

Introdução da UC - Revisão de POO

Sistemas Distribuídos e Mobile

Prof. Me. Gustavo Torres Custódio
gustavo.custodio@anhembi.br

Bem-vindos!

- A UC será ministrada duas vezes por semana por diferentes professores.
- A cada semana, teremos uma aula remota e uma aula presencial.

Professores

- Aula remota (quinta-feira):
 - Prof. Fernando Kakugawa
- Aula presencial (sexta-feira):
 - Gustavo Torres Custódio

Ementa

- Introdução aos Sistemas Distribuídos;
- Arquitetura Cliente-Servidor e Peer-to-Peer;
- Sockets;
- Padrões de transferência de dados na Internet;
- Web Service - REST;
- Microsserviços;
- Modelos de Falhas e Segurança;
- Transações Distribuídas e Controle de Concorrência;
- Cloud Computing;
- Padrões de Projetos (Design Patterns);
- Conceitos básicos de IoT.

Data das Avaliações

- A1:
 - **10 a 15/10** - avaliação online
- A2:
 - **05 a 06/12** - avaliação online

Data das Avaliações

- A3 :
 - **23/10** - Treinamento AWS completo
 - **03/11 e 10/11** - Apresentação Seminário da AWS (20%)
 - **22/11** - Entrega Projeto (50%)
 - **24/11 e 01/12** - Apresentação do projeto (20%)

Avaliações

- 100 pontos no total;
- A1:
 - Dissertativa;
 - 30 pontos.
- A2:
 - Múltipla escolha (nível nacional);
 - 30 pontos.
- A3:
 - Seminário / Projeto;
 - 40 pontos.

Aprovação e AI

- Para o aluno ser aprovado são necessários:
 - Nota mínima: 70;
 - Frequência: 75%.
- **AI:**
 - Caso o aluno não atinja a nota mínima, ele pode realizar a **AI**.
 - A AI será realizada no próximo semestre.
 - Ela substitui a menor nota entre a A1 e A2.

Academy Cloud Foundation

- Treinamento *Academy Cloud Foundation*
- Inscrição AWS:
 - <https://forms.office.com/r/J7NYxdY9uV>
- Prazo: 27/08






Academy Cloud Foundation

- Treinamento *Academy Cloud Foundation*

- Visão geral dos conceitos de nuvem
- Economia e faturamento da nuvem
- Visão geral da infraestrutura global da AWS
- Segurança na Nuvem AWS
- Redes e entrega de conteúdo
- Computação
- Armazenamento
- Bancos de dados
- Arquitetura de nuvem
- Auto Scaling e monitoramento



Bibliografia

-  George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair.
Sistemas Distribuídos: Conceitos e Projeto.
Bookman Editora, 5 edition, 2013.
-  Harvey M Deitel, Paul J Deitel, David R Choffnes, et al.
Sistemas Operacionais.
Pearson/Prentice Hall, 3 edition, 2005.
-  Maarten Van Steen and A Tanenbaum.
Sistemas Distribuídos: Princípios e Paradigmas.
Pearson/Prentice Hall, 2 edition, 2007.

Bibliografia Complementar

 Harvey M Deitel and Paul J Deitel.
Java, como programar. Ed.
Pearson/Prentice Hall, 8 edition, 2010.



Introdução da UC - Revisão de
POO

Programação Orientada a Objetos (POO)

Agenda

Programação Orientada a Objetos (POO)

Classes

Herança

Classes Abstratas

Polimorfismo

Introdução

- Abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes.
- Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.
- As entidades em Programação Orientada a Objetos (POO) devem representar entidades no mundo real.

Tipo Abstrato de Dados - TAD

- Um Tipo Abstrato de Dados (TAD) refere-se ao conceito de definição de um tipo de dado. A definição de TAD leva à criação de um novo tipo de dado.
- Exemplo:
 - Pode-se criar um tipo Racional, onde os valores armazenados têm a forma $\frac{1}{2}$, $\frac{2}{3}$, etc.
 - E sobre esse conjunto podem ser especificadas operações, +, -, *, /.

Tipo Abstrato de Dados - TAD

- Um determinado objeto deve representar uma entidade do mundo real em um sistema computacional.
 - Deve ser descrito através de suas características desejáveis (atributos) e as operações que são realizadas nele (métodos).



Introducao da UC - Revisao de
POO

Classes

Classes

- Uma classe é uma declaração de tipo que agrega:
 - contante, variáveis e funções.
- Classes possuem basicamente dois grupos de elementos:
 - atributos;
 - comportamentos.

Classes

- Atributos
 - definem as características que cada objeto de uma classe deve ter.
- Comportamentos
 - definem as ações que cada objeto de uma classe pode executar.

Classes em Java

Definição de classe

Atributos / variáveis

```
public class Pessoa {  
  
    private String nome;  
    private int idade;  
  
    public Pessoa(String nome, int idade) {  
  
        this.nome = nome;  
        this.idade = idade;  
    }  
  
    // ----- Métodos -----  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public int getIdade() {  
        return idade;  
    }  
  
    public void setIdade(int idade) {  
        this.idade = idade;  
    }  
  
    public void andar() {  
        System.out.println("A pessoa está andando.");  
    }  
}
```

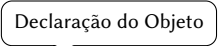
Construtor

Classes em Java

- `private` / `public`
 - encapsulamento;
 - define para quais classes o atributo é visível;
 - `public`: visível para todas as classes;
 - `private`: visível apenas para a própria classe.
- Atributos `get` / `set`
 - utilizados para acessar as variáveis **`private`**;
 - **`set`** muda o valor da variável;
 - **`get`** acessa o valor da variável.

Classes em Java

```
public class Main {  
    public static void main (String[] args) {  
        Pessoa pedro = new Pessoa("Pedro", 24);  
        System.out.println(pedro.getNome() + " " + pedro.getIdade());  
  
        pedro.andar()  
        pedro.setIdade(25);  
  
        System.out.println(pedro.getNome() + " " + pedro.getIdade());  
    }  
}
```



Classes em Java

- Saída do Programa:

Pedro 24

A pessoa está andando.

Pedro 25

Exercícios

- **Exercício 1** - Faça um programa em Java para termos um tipo abstrato de dado chamado Produto. No produto poderemos armazenar o nome, marca, preço e peso.
 - Crie pelo menos 3 objetos diferentes para o produto e em seguida altere os seus valores.



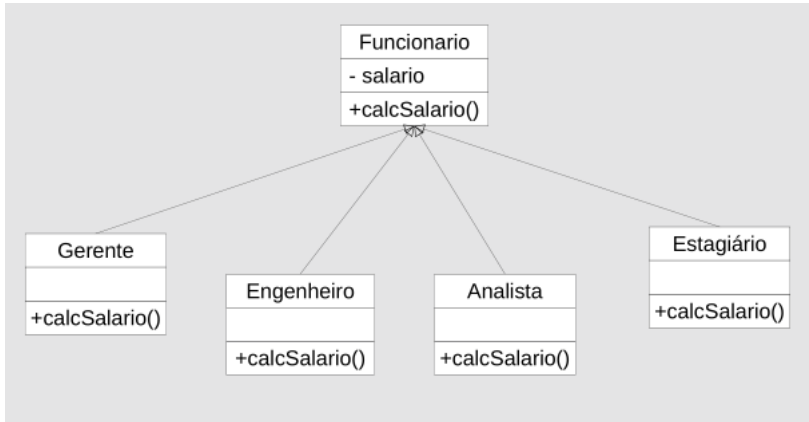
Introdução da UC - Revisão de
POO

Herança

Herança

- Permite que características comuns a diversas classes sejam “herdadas” de uma classe base, ou superclasse.
- É uma forma de reutilização de software
 - Novas classes são criadas a partir das classes existentes, absorvendo seus atributos e comportamentos e adicionando novos recursos.

Herança



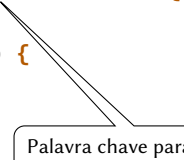
Herança em Java

Encapsulamento

```
public class Funcionario {  
  
    protected String nome;  
    protected double salario;  
  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
    public double getSalario() {  
        return salario;  
    }  
    public void setSalario(double salario) {  
        this.salario = salario;  
    }  
    public void printNome() {  
        System.out.println("Nome: " + this.nome);  
    }  
    public void printSalario() {  
        System.out.println("Salário: " + this.salario);  
    }  
    public void calcSalario() {  
        this.salario = 1000;  
    }  
}
```

Herança em Java

```
public class Gerente extends Funcionario {  
  
    public void calcSalario() {  
        this.salario = 20000;  
    }  
}
```

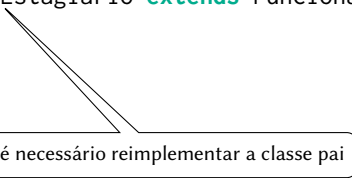


A callout box with the text "Palavra chave para herança" (Key word for inheritance) has two lines pointing to the `extends` keyword in the first code block and the `Funcionario` class name in the second code block.

```
public class Engenheiro extends Funcionario {  
  
    public void calcSalario() {  
        this.salario = 10000;  
    }  
}
```

Herança em Java

```
public class Analista extends Funcionario {  
  
    public void calcSalario() {  
        this.salario = 5000;  
    }  
}  
  
public class Estagiario extends Funcionario {  
  
}
```



Não é necessário reimplementar a classe pai

Herança em Java

```
public static void main(String[] args) {  
    Gerente g = new Gerente();  
    g.setNome("Pedro")  
    g.printNome();  
    g.calcSalario();  
    g.printSalario();  
  
    Engenheiro e = new Engenheiro();  
    e.setNome("Patricia")  
    e.printNome();  
    e.calcSalario();  
    e.printSalario();  
  
    Analista a = new Analista();  
    a.setNome("José")  
    a.printNome();  
    a.calcSalario();  
    a.printSalario();  
  
    Estagiario estag = new Estagiario();  
    estag.setNome("Julia")  
    estag.printNome();  
    estag.calcSalario();  
    estag.printSalario();  
}
```


Herança em Java

- Saída do programa:

Nome: Pedro

Salario: 20000.0

Nome: Patricia

Salario: 10000.0

Nome: José

Salario: 5000.0

Nome: Julia

Salario: 1000.0

Exercícios

- **Exercício 2** - Faça um programa em Java para uma concessionária, onde ela vende carros, motos e caminhões.



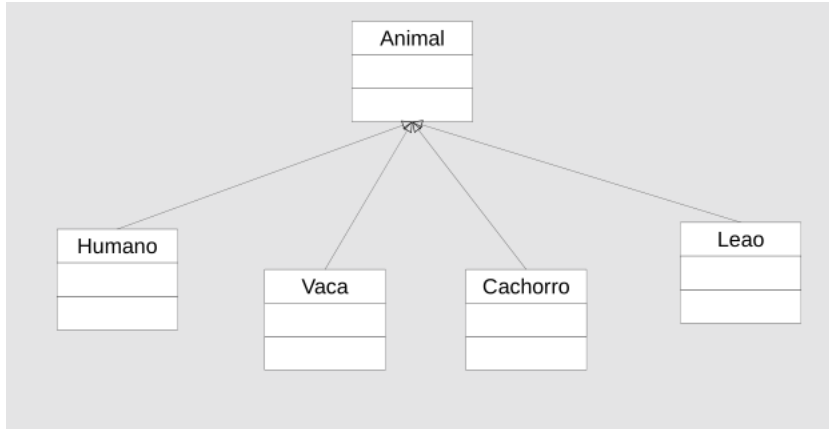
Introdução da UC - Revisão de
POO

Classes Abstratas

Abstração

- Essa característica permite que grandes sistemas sejam especificados em um nível muito geral, muito antes de ocorrer a implementação dos métodos individuais.
- Classes que não podem ser instanciadas!
- Permite definir métodos que sem implementação - que devem ser redefinidos nas subclasses.

Abstração



Abstração em Java

```
public abstract class Animal {  
    public abstract void falar()  
}
```

Define a classe abstrata

Define o método abstrato

```
public class Humano extends Animal {  
    public void falar() {  
        System.out.println("Eu posso falar - bla bla bla.");  
    }  
}
```

```
public class Vaca extends Animal {  
    public void falar() {  
        System.out.println("Eu posso mugir - muuuuuu.");  
    }  
}
```

Abstração em Java

```
public class Cachorro extends Animal {  
    public void falar() {  
        System.out.println("Eu posso latir - au au au.");  
    }  
}  
  
public class Leao extends Animal {  
    public void falar() {  
        System.out.println("Eu posso rugir - roooooaaaarr.");  
    }  
}
```

Abstração em Java

```
public class Main {  
  
    public static void main (String[] args) {  
        Humano h = new Humano();  
        h.falar();  
  
        Vaca v = new Vaca();  
        v.falar();  
  
        Cachorro c = new Cachorro();  
        c.falar();  
  
        Leao l = new Leao();  
        l.falar();  
    }  
}
```


Abstração em Java

- Será mostrado na tela:

Eu posso falar - bla bla bla.

Eu posso mugir - muuuuuu.

Eu posso latir - au au au.

Eu posso rugir - roooooaaaarr.

Exercícios

- **Exercício 3** - Faça um programa em Java para um aplicativo de desenho, onde temos uma Forma abstrata (cor e método para calcular a área).
 - Deve-se ter o círculo, quadrado, retângulo e triângulo.



Introdução da UC - Revisão de
POO

Polimorfismo

Polimorfismo

- A palavra polimorfismo significa ter muitas formas.
 - Em palavras simples, podemos definir polimorfismo como a capacidade de uma mensagem ser exibida em mais de uma forma.
- O polimorfismo é considerado uma das características importantes da programação orientada a objetos.

Polimorfismo

- Usando uma definição mais formal:
 - Polimorfismo é quando duas ou mais classes herdam da mesma classe mãe.
 - Ambas invocam métodos com nomes idênticos.
 - Porém com comportamentos diferentes.

Polimorfismo em Java

```
public class Ave {  
    public void introduzir() {  
        System.out.println("Existem muitas aves.");  
    }  
  
    public void voar() {  
        System.out.println("A maioria das aves podem voar, mas algumas não.");  
    }  
}
```

```
public class Cegonha extends Ave {  
    public void voar() {  
        System.out.println("Cegonha pode voar.");  
    }  
}
```

Polimorfismo em Java

```
public class Avestruz extends Ave {  
    public void voar() {  
        System.out.println("A maioria das aves podem voar, mas algumas não.");  
    }  
}
```

```
public class Andorinha extends Ave {  
    public void introduzir() {  
        System.out.println("Existem muitas aves e a Andorinha é uma delas.");  
    }  
  
    public void voar() {  
        System.out.println("Andorinha pode voar.");  
    }  
}
```

Polimorfismo em Java

```
public static void main (String[] args) {  
    Ave a = new Ave();  
    a.introduzir();  
    a.voar();  
  
    Cegonha c = new Cegonha();  
    c.introduzir();  
    c.voar();  
  
    Avestruz az = new Avestruz();  
    az.introduzir();  
    az.voar();  
  
    Andorinha and = new Andorinha();  
    and.introduzir();  
    and.voar();  
}
```


Polimorfismo em Java

- Será mostrado na tela:

Existem muitas aves.

A maioria das aves podem voar, mas algumas não.

Cegonha pode voar.

Existem muitas aves.

Avestruz não pode voar.

Andorinha pode voar.

Exercícios

- **Exercício 4** - Usando polimorfismo, faça um programa em java para venda de imóveis:
 - Crie a classe `Imovel`, que possui um endereço e um preço.
 - crie uma classe `NovoImovel`, que herda de `Imovel` e possui um adicional no preço. Crie métodos de acesso e impressão deste valor adicional.
 - crie uma classe `VelhoImovel`, que herda de `Imovel` e possui um desconto no preço. Crie métodos de acesso e impressão para este desconto.

Exercícios

- **Exercício 5** - Crie as seguintes classes em Java:

- Funcionário

- Atributos:

- `public` String nome;
 - `public` String cargo;
 - `public double` salario;

- Métodos:

- `public` String toString();
 - O toString() retorna informações sobre os atributos.

- Gerente

- Atributos:

- `public` String nome;

- Métodos:

- `public void` atualizar(Funcionario f, String cargo);
 - `public void` atualizar(Funcionario f, `double` salario);

Exercícios

- **Exercício 6**
- Construir em Java uma classe para representar um funcionário com:
 - nome, horas trabalhadas e valor pago por hora trabalhada;
 - implementar métodos para:
 - calcular e retornar o salário final de um funcionário;
 - mostrar as informações do funcionário.
- Criar uma subclasse para representar um funcionário senior.
 - a diferença entre eles é que um funcionário sênior recebe um bônus a cada 10 horas trabalhadas.
 - sobrescrever os métodos `calcularSalario` e `imprimir`.

Conteúdo



<https://gustavotcustodio.github.io/sdmobile.html>

Obrigado

gustavo.custodio@anhembi.br