Programação OO

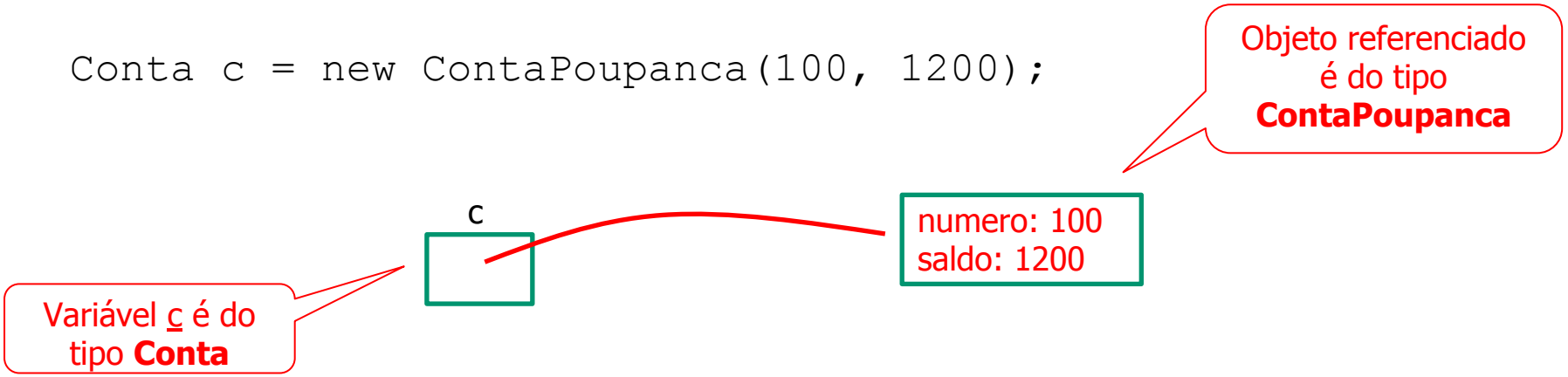
7ª aula

Prof. Douglas Oliveira
douglas.oliveira@prof.infnet.edu.br

Operador instanceof

- Com o mecanismo de herança, nem sempre a instância referenciada por uma variável é do tipo dessa variável.

```
Conta c = new ContaPoupanca(100, 1200);
```



Operador instanceof

- Para verificarmos a instância referenciada por uma variável, em tempo de execução, usamos o operador instanceof.

```
<variável> instanceof <nome da classe>
```

```
Conta c = new ContaPoupanca(100, 1200);
if (c instanceof ContaEspecial)
   System.out.println("Eu sou uma conta especial");
if (c instanceof ContaPoupanca)
   System.out.println("Eu sou uma conta poupança");
```

Operador instanceof

- Exemplo:

```
public class Teste {
  public static void main(String[] args) {
    Conta[] contas = new Conta[3];
    contas[0] = new Conta(100, 500);
    contas[1] = new ContaPoupanca(200, 1200);
    contas[2] = new ContaEspecial(300, 500, 1000);
    for (int i = 0; i < contas.length; i++)
       if (contas[i] instanceof ContaPoupanca)
         ((ContaPoupanca c).render(10);
}
```

Se for uma conta poupança então manda render 10%.

Classes Abstratas

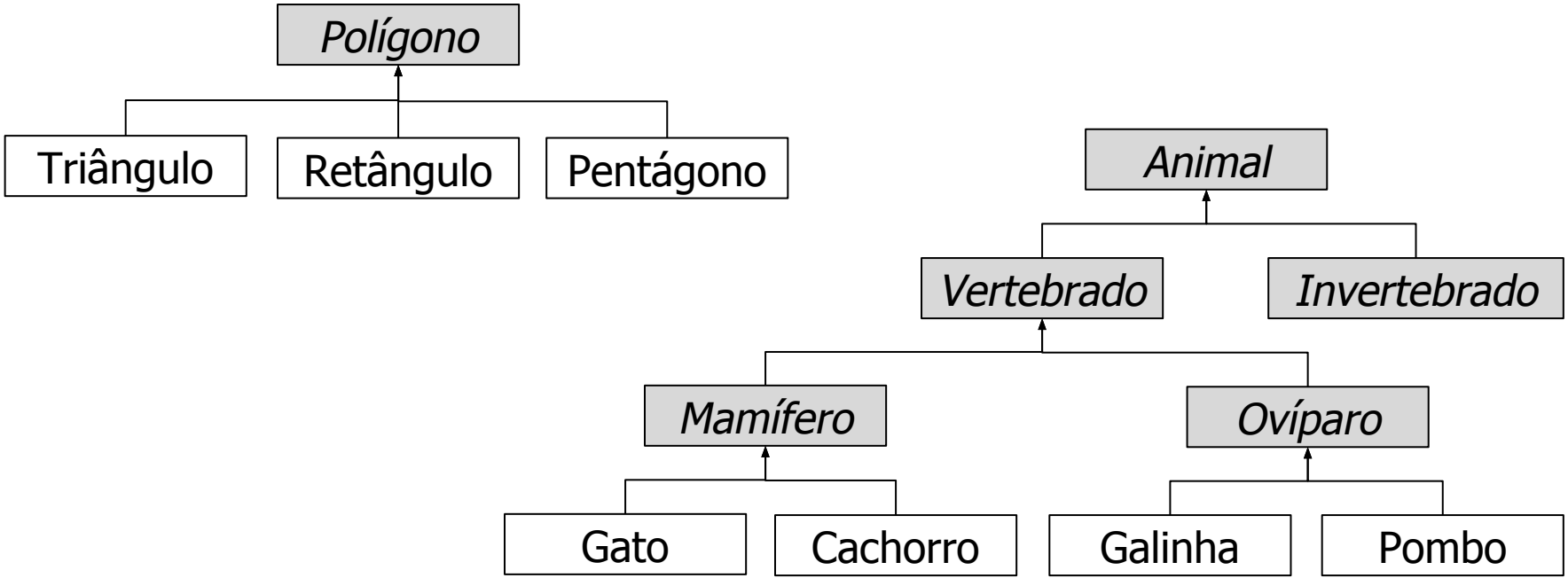
- No exemplo da concessionária temos:
  - ✔ Carro é um tipo de *Veículo*
  - ✔ Caminhão é um tipo de *Veículo*
  - ✔ *Veículo* possui placa, modelo e ano de fabricação
- Perguntas:
  - ✔ Se algum cliente entrar na concessionária e pedir para comprar um *Veículo*, o vendedor saberá o que deve vender?
  - ✔ Se a concessionária anunciar que está vendendo um Veículo, o cliente tem condições de saber exatamente o que está comprando ?
- Isso ocorre porque *Veículo* é uma abstração para Carro e Caminhão.

Classes Abstratas

- Classes Abstratas são classes que representam abstrações (conceitos) e não objetos concretos do mundo que estamos representando.
- São usadas como moldes para a criação de outras classes e podem conter atributos e comportamentos.
- No exemplo da concessionária *Veículo* é uma classe abstrata.
- *Veículo* define um molde a partir do qual as classes concretas Carro e Caminhão são definidas.

Classes Abstratas

- Outros exemplos:



Classes Abstratas

- Classes abstratas são definidas em Java com o uso da palavra abstract na sua definição.
- Por não representarem objetos concretos, classes abstratas não podem ser criadas com o operador new.

```
public abstract class Veiculo {
  protected String modelo;
  protected int anoFabricacao;
  public Veiculo(String modelo, int anoFabricacao)
  {
    this.modelo = modelo;
    this.anoFabricacao = anoFabricacao;
  }
}
Veiculo v = new Veiculo("gol", 2011);
```

Erro de compilação !

Métodos Abstratos

- Quando definimos uma classe abstrata, muitas vezes nos deparamos com métodos que fazem sentido para aquela classe, mas não temos como implementá-lo, porque a classe abstrata representa apenas um conceito.

```
public abstract class Poligono {
    public double area() {
        ???????
    }
}
```

O cálculo da área de um polígono faz sentido, mas como implementar esse cálculo aqui?

Métodos Abstratos

- Nesse caso, definimos esses métodos como métodos abstratos.

```
public abstract class Poligono {
    public abstract double area();
}
```

Métodos abstratos não possuem implementação!

Métodos Abstratos

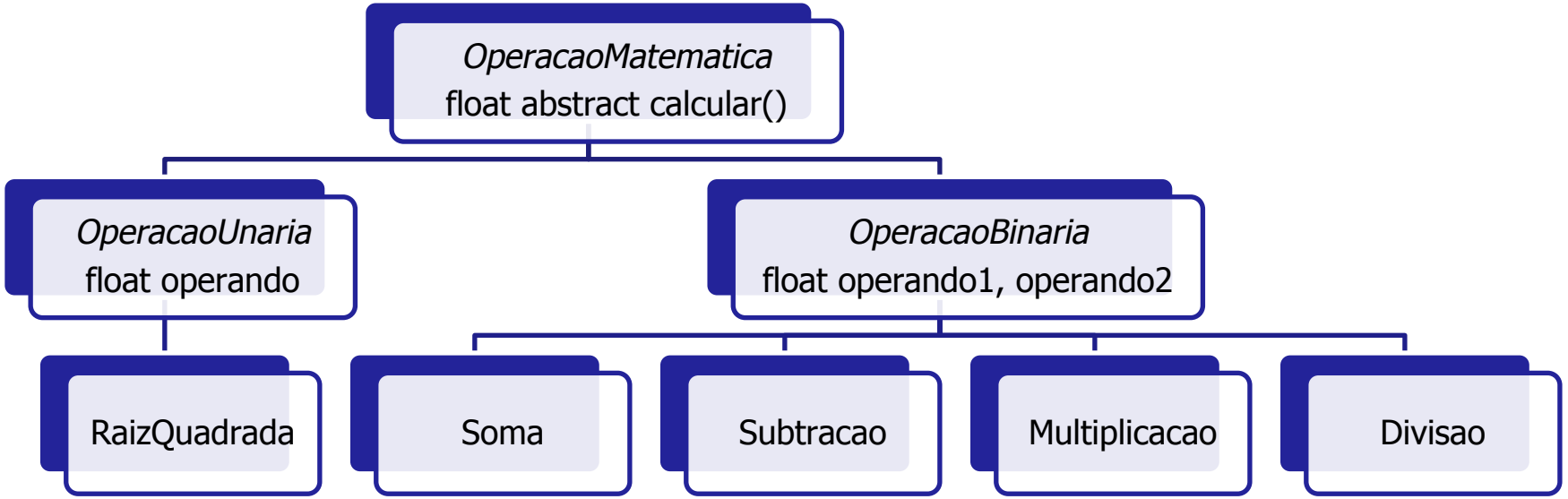
- As classes concretas que herdam da classe abstrata devem, obrigatoriamente , implementar os métodos abstratos.

```
public class Retangulo extends Poligono {
  private int x, y, largura, altura;
  public Retangulo (int x, int y, int largura, int altura){
    this.x = x;
    this.y = y;
    this.largura = largura;
    this.altura = altura;
  }
  public double area() {
    return largura * altura;
  }
}
```

As classes concretas que herdam de *Poligono* devem implementar o método **area()**

Exercícios

- Exercício 36: crie a seguinte hierarquia de classes



Modificador *final*

- Para evitar que sejam criadas subclasses a partir de uma classe podemos defini-la com o modificador final.
- Exemplo:

```
public class Animal {
  public void emiteSom() {
    System.out.println("???"); }
}
public final class Gato extends Animal {
  public void emiteSom() {
    System.out.println("miado"); }
}
public class Siames extends Gato {
}
```

Se tentarmos criar uma sub-classe de **Gato** haverá um erro de compilação, porque Gato está definido como **final**.

Modificador *final*

- Podemos usar o modificador final para evitar que um método seja sobrescrito pelas subclasses. Exemplo:

```
public class Animal {
  public void emiteSom() {
    System.out.println("???"); }
}
public class Mamifero extends Animal {
  public final boolean bebeLeite() {
    return true; }
}
public class Gato extends Mamifero {
  public void emiteSom() {
    System.out.println("miado"); }
  public boolean bebeLeite() {
    return false; }
}
```

Se tentarmos sobrescrever o método **bebeLeite()** haverá um erro de compilação, pois ele está definido, na superclasse, como **final**.

Exercícios

- Exercício 37: usando as classes implementadas anteriormente, implemente a classe Banco que possui: código, nome e um conjunto de contas (pode ser conta corrente, conta poupança ou conta especial). Implemente métodos para:

  a) Adicionar uma conta ao banco

  b) Remover uma conta do banco

  c) Sacar dinheiro de uma conta

  d) Depositar dinheiro em uma conta

  e) Listar todas as contas com número e saldo

  f) Imprimir o total existente no banco (somatório dos saldos)

Exercícios

- Exercício 38: em uma universidade:
  1. Todos os alunos são cadastrados com nome e matrícula.
  2. Cada disciplina possui código, nome, nº de créditos e valor mensal/crédito.
  3. Cada turma possui um código e está associada à disciplina que oferece.
  4. Uma turma pode ter, no máximo, 30 alunos inscritos.
  5. Os alunos podem se matricular em até 20 créditos.
  6. O valor da mensalidade do aluno é calculado somando o valor de cada disciplina na qual ele está matriculado.
  7. Existem alunos bolsistas que recebem um percentual de desconto no valor da mensalidade.

  Crie as classes necessárias para modelar o cenário acima e implemente métodos para:

  a) Matricular um aluno em uma turma
  b) Listar a disciplina e os alunos de uma turma
  c) Calcular o valor a ser pago mensalmente pelo aluno

  Importante: perceba que uma turma possui um conjunto de alunos e que um aluno possui um conjunto de turmas nas quais está inscrito.

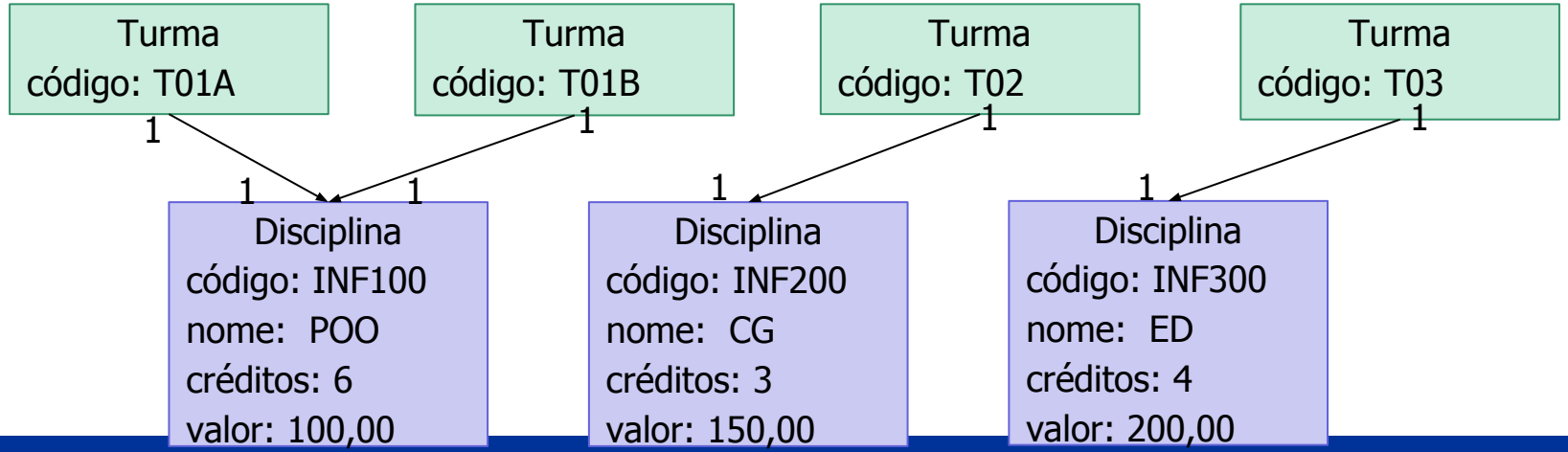
Exercícios

- Exercício 38:

Disciplina	Disciplina	Disciplina
código: INF100	código: INF200	código: INF300
nome: POO	nome: CG	nome: ED
créditos: 6	créditos: 3	créditos: 4
valor: 100,00	valor: 150,00	valor: 200,00

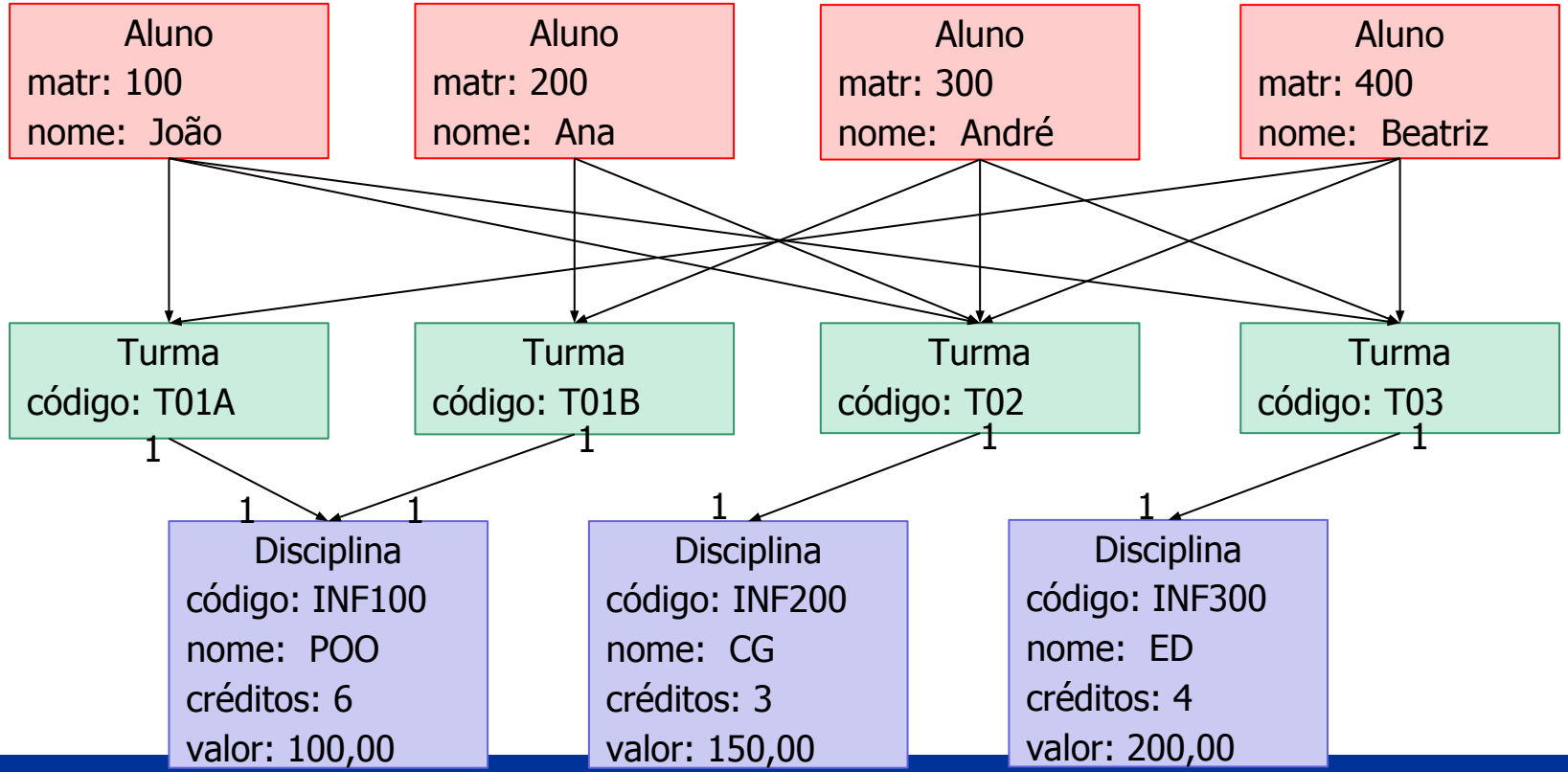
Exercícios

- Exercício 38:



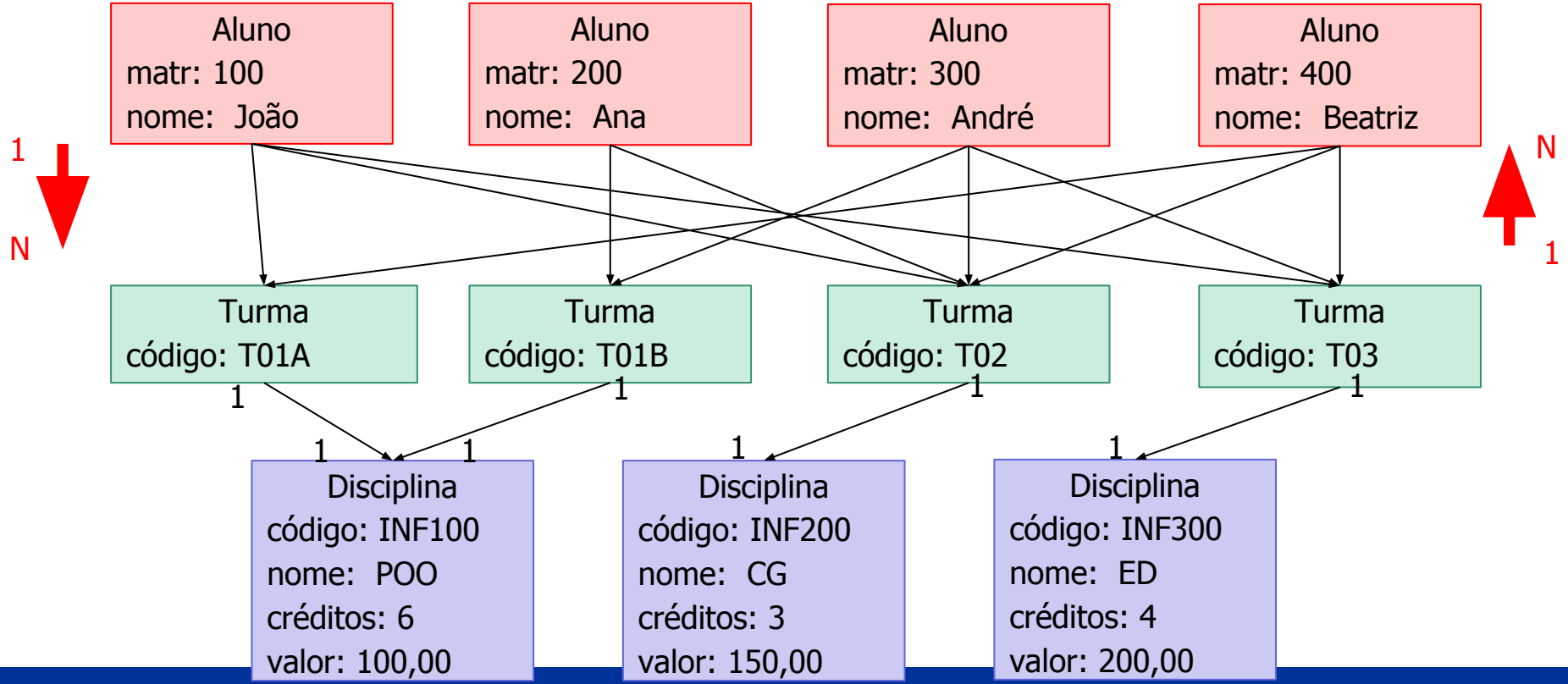
Exercícios

- Exercício 38:



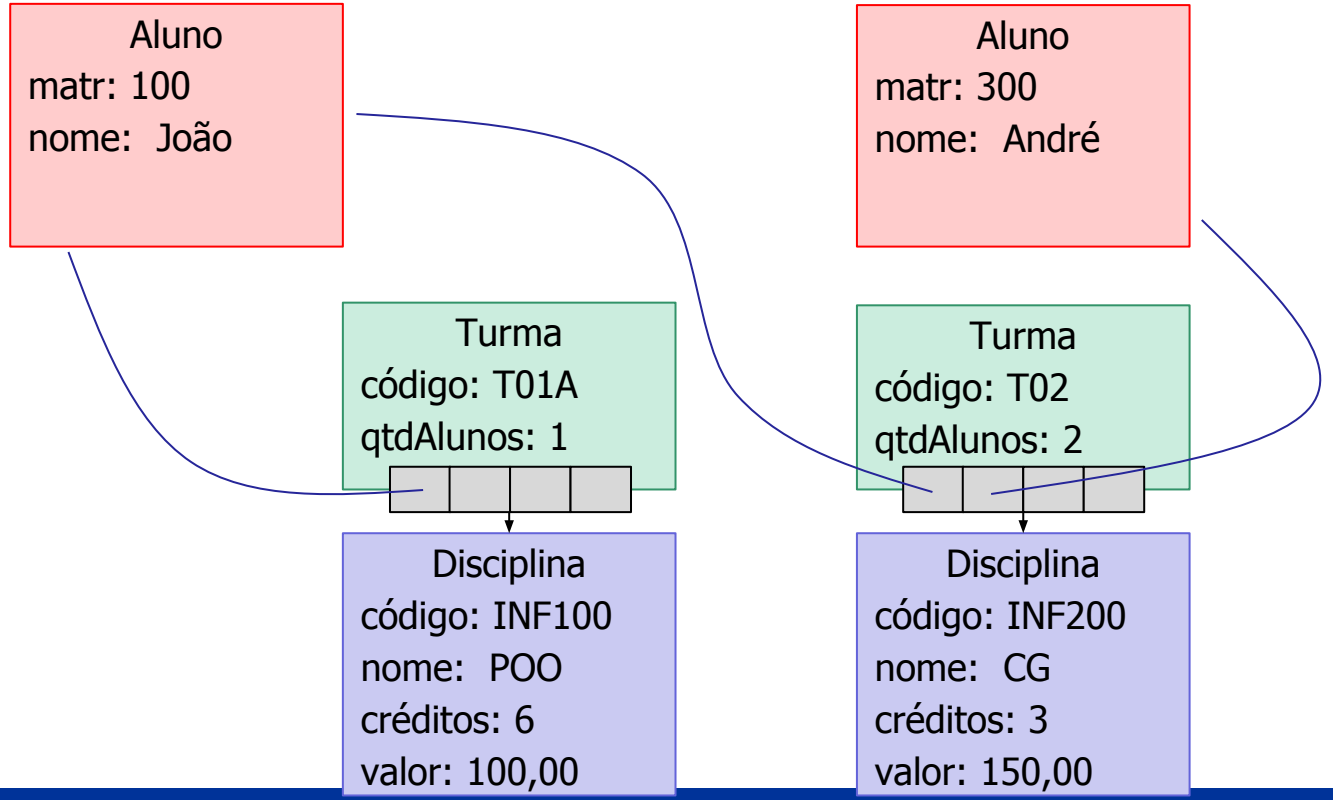
Exercícios

- Exercício 38:



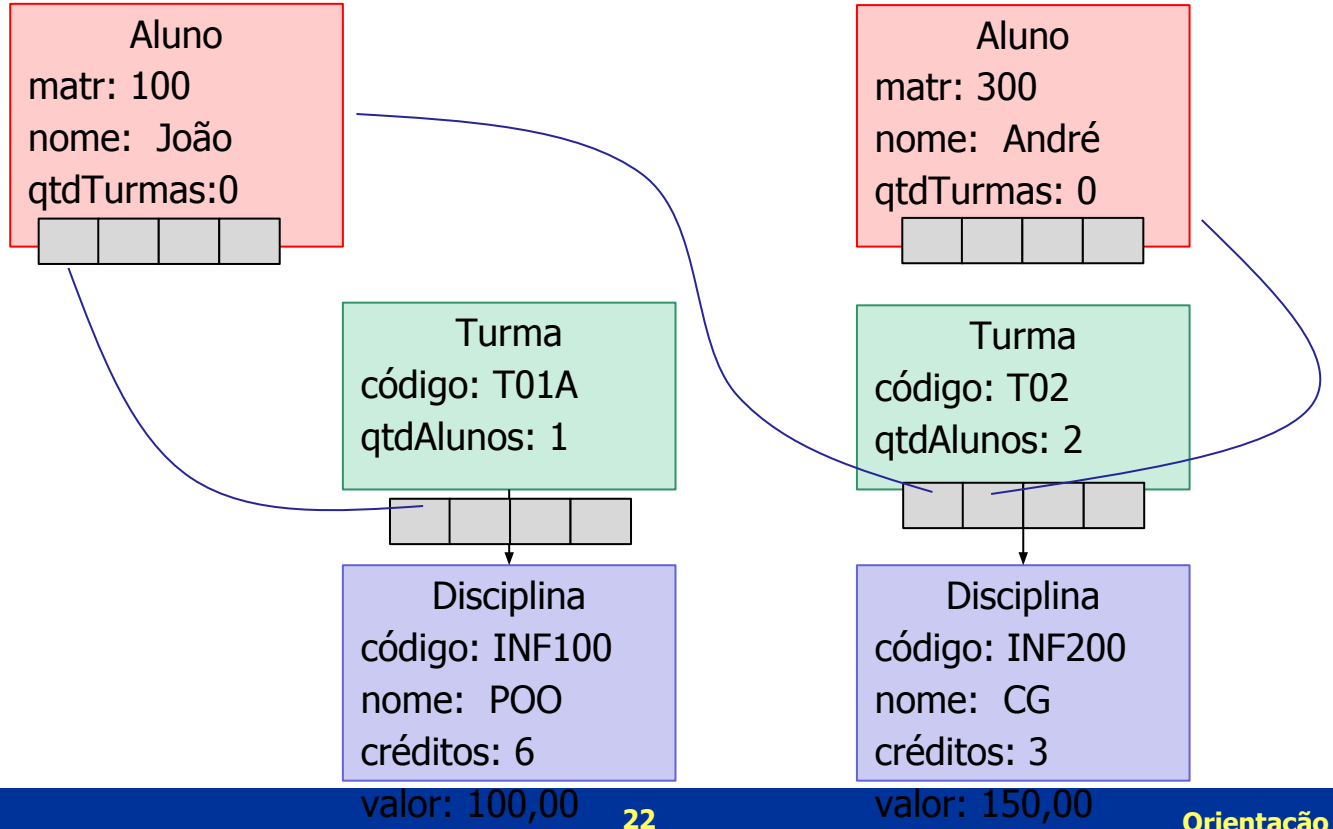
Exercícios

- Exercício 38:



Exercícios

- Exercício 38:



Exercícios

- Exercício 38: