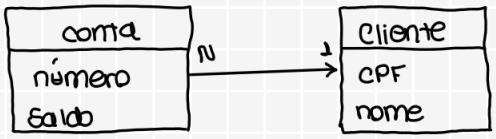
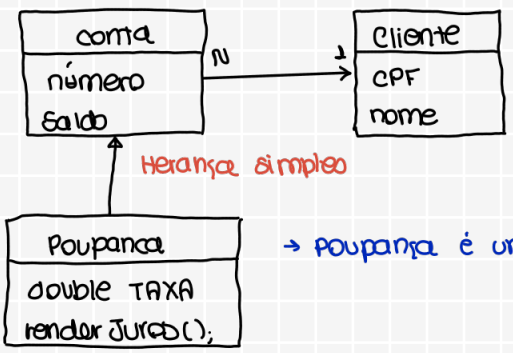
RR1 - 01 Revisão Arrays e interfaces

→ Herança: Recurso de OO que permite definir novas classes a partir de classes já existentes.
↳ Uma classe que herda de outra traz para si atributos, métodos e construtores.
↳ Caste → conversão explícita de um tipo em um outro.
↳ Há o caso de existirem métodos herdados que precisam ser sobreescritos para se adaptarem à realidade da sub-classe.
↳ Herança com redefinição de método.
↳ Precisamos manter o comportamento do método.
↳ Quando não é possível preservar o comportamento usamos o conceito de classes abstratas - classes que não podem ser instanciadas, mas permitem que a semântica de métodos herdados e sobreescritos fiquem em aberto.
↳ Quando queremos agrupar tipos por funcionalidade, usamos o recurso de interface - um contrato onde apenas serviços são descritos de forma abstrata, mas precisamos respeitar a assinatura dos métodos.

Estudo de caso → Sistema Bancário.

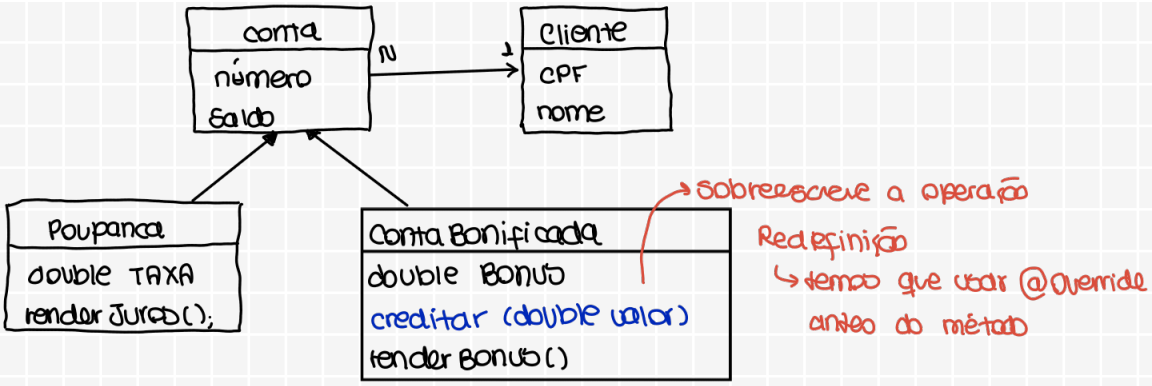


→ estendendo o sistema bancário:
- Novo tipo de conta: Poupança;
- Tem um novo serviço de rendimento de juros que acrescenta uma quantidade ao saldo.
- Como estender o sistema?
- Qual impacto causado na coleção de contas? nenhum impacto, princípio da substituição.



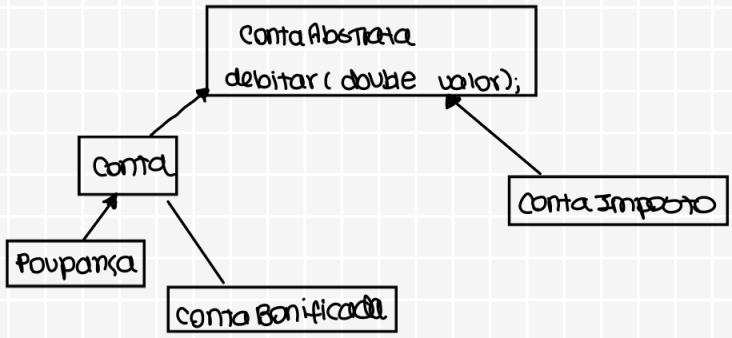
→ Poupança é uma especialidade de conta

→ "Onde uma conta é esperada, qualquer subtipo dela pode ser utilizado".
→ Estendendo o sistema Bancário:
- Novo tipo de conta: ContaBonificada;
- A cada crédito, um percentual é acumulado em um bônus até que, em algum momento, o bônus é acrescentado ao saldo;
- Como estender o sistema?
- Qual impacto causado na coleção de contas?



→ O casting é necessário quando queremos jogar um tipo genérico em um tipo específico.
→ Devemos ler o instanceOf como "é do tipo?".

→ Estendendo o sistema bancário:
- Novo tipo de conta: ContaImposto;
- Os débitos não debitam apenas o valor, mas também cobram uma taxa pela operação;
- Redefinições devem preservar a semântica.
  ↳ Precisamos de uma entidade acima em que eu não tenha uma semântica definida para o debitar.
    ↳ Conta Abstrata.



→ Interfaces:
Imagine que existam diferentes tipos de bancos:
- Banco de investimentos;
- Banco de seguros

} tipos com o mesmo tipo de serviço

Os bancos podem ser fiscalizados por um auditor:
- Auditor para banco de investimento;
- Auditor para banco de seguros.

Como usar um mesmo auditor para os dois bancos?
Criando uma interface QualquerBanco
↳ métodos abstratos

Universidade Federal de Campina Grande
Centro de Engenharia Eletrica e Informática
Departamento de Sistemas e Computação
Graduação em Ciência da Computação

Exercício de revisão de Java

Objetivo: Relembrar os conceitos de Java relacionados a Interfaces e Arrays. É importante que você resolva com atenção pois diversos conceitos e práticas de programação serão essenciais no decorrer da disciplina. Dessa forma, se você tiver qualquer dúvida, procure o professor ou o monitor. Utilize o horário da aula para resolver os exercícios.

1. Para que serve o recurso de interface em Java? Quais situações/cenários requerem seu uso?
2. Baseando-se nos itens abaixo e usando os recursos de orientação a objeto que você conhece, implemente entidades da forma que você julgar mais adequado.
   - Como sabemos, cada forma geométrica tem sua área calculada através de uma fórmula específica. Considerando apenas algumas delas, temos:
     a. Triângulo: (base*altura)/2
     b. Retângulo: (base*altura) aqui não tinha nada para fazer
     c. Quadrado: lado^2 (um tipo especial de retangulo)
     d. Círculo: π*r^2 (Em Java π é definido pela constante Math.PI)
   - Imagine uma classe com um método main e que faz uso de uma forma geométrica. O objetivo dessa classe é mostrar a área da forma geométrica (com System.out.println mesmo). Tente usar recursos de forma que essa classe não conheça os detalhes de como cada forma geométrica calcula sua área.
3. Note que as implementações de Produto (Perecivel e NaoPerecivel) já foram fornecidas com este projeto. A ideia agora é implementar coleções de objetos do tipo ProdutoPerecivel e ProdutoNaoPerecivel que utilizam arrays para manipular os objetos armazenados. Observe que já existem implementações incompletas dessas coleções: RepositorioProdutoPerecivelArray e RepositorioProdutoNaoPerecivelArray. Complete as implementações dos métodos conforme os comentários.

4. Observe suas implementações de repositório e responda aos seguintes questionamentos:
   - Elas possuem algo em comum?
   - Existe replicação de código entre elas?
   - Seria possível ter apenas uma coleção para todos os tipos de produtos? Em caso positivo, implemente essa solução.

5. Imagine agora que você deseja flexibilizar a implementação de repositórios em seu sistema para permitir que outras implementações de repositório também possam ser utilizadas. Por exemplo, você dispõe de uma implementação de repositório que guarda os objetos em um ArrayList e deseja usá-lo em seu sistema. Isso é possível? Em caso positivo, implemente a solução e procure utilizar os recursos de refatoramento do eclipse para fazer isso.

6. Em seguida, faça as devidas adequações na classe RepositorioProdutoArrayList para este novo cenário e implemente seus métodos. Se tiver alguma dúvida, consulte a documentação da classe ArrayList.java.

(FO) Serve para criar um contrato / padrão de comportamento, que promove abstração e polimorfismo. Usamos interfaces quando há a necessidade de múltiplas implementações de um mesmo conjunto de método.

Não entendi o conceito, mas não mudamos nada na classe.