Universidade Federal de Ouro Preto

CSI032 - Programação de Computadores II

Associação, Agregação e Composição

Professor: Dr. Rafael Frederico Alexandre
Contato: rfalexandre@decea.ufop.br
Colaboradores
Renan Saldanha
Eduardo Matias Rodrigues

Agenda

Agenda

Diagrama de Classes

- Modelo para representação da estrutura e dos relacionamentos entre as classes;
- Uma classe é representada por um retângulo com 3 divisões:

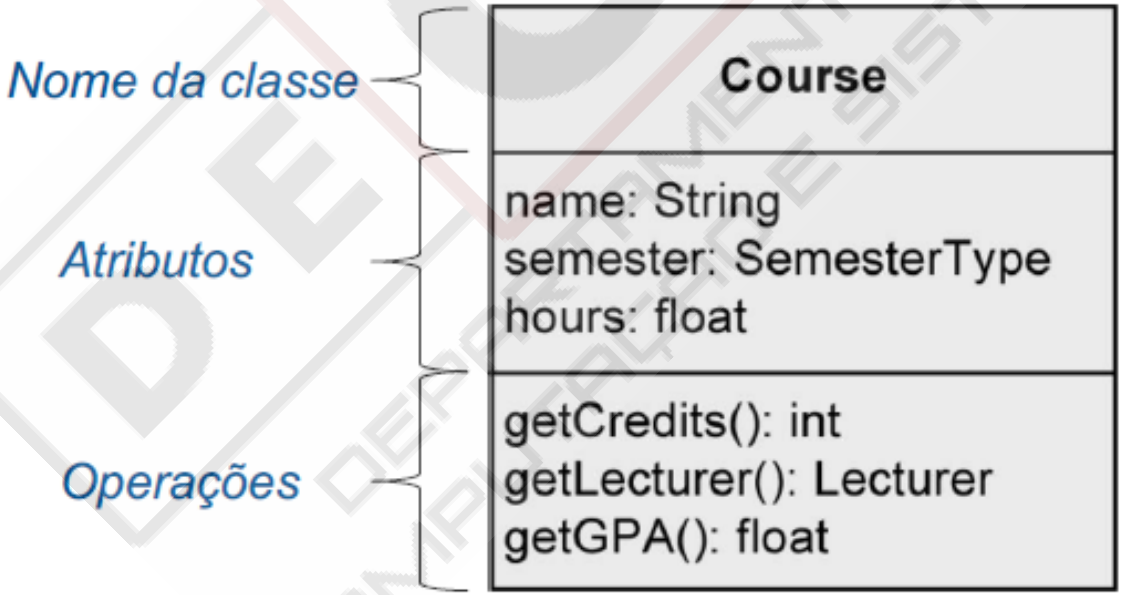


Figura: [of Technology, 2015]

Atributos

Representação dos atributos

- (Visibilidade) NomeDoAtributo : Tipo
- Visibilidade:
  - + : public (todo mundo);
  - - : private (só a classe);
  - # : protected (mesmo pacote, classe e suas filhas);
  - ~ default (mesmo pacote);

Métodos

Representação dos métodos

- (Visibilidade) NomeDoMetodo(TiposDosParametros) : Tipo do Retorno;
  - public (+), private (-), protected (#)
- Exemplo:
  - + getName() : String
  - + setName(String) : void
  - + calculaPotencia(double, double) : double

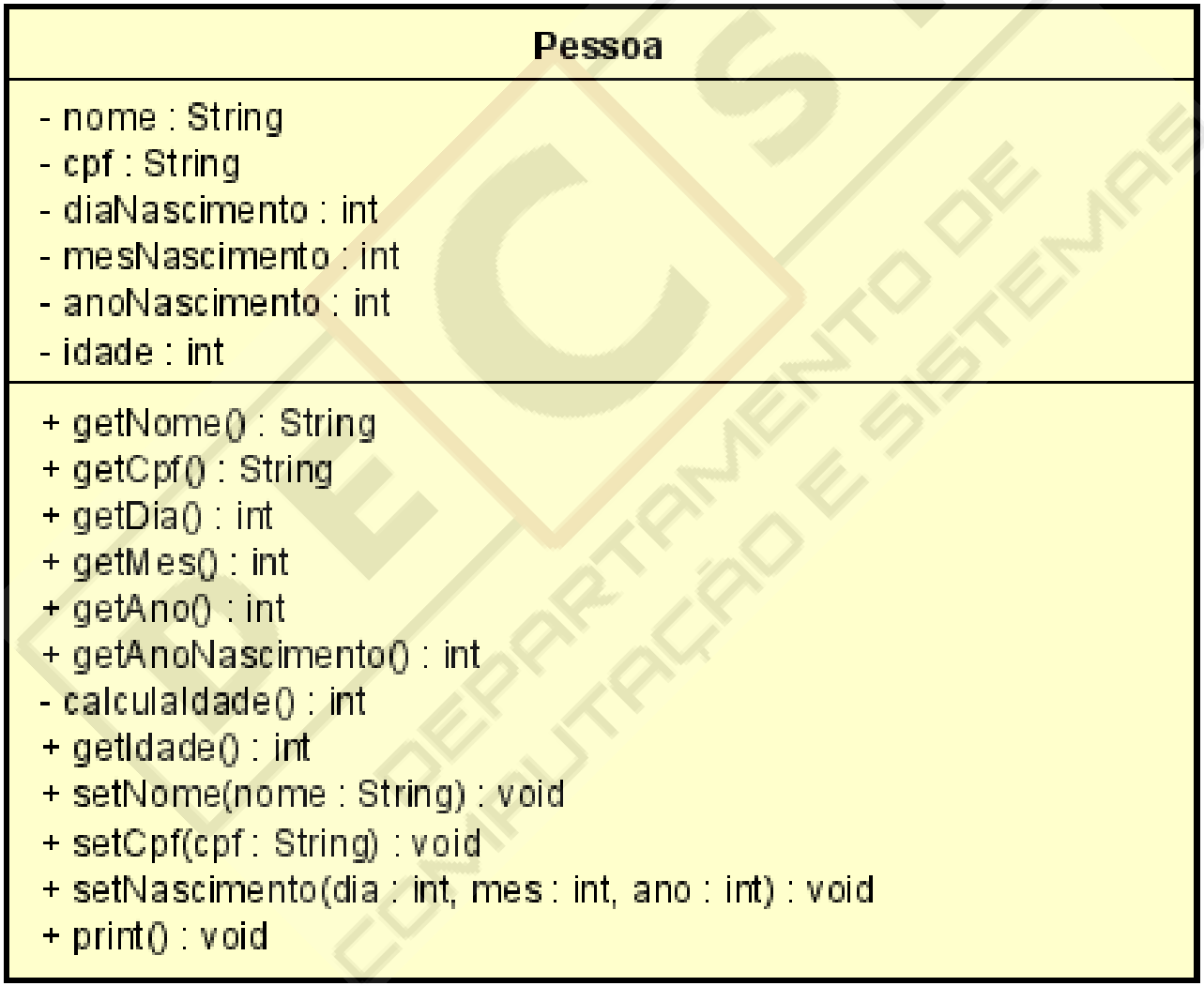
Diagrama de Classe

Pessoa: Descrição em linguagem natural

- Crie uma classe *Pessoa*, uma pessoa deve ter um atributo nome (private), um cpf (private), uma data de nascimento (private) e uma idade (private);
- Crie os métodos *Getters* e *Setters* para todos os atributos da classe *Pessoa*;
  - O método getIdade() deve chamar o método calculaIdade(), que é um método privado que calcula a idade da pessoa com base no atributo data de nascimento e o ano atual;
- Crie um método print() que exibe os dados das pessoas.

Diagrama de Classe

Pessoa: Representação completa com Diagrama de Classe



Agenda

Associação
Introdução

- Em um projeto teremos diversas classes;
- Associação modela os possíveis relacionamentos entre as instâncias dessas classes;
- Representação no diagrama de classes:

Exemplo

- Associação entre médico e paciente;
  - Um médico pode atender nenhum ou vários pacientes;
  - Um paciente pode ter nenhuma ou várias consultas marcadas;

Figura: Associação entre médico e paciente

Exemplo (Cont.)

- Um médico pode realizar várias consultas e um paciente também pode ter várias consultas. Associação de **muitos para muitos**.
- Podemos associar médico e paciente através de uma terceira classe, a classe *Consulta*.
- A classe consulta é dita uma *Classe Associativa*.

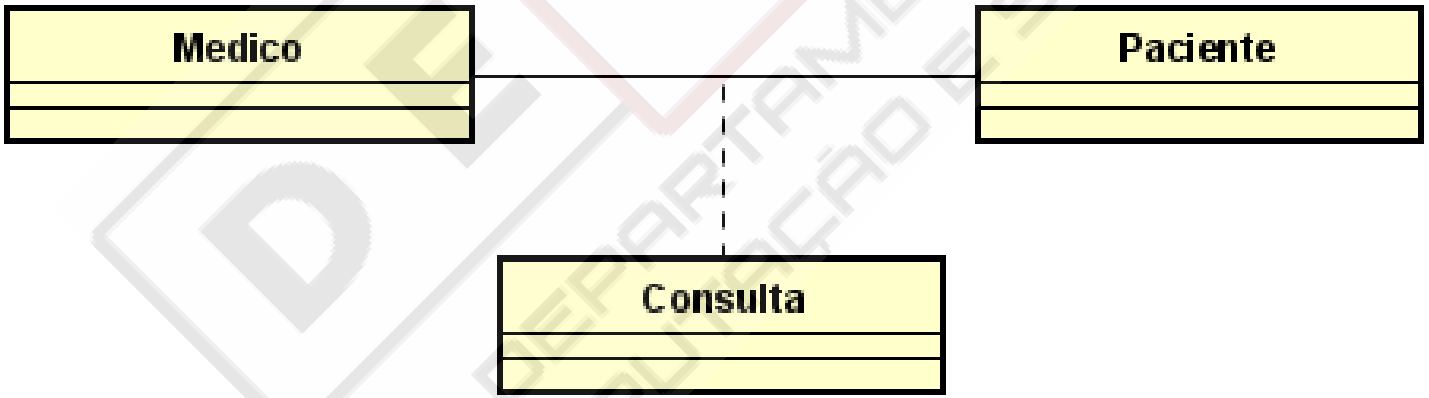


Figura: Associação entre médico e paciente

Associação

Exemplo (Cont.) : Implementação da classe *Médico*

```
public class Medico {
    private String nome;
    private String especialidade;

    public Medico(String nome, String especialidade) {
        this.nome = nome;
        this.especialidade = especialidade;
    }
    // (...) Getters e Setters (...) //
```

Associação

Exemplo (Cont.) : Implementação da classe *Paciente*

```
public class Paciente {

    private String nome;
    private String cpf;
    private int cadastro;

    public Paciente(String nome, String cpf, int cadastro) {
        this.nome = nome;
        this.cpf = cpf;
        this.cadastro = cadastro;
    }
    // (...) Getters e Setters (...) //
```

Associação

Exemplo (Cont.) : Implementação da classe *Consulta* part. 1

```
public class Consulta {

    private int dia, mes, ano;
    private int hora, min;
    private Medico medico;
    private Paciente paciente;
    private String receitaMedica;

    public Consulta(int dia, int mes, int ano, int hora, int min, Medico medico, Paciente paciente) {
        this.dia = dia;
        this.mes = mes;
        this.ano = ano;
        this.hora = hora;
        this.min = min;
        this.medico = medico;
        this.paciente = paciente;
    }
```

Associação

Exemplo (Cont.) : Implementação da classe *Consulta* part. 2

```
// Remédios completamente aleatórios
public void realizarConsulta() {
    String[] remedios = {"Dipirona", "Benegripe", "Histamin", "Dorflex"};
    Random random = new Random();
    this.receitaMedica = remedios[random.nextInt(3)];
}

public void exibe() {
    System.out.println("Data: " + this.dia + "/" + this.mes + "/" + this.ano);
    System.out.println("Hora: " + this.hora + ":" + this.min);
    System.out.println("Médico: " + this.medico.getNome() + "| Especialidade: " + this.medico.getEspecialidade());
    System.out.println("Paciente: " + this.paciente.getNome());
    System.out.println("Receita: " + this.receitaMedica);
    System.out.println();
}
```

Associação

Exemplo (Cont.) : Driver, teste da aplicação, part. 1

```
public static void main(String[] args) {

    // Médicos
    Medico eduardoM = new Medico("Eduardo M. Rodrigues", "Pediatra");
    Medico aliceM = new Medico("Alice", "Cardiologista");
    Medico mariaM = new Medico("Maria", "Dermatologista");

    // Pacientes
    Paciente fernandoP = new Paciente("Fernando da Silva", "12345678", 1);
    Paciente larissaP = new Paciente("Larissa Rodrigues", "987456111", 2);
    Paciente lauraP = new Paciente("Laura Marias Martins", "74185296", 3);
    Paciente savioP = new Paciente("Savio Cavati", "753159842", 4);
    Paciente gustavoP = new Paciente("Gustavo Gonsalves", "74196325", 5);
```

Associação

Exemplo (Cont.) : Driver, teste da aplicação, part. 2

```
// Consultas
ArrayList<Consulta> consultas = new ArrayList<>();
Consulta c1 = new Consulta(11, 10, 2020, 15, 30, aliceM, fernandoP);
Consulta c2 = new Consulta(11, 10, 2020, 16, 00, aliceM, larissaP);
Consulta c3 = new Consulta(11, 10, 2020, 16, 00, eduardoM, lauraP);
Consulta c4 = new Consulta(11, 10, 2020, 17, 00, eduardoM, savioP);
Consulta c5 = new Consulta(11, 10, 2020, 16, 00, eduardoM, gustavoP);
Consulta c6 = new Consulta(12, 12, 2020, 16, 00, mariaM, fernandoP);
Consulta c7 = new Consulta(12, 12, 2020, 18, 00, mariaM, larissaP);
consultas.add(c1);
consultas.add(c2);
consultas.add(c3);
consultas.add(c4);
consultas.add(c5);
consultas.add(c6);
consultas.add(c7);
```

Associação

Exemplo (Cont.) : Driver, teste da aplicação, part. 3

```
// Realizando as consultas
for(int i=0; i<consultas.size(); i++) {
    Consulta corrente = consultas.get(i);
    corrente.realizarConsulta();
}
// Exibindo o histórico de consultas
for (int i=0; i<consultas.size(); i++) {
    Consulta corrente = consultas.get(i);
    corrente.exibe();
}
```

Associação

Exemplo (Cont.) : Saída

```
Data: 11/10/2020
Hora: 15:30
Médico: Alice| Especialidade: Cardiologista
Paciente: Fernando da Silva
Receita: Benegripe

Data: 11/10/2020
Hora: 16:0
Médico: Alice| Especialidade: Cardiologista
Paciente: Larissa Rodrigues
Receita: Dipirona

Data: 11/10/2020
Hora: 16:0
Médico: Eduardo M. Rodrigues| Especialidade: Pediatra
Paciente: Laura Marias Martins
Receita: Dipirona
```

Associação

Exemplo (Cont.) : Saída

```
Data: 11/10/2020
Hora: 17:0
Médico: Eduardo M. Rodrigues| Especialidade: Pediatra
Paciente: Savio Cavati
Receita: Benegripe

Data: 11/10/2020
Hora: 16:0
Médico: Eduardo M. Rodrigues| Especialidade: Pediatra
Paciente: Gustavo Gonsalves
Receita: Dipirona

Data: 12/12/2020
Hora: 16:0
Médico: Maria| Especialidade: Dermatologista
Paciente: Fernando da Silva
Receita: Benegripe
```

Associação

Exemplo (Cont.) : Saída3

```
Data: 12/12/2020
Hora: 18:0
Médico: Maria| Especialidade: Dermatologista
Paciente: Larissa Rodrigues
Receita: Dipirona
```

Agenda

Associação
Agregação

- Forma especial de associação;
- Usada para expressar que uma classe é parte da outra;
  - O objeto recebe o objeto agregado já estânciado;
  - O objeto não é responsável pela criação e destruição do objeto agregado;
  - O objeto agregado é criado fora da classe que o agrega;
- Representação no diagrama de classes:

Figura: Objeto da Classe B tem um objeto da Classe A

Associação
Exemplo de Agregação

- Considere uma classe Pessoa e uma classe Carro (representa o modelo de um carro);
- Uma pessoa **tem um** carro de um determinado modelo;
- Outras pessoas podem ter um carro do mesmo modelo;
  - Carro não depende de uma pessoa para existir.
- Representação no diagrama de classes:

Associação
Exemplo de agregação (Cont.)

```java
package br.ufop.pessoa;

import br.ufop.carro.Carro;

public class Pessoa {
    private String nome;
    private String cnh;
    private Carro carro;

    public Pessoa(String nome, String cnh, Carro carro) {
        this.nome = nome;
        this.cnh = cnh;
        this.carro = carro; /*Aqui um exemplo de agrecão,
        pois a classe
        pessoa não vai contruir um carro
        no seu construtor,
        o carro é um objeto recebido já instânciado*/
    }

    @Override
    public String toString() {
        return "Pessoa: nome=" + nome + ", cnh=" + cnh + ", tem " + carro + "]";
    }

}
```

```java
package br.ufop.test;

import br.ufop.carro.Carro;
import br.ufop.pessoa.Pessoa;

public class Teste {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Carro carro = new Carro("punto", 2015, "preto");
        Pessoa pessoa = new Pessoa("José", "23456-3", carro);
        System.out.println(pessoa);

    }

}
```

Figura: Exemplo de agregação, para ter acesso ao exemplo completo acesse: `https://github.com/renansald/TutoriaProgII`

Agregação
Exercício

[Caelum, 2017] [Adap.]

- Crie uma classe *Cliente*, um cliente tem nome, cpf e uma profissão.
- Crie uma classe *ContaBancaria*, uma conta bancária **tem um** titular (que é um cliente), número, saldo e um limite. A conta deve permitir ao cliente:
  - Sacar uma determinada quantia de sua conta;
  - Fazer um depósito em sua conta;
- Crie uma classe *Agência*, uma agência possui um id, banco, endereço e uma *lista de contas bancárias*. Crie um método print() que exiba as informações de todas as contas cadastradas.
- Crie também os métodos e atributos que julgar necessário.

Agenda

Associação
Composição

- Tipo de relacionamento todo/parte;
- O objeto todo é responsável por instânciar os objetos parte que fazem parte do mesmo;
- Se o objeto todo for destruído, todos os objetos parte também são destruídos
- Representação no diagrama de classes:

Figura: Objeto da Classe A compõe Objeto da Classe B

Associação
Exemplo de composição

- Considere as classes Pessoa e Endereço;
- Uma pessoa tem um determinado endereço, caso ela deixe de existir, seu endereço também deverá ser apagado.
  - Não existem regras de quando se deve usar uma agregação ou composição, depende da sua interpretação do problema. Faz sentido um endereço deixar de existir quando seu dono é apagado? Depende do problema e como você irá implementá-lo!
- Representação no diagrama de classes:

Figura: Um endereço pertence a uma única pessoa

Associação
Exemplo de composição (Cont.)

```java
package br.ufop.pessoas;

public class Pessoa {
    private String nome;
    private String cpf;
    private Endereco endereco;



    public Pessoa(String nome, String cpf, String rua, String bairro, int numero, String cidade, String cep) {
        super();
        this.nome = nome;
        this.cpf = cpf;
        this.endereco = new Endereco(rua, bairro, numero, cidade, cep);/*Objeto criado dentro da classe,
        quando o objeto da classe pessoa deixar de existir
        o objeto do endereco também deixa de existir*/
    }


    @Override
    public String toString() {
        return "nome=" + nome + ", cpf=" + cpf + ", " + endereco;
    }
}
```

```java
package br.ufop.teste;

import br.ufop.pessoas.Pessoa;

public class Teste {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Pessoa pessoa = new Pessoa("João", "1111111", "Rua Pedro I", "São João", 110, "João Pessoa", "35900-000");
        System.out.println(pessoa);

    }
}
```

Figura: Exemplo de composição, para ter acesso ao exemplo completo acesse: `https://github.com/renansald/TutoriaProgII`

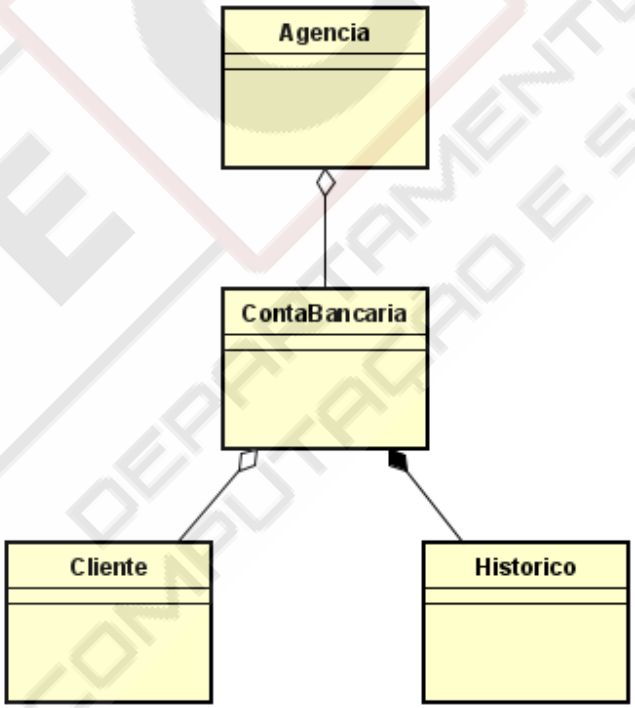
Composição
Exercício

[Caelum, 2017] [Adap.]

- ... Continuação do exercício de agregação;
- Crie a classe *Histórico*, que representa o histórico de uma conta bancária. O histórico têm uma data na qual a conta foi aberta e uma lista de transações (saque ou depósito);
- Modifique a classe *Conta* para que Histórico componha ela;
- Crie um método na classe *Conta* que exiba o extrato da conta (note que os métodos saca e deposita deverão ser alterados);
- Crie um driver (main) para testar sua aplicação.

Diagrama de classes

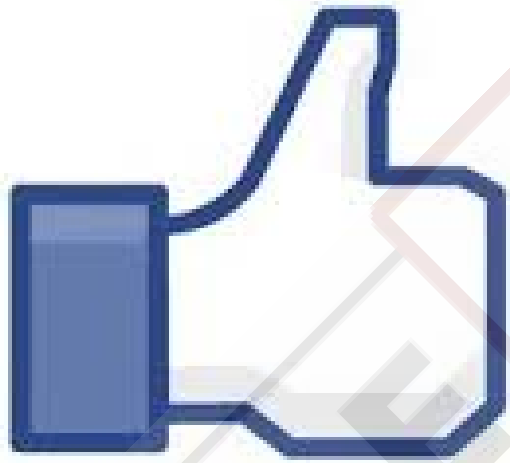
- Conta bancária **tem um** titular (que é um cliente);
- Histórico **compõe** Conta bancária;
- Agência possui uma lista de contas;
- Atributos, métodos e multiplicidade foram omitidos.



Agenda

Considerações

- Dicas de pesquisas:
  - UML (Unified Modeling Language);
  - Tipos de relacionamentos entre classes presentes na UML.
- Software utilizado para criar os diagramas de classes: Astah UML. Pode ser adquirico sob licença estudantil, link:
  - https://astah.net/downloads/



MUITO OBRIGADO!!!

Referncias Bibliogrficas I

Caelum (2017).
Java e orientação a objeto.
https://www.caelum.com.br/download/caelum-java-objetos-fj11.pdf.

of Technology, V. U. (2015).
Object-oriented modeling.
https://www.big.tuwien.ac.at/.