Programação Orientada a Objetos (POO)

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

Herança

- Mesmo conceito do mundo real;
- Classes podem herdar *atributos* e *métodos* de outra classe;
- Em Java, não se pode herdar de duas classes ao mesmo tempo (herança múltipla). Em C++ isso é possível;

Herança

- Na fase de projeto: perguntar **É-UM**;
- Atributos e métodos configurados como "private" **não** são herdáveis;
- Proporciona o **reuso** do código, otimizando:
  - tempo
  - linhas de código
  - entendimento do projeto
  - manutenção do projeto

Herança

Quais os relacionamentos abaixo fazem sentido?

- Forno herda Cozinha;
- Guitarra herda InstrumentoMusical;
- Gavião herda Mamífero;
- Bebida herda Suco;
- Camisa herda Roupa;
- Região herda Norte;
- Imóvel herda Apartamento.

Herança

Exemplo de programação de herança em Java (1/6):

```
package br.unama.projetouniversidade;
public class Pessoa {
        String nome;
        private String cpf;
        void realizarCadastro( ) {
                //rotina para realizar cadastro...
        }
}
```

Herança

Exemplo de programação de herança em Java (2/6):

```
package br.unama.projetouniversidade;
public class Aluno {
        int matricula;
        void realizarMatricula( ){
                //rotina para realizar matrícula...
        }
}
```

Herança

Exemplo de programação de herança em Java (3/6):

```
package br.unama.projetouniversidade;
public class Aluno extends Pessoa {
        int matricula;
        void realizarMatricula( ){
                //rotina para realizar matrícula...
        }
}
```

Herança

Exemplo de programação de herança em Java (4/6):

```
package br.unama.projetouniversidade;
public class ProjetoUniversidade {
    public static void main( String args[ ] ) {
            Aluno aluno1 = new Aluno( );
            aluno1.matricula = 260107700; funciona? SIM, é um atributo da classe Aluno.
            aluno1.nome = "João Silva";     funciona? SIM, graças a herança.
            aluno1.cpf = "037.802.142-99"; funciona? NÃO, 'cpf' é privado em Pessoa.
    }
}
```

Herança

Exemplo de programação de herança em Java (5/6):

```
package br.unama.projetouniversidade;
public class Professor extends Pessoa {
        boolean hasMestrado;
        void lancarNotas( ){
                //rotina para lançar notas...
        }
}
```

Herança

Exemplo de programação de herança em Java (6/6):

```
package br.unama.projetouniversidade;
public class ProjetoUniversidade {
    public static void main( String args[ ] ) {
            Professor prof1 = new Professor( );
            prof1.hasMestrado = true;   funciona? SIM, é um atributo da classe Professor.
            prof1.nome = "Paulo Silva"; funciona? SIM, graças a herança.
            prof1.matricula = 2014009;  funciona? NÃO, 'matricula' faz parte da Aluno.
            prof1.cpf = "037.802.142-99"; funciona? NÃO, 'cpf' é privado em Pessoa.
    }
}
```

Herança

```
public class Pessoa {
    String nome;
    private String cpf;
}

public class Aluno extends Pessoa {
    int matricula;
}

public class Professor extends Pessoa {
    boolean hasMestrado;
}
```

```
public class ProjetoUniversidade {
    public static void main( String args[ ] )
    {
        Aluno aluno2 = new Aluno( );
        Professor prof2 = new Professor( );
        Pessoa pessoa1 = new Pessoa( );
    }
}
```

Faz sentido a criação deste objeto para a universidade? Resp: Não...

Herança

Objeto da classe Aluno

Objeto da classe Professor

Objeto da classe Pessoa
(não tem sentido para o negócio)

Herança - classe abstrata

- Em alguns casos, a classe-mãe (super classe) não deve ser instanciada;
- O objeto pode não ter significado no mundo real. Lembre-se: o sistema deve refletir o mundo real (o negócio);
- Para não permitir a instanciação de uma classe, utiliza-se **abstract**.

Herança - classe abstrata

Exemplo:

```
public abstract class Pessoa {


}
```

Herança – classe abstrata

```
public abstract class Pessoa {
    String nome;
    private String cpf;
}

public class Aluno extends Pessoa {
    int matricula;
}

public class Professor extends Pessoa {
    boolean hasMestrado;
}
```

```
public class ProjetoUniversidade {
    public static void main( String args[ ] )
    {
        Aluno aluno2 = new Aluno( );
        Professor prof2 = new Professor( );
        Pessoa pessoa1 = new Pessoa( );
    }
}
```

funciona?

NÃO, pois a classe Pessoa é abstrata.

Modificador de acesso protected

- O membro só pode ser acessado:
  - pelo próprio pacote (igual o **default**) e
  - por subclasses, mesmo estando em outros pacotes.
- Pode ser usado em atributos, métodos e construtores de uma classe;
- Não pode ser usado por classes.

Herança – classe abstrata

```
public abstract class Pessoa {
    protected String nome;
    private String cpf;
    String rg;
}
public class Aluno extends Pessoa {
    int matricula;
}
public class Professor extends Pessoa {
    boolean hasMestrado;
}
```

```
public class ProjetoUniversidade {
    public static void main( String args[ ] ) {
        Aluno aluno2 = new Aluno( );
        aluno2.nome = "José Santos";
        aluno2.cpf = "392.987.022-10";
        aluno2.rg = "4449936";
    }
}
```

funcionam?

SIM para atributo 'nome', pois ele é *protected* / NÃO para 'cpf', pois ele é *private* / SIM para 'rg', pois ele é *default* e as classes fazem parte do mesmo pacote.

Exercício 1

- Acesse **https://github.com/amarcel/Unama2018p1**
- Baixe os projetos-exercícios **Exerc04_Banco, Exerc05_Banco, Exerc06_Banco,** da pasta **POO -EXERCICIOS - TODO**
- Abra os projetos no Netbeans;
- Faça os TODOs de cada projeto, começando pelo **Exerc04_Banco**;