliente específico, com seu nome e CPF, é uma instância da classe "Cliente".
Uma vantagem da programação orientada a objetos é a possibilidade de representar em um documento os objetos (ou classes) e suas relações, tornando mais fácil o entendimento humano sobre o funcionamento do sistema. O UML (Unified Modeling Language) é um padrão para essa forma de documentação, que traz consigo uma série de diagramas que visam a modelar e documentar um sistema de software dependendo ponto de vista que se quer abordar.
Para descrever como os objetos de um sistema se relacionam, a UML emprega o Diagrama de objetos, que fornece uma fotografia dos objetos existentes e de seus estados em um determinado instante da execução do software. Assim, em determinado momento os objetos do software do banco pode ser representado como



Aqui consideramos que o há apenas um objeto da classe Banco no sistema, sendo que o banco depende ou interage com as informações da conta bancária, bem como dos clientes. Já os clientes interagem apenas com o banco, que gerencia suas contas. Dá para melhorar a modelagem? É importante que essa modelagem reflita o que ocorre no mundo (real ou imaginado) o mais fidedignamente possível, para que o processo de desenvolvimento do software ocorra o mais suavemente possível.

Assim o diagrama de objetos representa as dependências e interações dos diversos objetos. Talvez seja interessante permitirmos que o cliente converse com outro cliente para recuperar seus dados sempre que precisar fazer uma transferência. Podemos ainda pensar que uma conta pode ter mais de um CPF vinculado, como numa conta conjunta. O diagrama de objetos os ajuda a enxergar mais facilmente esses relacionamentos, antes de iniciar a implementação de fato do projeto.