

Sejam bem-vindos!

UC - Sistemas Distribuídos e Mobile



Universidade
Anhembi Morumbi



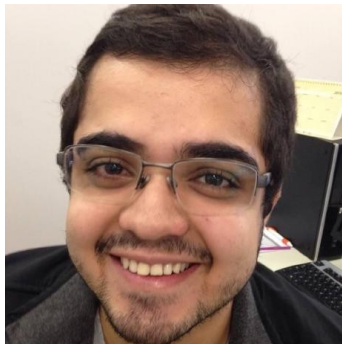
Prof. Gustavo Torres Custodio

Mestre em Ciência da Computação

Docente: Ambiente Presencial



gustavo.custodio@ulife.com.br



Universidade
Anhembi Morumbi

Sistemas Distribuídos e mobile

Ementa

- Programação orientada à Objetos (Revisao POO)
- Implementação Sockets TCP/UDP
- Sockets em ambiente Paralelo (Multithreaded)
- Padrão de Projeto Proxy
- Remote Procedure Call (RPC)
- Web Services SOAP
- Web Services REST

- Ambiente de progração JAVA



Metas de Compreensão

- Identificar a arquitetura web, seus principais protocolos e a comunicação entre computadores na Internet;
- Reconhecer novas tecnologias potenciais para implementações em sistemas paralelos e distribuídos;
- Planejar e implementar medidas de segurança em sistemas distribuídos e/ou nos serviços gerados;
- Aplicar e integrar soluções IoT no desenvolvimento de sistemas distribuídos e mobile;



Metas de Compreensão

- Aplicar e integrar conceitos de computação, armazenamento e processamento em nuvem em soluções de sistemas distribuídos;
- Selecionar recursos para implementação de sistemas paralelos e distribuídos;
- Implementar soluções de processamento paralelo e/ou distribuído em aplicações que exigem velocidade e escalabilidade;

Meta máxima:

- Projetar e desenvolver sistemas com arquiteturas baseadas em serviços e micro serviços seguindo padrões de projetos.



NOSSO ECOSSISTEMA DE APRENDIZAGEM

UM INOVADOR MODELO DE ENSINAR E APRENDER



UCs Core Curriculum

Escolha as competências de seu interesse, dentre artes, mindfulness, raciocínio lógico, línguas, e outros. O objetivo é que ele tenha uma visão global da realidade.

UCs da Área e da Profissão

Resolva problemas em equipes multiprofissionais, da mesma forma que acontece no mercado de trabalho. O estudante tem contato com colegas de outros cursos, não apenas da graduação escolhida.

UCs Específicas

Nesse eixo o estudante interage com alunos do próprio curso, aprendendo e resolvendo problemas ligados à profissão de sua escolha.

UCs Duais

O aluno poderá cursar UCs dentro de empresas. Desenvolvendo projetos reais dentro de companhias e indústrias.



**Universidade
Anhembi Morumbi**

MATRIZ CURRICULAR



TEPO MÍNIMO
DE INTEGRALIZAÇÃO:
10 SEMESTRES

→ CURRÍCULO INTEGRADO POR COMPETÊNCIAS, PERSONALIZADO, CONECTADO ÀS DEMANDAS DO MUNDO DO TRABALHO

Por meio de quatro diferentes comunidades de aprendizagem, você pode construir o seu projeto de vida desde o início do curso, aprender na prática, trocar conhecimento com outras áreas, ampliar suas redes e viver uma experiência universitária plena.

COMUNIDADES DE APRENDIZAGEM

Unidades curriculares
organizadas por competências

Core Curriculum

Adequado aos melhores cursos do mundo, o Core Curriculum integra os conhecimentos necessários às estruturas das sete disciplinas essenciais, independentemente da carreira escolhida. A comunidade de aprendizagem, com alunos de diferentes cursos, permite o diálogo e o desenvolvimento da visão de mundo e o exercício da cidadania.

Área

Esta comunidade aborda assuntos relacionados à grande área de conhecimento. Você se desenvolverá competências comuns à formação, a partir de atividades presenciais e virtuais de ensino remoto.

Profissional

As Unidades Curriculares desta comunidade inserem os estudantes no mundo do trabalho. Você se especializa em áreas de atuação, desenvolvendo competências em áreas multidisciplinares, de maneira formativa que atenda ao mercado de trabalho.

Específico

Nas comunidades de aprendizagem específicas, o foco está no desenvolvimento de competências de sua futura carreira junto aos outros estudantes da sua turma.

DIVERSIDADE DE AMBIENTES

Preparação para prosperar sua
experiência de sala de aula

Presencial

Atividades presenciais integradas, acompanhamento acadêmico por tutores e mentores, laboratórios e ambientes de aprendizagem.

Ambientes Virtuais

Sala de aula digital, laboratórios de simulação, gamificação, plataforma digital com desafios reais de projetos de aprendizagem e atividades digitais.

Mundo do Trabalho

Projeto Vida & Carreira, com desenvolvimento de competências socioemocionais em uma plataforma de escuta pelo estudante. A cada Unidade Curricular, você realizará uma certificação e as competências adquiridas serão inseridas em seu currículo do futuro. Você também poderá atuar como um profissional em uma empresa.

Comunidade

Projeto de extensão que estimula seu protagonismo na transformação da realidade do entorno do campus, com ações locais de impacto social e de responsabilidade social e a aplicação de conhecimentos desenvolvidos na turma.

VIDA & CARRERA

Componente curricular que faz a conexão do seu Projeto de Vida com o Mundo do Trabalho. Com o apoio de tutores e mentores, você terá acesso a uma plataforma que auxilia na adequação de sua carreira durante toda a vida (Life Long Learning).



Universidade
Anhembi Morumbi



Universidade
Anhembi Morumbi

Vale Saber que...

As Unidades Curriculares (UCs) fazem parte do cronograma de atividades dos alunos e são completadas com outras ações que também precisam ser desenvolvidas:

Atividades Complementares,
Projetos de Extensão (10% obrigatórios - Resolução n. 7 de 2018 do MEC),
Projeto Vida & Carreira,
TCC e Estágio, quando previstos.

AVALIAÇÃO UNIDADE CURRICULAR

Avaliação
dissertativa - saber
se expressar de
forma escrita, de
acordo com a área.



A1
ESCRITA

30%



A2
LEITURA E
INTERPRETAÇÃO

30%



A3
DESEMPENHO

40%

Substitui A1 ou A2
(a menor nota) e,
após, somam-se
as novas notas.



AI
AVALIAÇÃO
INTEGRADA

30%

Avaliação de múltipla
escolha - ler,
interpretar,
correlacionar e
selecionar a alternativa
correta com base na
aprendizagem.

**Aprovação na
UC:**

70%



INFORMAÇÕES



8:50h às 10h15min: 1º período de aula

10h15min às 10h35min: **Intervalo**

10h35min às 11h40min: 2º período de aula

Nossas BIBLIOTECAS

Acervo físico:

- Livros: **+ de 300.000 volumes**
- Multimeios: **+ de 19.000 volumes**
- TCC: **+ de 18.000 títulos**



[https://portal.anhembi.br/
biblioteca](https://portal.anhembi.br/biblioteca)



Acervo eletrônico:

- Livros: **+ de 48.000 títulos**
- Periódicos: **+ de 48.000 títulos**
- Teses e Dissertações: **56 fontes**



MODO DE ACESSO

Para renovações e acesso às bases de dados, utilize seu login e senha do Ulife.



Universidade
Anhembi Morumbi

Algumas **DICAS**



Não deixe para estudar nas vésperas das avaliações. Acostume-se a estudar um pouco todos os dias. Isso fará uma enorme diferença no seu desempenho final.

Utilize uma agenda para se organizar e não perder os prazos. Existem várias opções de aplicativos para esse fim.



Fique atento aos prazos das pesquisas institucionais (CPA). Sua participação é muito importante para nosso planejamento de melhorias.

Insira na sua rotina, momentos de leitura de temas variados. Já é comprovado que a leitura desperta o raciocínio lógico e melhora sua escrita e habilidades de comunicação.





Disponível gratuitamente através do Ulife:

Acesse o Ambiente Virtual de Aprendizagem - Sala de Aula Virtual.

Em Campus, no canto esquerdo superior da tela, escolha a opção Programa de Nivelamento

Conteúdo inclui:

Avaliação Diagnóstica; Podcast; Videoaulas; Conteúdo interativo;

Infográficos; Exemplos e Questões de fixação.

Certificado:

Em cada um dos cursos, você conta com certificado e ainda pode utilizar a carga horária como atividades complementares.



A CPA - Comissão Própria de Avaliação, é a responsável pelo processo de autoavaliação institucional.

Com o objetivo de promover uma reflexão sobre a prática, compromissos com a sociedade e desenvolvimento das diferentes atividades na busca permanente e sistemática do aperfeiçoamento da nossa Universidade.



<https://portal.anhembibr/cpa/>



O NAPI é o Núcleo de Apoio Psicopedagógico e Inclusão.

Um serviço de acolhimento, aconselhamento e desenvolvimento de habilidades socioemocionais para os estudantes universitários.

Atende necessidades específicas desse público, com foco em promover competências para o desenvolvimento pessoal e profissional dos discentes.



<https://portal.anhembibr/napi/#tab1>

CONTATOS DOS COORDENADORES DE GRANDE ÁREA



Prof. Guilherme Duarte de Barros

guilherme.d.barros@animaeducacao.com.br

Câmpus Vila Olímpia



Profa. Daniele Maria Pilla Junqueira Cafange

daniele.cafange@animaeducacao.com.br

Câmpus Paulista



Lilian Montanari (coordenadora interina)

lilian.montanari@animaeducacao.com.br

Gerente do Campus Mooca



Universidade
Anhembi Morumbi



TRANS_
FORMAR
O PAÍS PELA
EDUCAÇÃO
É O QUE
NOS MOVE

ecossistema
ânima



Introdução da UC - Revisão de POO

Sistemas Distribuídos e Mobile

-
-

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.

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.

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.

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.

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).



Introdução da UC - Revisão de
POO

Classes

Classes

- Uma classe é uma declaração de tipo que agrega:
 - contante, variáveis e funções.

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.

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

```
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.");  
    }  
}
```

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.

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

```
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());  
    }  
}
```

Declaração do Objeto

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.

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.

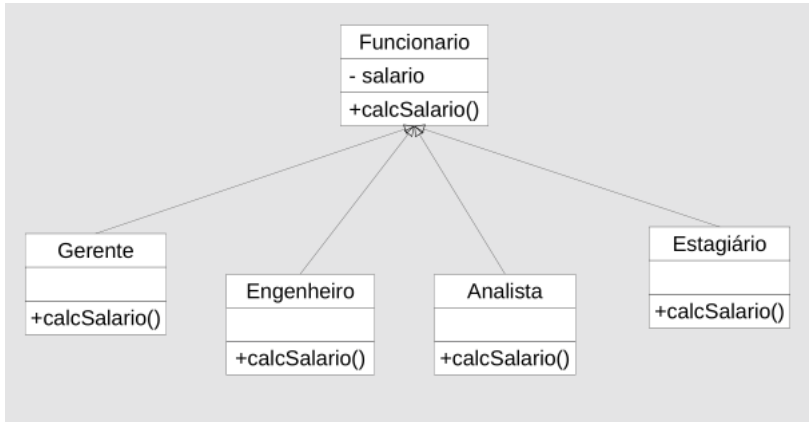
Herança

- Permite que características comuns a diversas classes sejam “herdadas” de uma classe base, ou superclasse.
- É uma forma de reutilização de software

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

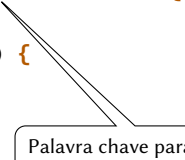
```
public class Funcionario {  
  
    protected String nome;  
    protected double salario;  
  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
    public double getSalario() {  
        return salario;  
    }  
    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;  
    }  
}  
  
public class Engenheiro extends Funcionario {  
  
    public void calcSalario() {  
        this.salario = 10000;  
    }  
}
```

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.

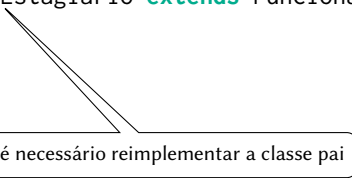
```
public class Engenheiro extends Funcionario {  
  
    public void calcSalario() {  
        this.salario = 10000;  
    }  
}
```

Herança em Java

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

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.

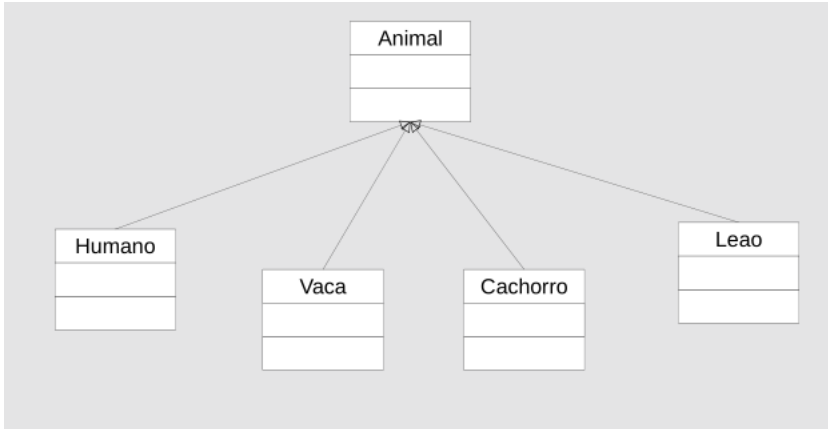
Abstração

- 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!

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()  
}
```

```
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 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.

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.

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`.

Obrigado

.