Universidade Federal de Ouro Preto

CSI032 - Programação de Computadores II

Associação e Herança

Professor: Dr. Rafael Frederico Alexandre
Contato: rfalexandre@decea.ufop.br
Colaboradores: Renan SaldanhaContatos:
renansaldlinhares@gmail.com

Agenda

Agenda

Associação
Introdução

- Vínculo que ocorre entre classes;
- Ocorre quando uma classe tem um tipo de relacionamento "tem um"com outra classe;
- Define classes que se interagem entre elas num projeto;

Associação
Agregação

- 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;

Associação
Exemplo de agregação

```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`

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

Associação
Exemplo de composição

```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`

Associação
Exercício

1. Complemente o projeto do nosso comércio com os seguintes pontos:
   - O comércio além de possui gerente e vendedor deve possuir cliente;
   - O cliente deve ter um histórico de todos os produtos que ele comprou;
   - As compras dos cliente devem conter data e podem ser comprados mais de um produto na mesma compra;
   - Método que retorne todas as compras do cliente com data;
   - Os clientes devem ter nome, cpf, e telefone;

Dica: utilize a classe **ArrayList**;

Agenda

Herança
Introdução

Conceito [Devmedia, 2015]

A herança é um mecanismo da Orientação a Objeto que permite criar novas classes a partir de classes já existentes, aproveitando-se das características existentes na classe

- **Toda classe em Java é uma subclasse da classe Object**

Herança

- Classes diferente com características em comum (atributos, métodos);
- Derivação de uma classe em classes diferentes com propósitos próprios;
- Classes derivadas herdam atributos e métodos;
- Identificar uma herança por meio da palavra "é"

Herança

- A herança é difinida na classe pela palavra **extends**
- Classes em **Java** só pode herdar uma classe
- Para acessar atributos, métodos e contrutores da classe mãe deve ser utilizada a palavra **super**

Sintaxe herança

[modificador_de_acesso] class NomeDaClasse extends NomeDaClasseMãe{atibutos e métodos da classe}

Herança
Construtores

- O contrutor da classe filha deve ter super referênciando o construtor da classe mãe a ser utilizada
- O super dever estar na primeira linha dentro do bloco do construtor
- Para acessar atributos, metódos e contrutores da classe mãe deve ser utilizada a palavra **super**

Herança
Exemplo

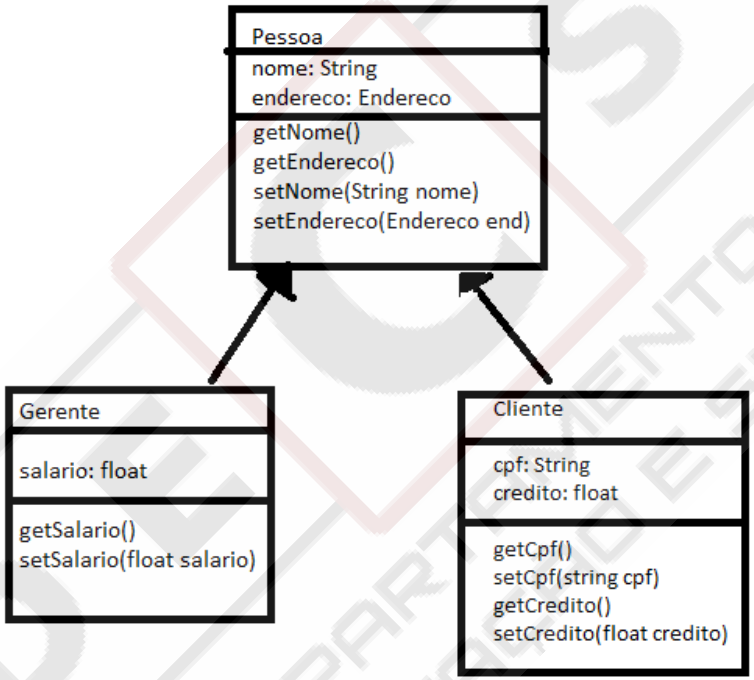


Figura: Diagrama de herança de classes.

Herança
Exemplo

```java
package br.ufop;

public class Cliente extends Pessoa {

    private float credito;

    public Cliente(String nome, int idade, String cpf, float credito) {
        super(nome, idade, cpf);
        this.credito = credito;
    }

    public float getCredito() {
        return credito;
    }

    public void setCredito(float credito) {
        this.credito = credito;
    }
}
```

Figura: Classe Cliente herdando classe Pessoa.

Herança
Super e sub classe

Definindo super e sub classe [Caelum, 2017]

Tomando como exemplo o exemplo anterior, podemos dizer que Pessoa é a superclasse de Cliente, e Cliente é a subclasse de Pessoa. Dizemos também que todo Cliente é uma Pessoa. Outra forma é dizer que Pessoa é classe mãe de Cliente e Cliente é classe filha de Pessoa.

Herança
Vantagens

- Reutilização de código
- Organização
- Maior produtividade

Herança
Exercício

1. Agora que você conhece herança, utilize esse conhecimento e melhore seu projeto referente ao comércio.

Agenda

Sobreescrita

- Utilizada para especializar os métodos herdados;
- Altera o comportamento do método na subclasse;
- Mesmo método na classe filha com comportamento diferente da classe mãe;
- O método deve ter o mesmo nome e mesma assinatura que o da classe mãe.

Sobreescrita

Exemplo

Em nosso projeto de comércio tanto vendedor quanto gerente recebem comissão, porém calculadas de maneiras diferente, logo podemos deixar o método que retorna o salario mais comissão de uma das classes na classe mãe e simplemste sobreescrever esse método na outra classe

Sobreescrita
Exemplo

```java
package br.ufop.carro;

public class Carro {
    private String modelo;
    private int ano;
    private String cor;

    public Carro(String modelo, int ano, String cor) {
        this.modelo = modelo;
        this.ano = ano;
        this.cor = cor;
    }

    @Override
    public String toString() {
        return "Carro: modelo=" + modelo + ", ano=" + ano + ", cor=" + cor;
    }


}
```

Figura: Sobreescrita do método toString

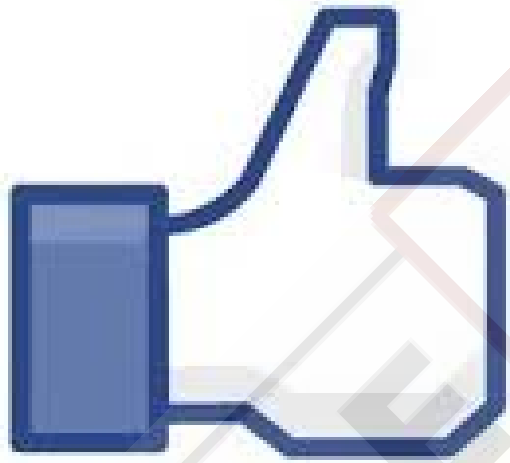
Sobreescrita
Boas práticas

@overrider [Caelum, 2017]

Há como deixar explícito no seu código que determinador método é a reescrito de um método da sua classe mãe. Fazemos isso colocando @Override em cima do método. Isso é chamado anotação. Vale resaltar que, por questões de compatibilidade, isso não é obrigatório. Mas caso um método esteja anotado com @Override, ele necessariamente precisa estar reescrevendo um método da classe mãe.

Agenda

- Respostas dos exercícos em : `https://github.com/renansald/TutoriaProgII`
- Dica de Pesquisa: Outros tipos de anotações utilizadas em **Java**



MUITO OBRIGADO!!!

Referncias Bibliogrficas I

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

Devmedia (2015).
Principais conceitos de programação orientada objetos.
devmedia.com.br/principais-conceitos-da-programacao-orientada-a-objetos/32285.