INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PIAUÍ
Campus Picos

Apresentação

Programação Orientada a Objetos

Pilares da OO

TEC INFO M3

Prof. João Paulo
joao.nascimento@ifpi.edu.br

Por favor...

- Manter o telefone celular sempre silencioso ou desligado em sala de aula;
- Se for receber uma ligação importante, atender fora da sala de aula;
- Fazer silêncio em sala de aula!

INSTITUTO FEDERAL PIAUÍ Campus Picos

Conteúdo Programático

- Encapsulamento
- Herança
- Polimorfismo
- Composição
- Bibliografia

INSTITUTO FEDERAL PIAUÍ Campus Picos

Encapsulamento

- *Encapsulamento* é uma técnica na POO que faz com que os objetos possuam seus detalhes internos ocultos para outros objetos;
- Esta técnica é bastante útil para diminuir o nível de acoplamento entre os componentes, separação das responsabilidades e proteção dos dados envolvidos.

INSTITUTO FEDERAL PIAUÍ Campus Picos

Encapsulamento

- Protegendo os atributos de uma classe com modificador de acesso *private* e permitindo acesso somente via *getters* e *setters* a partir de outras classes:

```
public class Pessoa {

        private String nome;
        private String sexo;

        public void setNome( String nome ) {
                /*Aqui você poderá implementar
                a melhor forma de configurar o nome,
                por exemplo, colocar Sr ou Sra antes
                após verificar o sexo. */
        }

        public String getNome() {
                //implementar o retorno do valor de nome
        }

}
```

Encapsulamento

- Resumo dos modificadores de acesso quanto à visibilidade:

	private	default	protected	public
mesma classe	sim	sim	sim	sim
mesmo pacote	não	sim	sim	sim
pacotes diferentes (subclasses)	não	não	sim	sim
pacotes diferentes (sem subclasses)	não	não	não	sim

Fonte: https://www.devmedia.com.br/metodos-atributos-e-classes-no-java/25404

Encapsulamento

- Links para consulta para saber mais sobre Encapsulamento:

  - ORIENTAÇÃO A OBJETOS em 8 minutos: Encapsulamento:
  https://www.youtube.com/watch?v=vkLEVgPGcyo

  - Revisitando a Orientação a Objetos: encapsulamento no Java:
  https://blog.caelum.com.br/revisitando-a-orientacao-a-objetos-encapsulamento-no-java/

  - O que é encapsulamento?:
  https://www.alura.com.br/artigos/o-que-e-encapsulamento

  - Curso POO Teoria #06a - Pilares da POO: Encapsulamento:
  https://www.youtube.com/watch?v=1wYRGFXpVlg

  - Curso POO Java #06b – Encapsulamento:
  https://www.youtube.com/watch?v=x4JfzV0Wb5w

INSTITUTO FEDERAL PIAUÍ Campus Picos

Herança

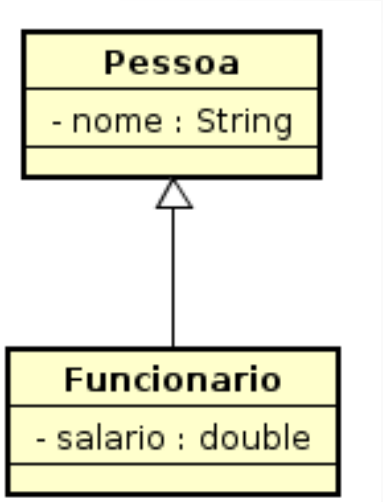
- Através deste mecanismo é possível uma classe, denominada de classe Mãe ou Superclasse, compartilhar para outras classes, estas denominadas de classe Filha ou Subclasse, seus atributos, métodos e outros membros;
- A *Herança* é útil para reaproveitamento de código na Orientação a Objetos;
- Uma desvantagem é que a classe filha (subclasse) fica bastante acoplada ao código de sua classe mãe (superclasse).

INSTITUTO FEDERAL PIAUÍ Campus Picos

Herança

- Uma classe *Funcionario* herdando da classe *Pessoa*. Ela terá acesso a todos os membros públicos ou protegidos (*protected*) de sua superclasse:

```
public class Pessoa {

        private String nome;

        public void setNome( String nome ) {
                this.nome = nome;
        }

        public String getNome() {
                return nome;
        }

}
```




```
//numa classe teste...
Funcionario func = new Funcionario();
func.setNome(); //acessando método da classe Pessoa
```

```
public class Funcionario extends Pessoa {

        private double salario;
}
```

Herança

- Alguns links para estudar mais sobre Herança:

  - Curso POO Teoria #10a - Herança (Parte 1):
  https://www.youtube.com/watch?v=_PZldwo0vVo

  - Curso POO Java #10b - Herança (Parte 1):
  https://www.youtube.com/watch?v=19IGAeoFKlU

  - Curso POO Teoria #11a - Herança (Parte 2):
  https://www.youtube.com/watch?v=He887D2WGVw

  - Curso POO Java #11b - Herança (Parte 2):
  https://www.youtube.com/watch?v=5pwV2WdD-_Y

  - Entendendo e Aplicando Herança em Java:
  https://www.devmedia.com.br/entendendo-e-aplicando-heranca-em-java/24544

  - Ler o Capítulo 9 (Herança), Java: como Programar, 8ª edição (DEITEL)

INSTITUTO FEDERAL PIAUÍ Campus Picos

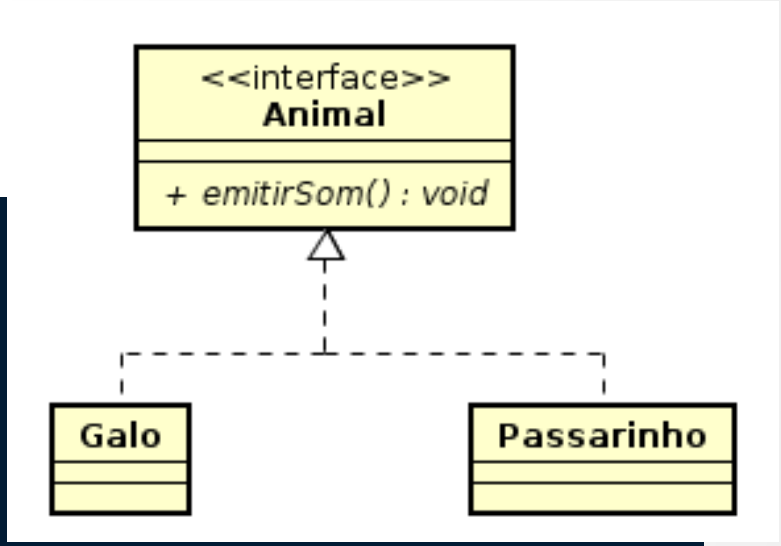
Polimorfismo

- Com este princípio objetos que derivam de uma mesma entidade (classe ou interface) podem fazer chamada, em tempo de execução, a métodos em comum (de mesma assinatura), mas com comportamentos distintos (sobreposição);
- Ou também, um método de mesmo nome pode ser sobrecarregado e com isso produzir diferentes resultados com dados diversos (sobrecarga);
- Por exemplo, objetos *Galo* e *Passarinho*, ambos possuem relacionamento com *Animal* do tipo “é um”, ou seja, via herança ou realização. Todo *Animal* pode emitir um som, mas tanto o *Galo* e tanto o *Passarinho*, o fazem à sua maneira;
- Em outras palavras, *Polimorfismo* é um objeto se ‘transformar’ em outro tipo de objeto, em tempo de execução.

INSTITUTO FEDERAL PIAUÍ Campus Picos

Polimorfismo

- Em tempo de execução, é decidido qual animal irá realizar o som:




```
public interface Animal {
        public void emitirSom();
}

public class Galo implements Animal {
        public void emitirSom() {
                //implementação de fazer som do Galo...
        }
}

public class Passarinho implements Animal {
        public void emitirSom() {
                //implementação de fazer som do Passarinho...
        }
}
```

INSTITUTO FEDERAL PIAUÍ Campus Picos

Polimorfismo

- Em tempo de execução, é decidido qual animal irá realizar o som (continuação):

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

                Animal animal = null;
                String escolha = "passarinho";

                if( escolha.equals("galo") )
                        animal = new Galo();
                else
                        animal = new Passarinho();

                /*o som reproduzido será de acordo
                com a escolha do animal no if...*/
                animal.emitirSom();
        }
}
```

INSTITUTO FEDERAL PIAUÍ Campus Picos

Polimorfismo

- Links para estudar mais acerca de Polimorfismo:

  - Curso POO Teoria #12a - Conceito Polimorfismo (Parte 1):
  https://www.youtube.com/watch?v=9-3-RMEMcq4

  - Curso POO Java #12b - Polimorfismo em Java (Parte 1):
  https://www.youtube.com/watch?v=NctjqlfKC0U

  - Curso POO Teoria #13a - Conceito Polimorfismo (Parte 2):
  https://www.youtube.com/watch?v=hYek1xqWzgs

  - Curso POO Java #13b - Polimorfismo Sobrecarga (Parte 2):
  https://www.youtube.com/watch?v=b7xGYh3NHZU

  - Uso de Polimorfismo em Java:
  https://www.devmedia.com.br/uso-de-polimorfismo-em-java/26140

  - Ler o Capítulo 10 (Polimorfismo), Java: como Programar, 8ª edição (DEITEL)

INSTITUTO FEDERAL PIAUÍ Campus Picos

Composição

- Um objeto pode ser composto de outros objetos, para assim cumprir suas tarefas;
- Associação: um objeto possui um ou mais objetos em sua implementação, mas não se tratando de um relacionamento “todo-parte”;
- Agregação: um objeto possui um ou mais objetos em sua implementação, constituindo uma relação todo-parte, porém, as suas partes podem existir sem o objeto que representa o todo;
- Composição: um objeto possui um ou mais objetos em sua implementação, constituindo uma relação todo-parte, sendo que se o mesmo (o objeto que representa o todo) deixar de existir, suas partes também deixam de existir.

INSTITUTO FEDERAL PIAUÍ Campus Picos

Composição

- Exemplos em Java (Composição):

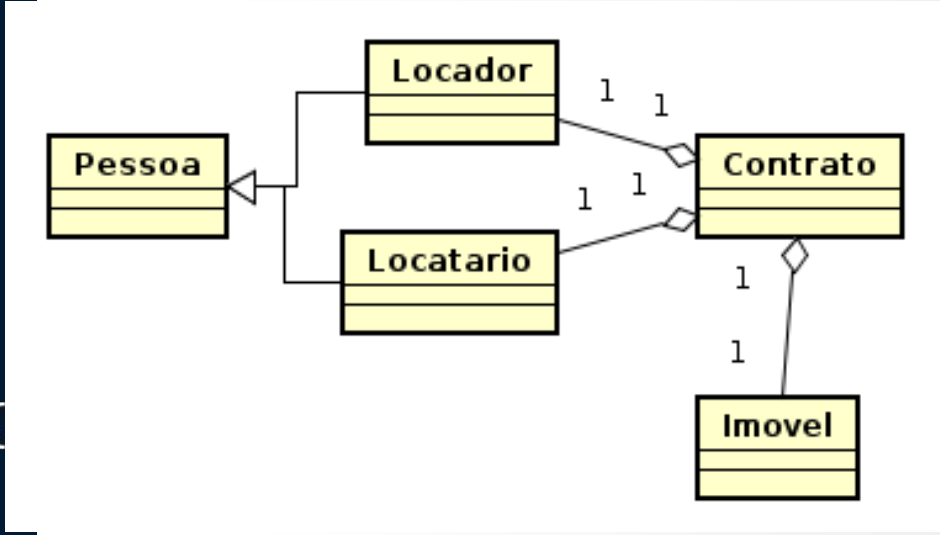
```
public class Carro {
        private Motor motor;
        private Roda[] rodas;
}

public class Motor {

}

public class Roda {

}
```

Nota: neste exemplo assumimos que Motor e as Rodas não podem existir sem o Carro.

Composição

- Exemplos em Java (Agregação):

```
public class Contrato {

        private Locatario locatario;
        private Locador locador;
        private Imovel imovel;
}

public class Locatario extends Pessoa {

}

public class Locador extends Pessoa {

}

public class Imovel {

}
```

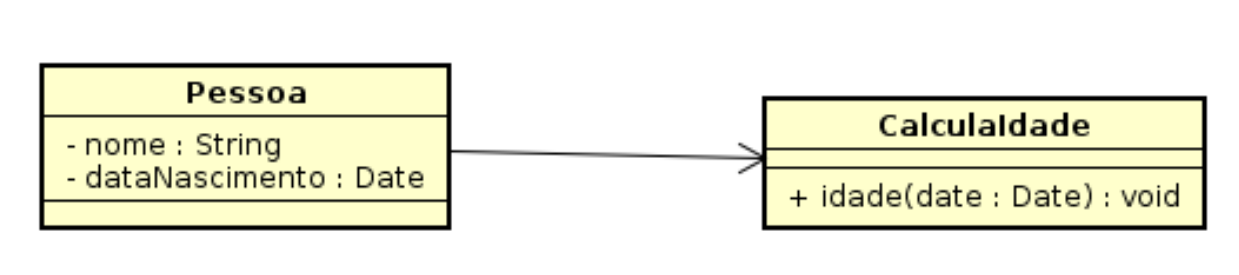


Nota: neste exemplo assumimos que Locatario, Locador e Imovel podem existir mesmo sem o Contrato.

Composição

- Exemplos em Java (Associação):

```
public class Pessoa {

        private String nome;
        private Date dataNascimento;

        public int getIdade() {

                return new CalculaIdade().idade(dataNascimento);
        }
}
```



Nota: neste exemplo assumimos que Pessoa utiliza o serviço da classe CalculaIdade, sem manter relação "todo-parte".

Composição

- Links para estudo acerca de composição na POO:

  - Curso POO Teoria #07a - Relacionamento entre Classes: https://www.youtube.com/watch?v=GLHbxDU9iBA

  - Curso POO Java #07b - Objetos Compostos em Java: https://www.youtube.com/watch?v=BfrbCQ3XcrA

  - Curso POO teoria #08a - Relacionamento de Agregação: https://www.youtube.com/watch?v=ERdvijGtrq0

  - Curso POO Java #08b - Agregação entre Objetos com Java: https://www.youtube.com/watch?v=8R9RpqpXI_c

  - Dependência entre Classes: https://www.ateomomento.com.br/orientacao-a-objetosuml-relacionamento-entre-classes-dependencia/

  - Relacionamento entre Classes – Composição: https://www.ateomomento.com.br/orientacao-a-objetosuml-relacionamento-entre-classes-composicao/

  - UML – Relacionamento entre Classes – Agregação: https://www.ateomomento.com.br/uml-classes-agregacao/

INSTITUTO FEDERAL PIAUÍ Campus Picos

Bibliografia

- DEITEL, Harvey; DEITEL, Paul. Java Como Programar. 8a Ed. Bookman, 2010.
- SANTOS, Rafael. Introdução à programação orientada a objetos usando Java. Rio de Janeiro: Campus, 2003. 319, [4] p. (Editora Campus ; Sociedade Brasileira de Computação). ISBN 85-352-1206-X (broch.).
- SINTES, Anthony. Aprenda Programação Orientada a Objetos. São Paulo: Pearson, 2002. 693p. (Editora Pearson ; Sociedade brasileira de computação). ISBN 85-34-1461-0.
- SILVA FILHO, Antonio Mendes da. Introdução à programação orientada a objetos com C++. Rio de Janeiro: Elsevier, 2010. 283 p. ISBN 978-85-352-3702-3
- VILARIM, Gilvan de Oliveira. Programação orientada a objetos: um curso básico. Curitiba: Editora do Livro Técnico, 2015. 168p. ISBN 9788563687920.
- CARDOSO, Caíque. Orientação a objetos na prática: aprendendo orientação a objetos com Java . Rio de Janeiro: Ciência Moderna, 2006. 175 p. ISBN 85-7393-538-3.