Linguagem de Programação I

Prof. Edson Kaneshima

Programação Orientada a Objetos

Aula 03

Programação Orientada a Objetos (POO)

- **Enfoque tradicional**: sistema é um conjunto de programas inter-relacionados que atuam sobre um determinado conjunto de dados que se deseja manipular de alguma forma para obter os resultados desejados.
- **Enfoque da modelagem de sistemas por objetos**: procura enxergar o mundo como um conjunto de objetos que interagem entre si e apresentam características e comportamento próprios representados por seus atributos (dados) e suas operações (métodos).

Programação Orientada a Objetos (POO) - Benefícios

- Facilita a reutilização de código e aumento da produtividade;
- Segurança por meio do encapsulamento de dados (atributos) e operações (métodos);
- Força a modularização dos sistemas;
- Oferece suporte a dados complexos e programação baseada em eventos;
- Facilidade de desenvolvimento, alteração e manutenção dos programas e aplicações.

POO – Princípios Básicos

- A programação Orientada a Objetos apoia-se basicamente nos princípios de:
  - Objeto;
  - Classe;
  - Abstração;
  - Mensagens.

Objeto

- Estrutura lógica que agrupa dados e instruções para manipular esses dados.
  - Dados (Variáveis de Instância): definem as propriedades (**Atributos**) do objeto;
  - Operações (**Métodos**): definem o comportamento do objeto.
- Extensão do conceito de objeto do mundo real.

Objeto

- Exemplo: Sistema acadêmico em que João é um aluno e Carlos é um professor que ministra aulas da disciplina de linguagem de programação. Para poder assistir às aulas da disciplina do prof. Carlos, João precisa fazer uma matrícula no curso de ciência da computação.
  - Objetos: Aluno, Professor, Curso, Disciplina, Aula, Matricula.

Classe

- Classe é uma coleção de objetos com características comuns que podem ser descritos por um conjunto de atributos e possuem operações semelhantes;
- Classe é um modelo para a criação de objetos.

Diferença entre Classe e Objeto

- Uma classe corresponde a definição de um tipo de dado ABSTRATO com as operações associadas a este tipo.

```
Classe Veiculo{
        private String modelo;
        private String cor;
        private double velocidade;
        public void acelerar( ){
                this.velocidade++;
        }
}
```

Diferença entre Classe e Objeto

- Um objeto corresponde à variável de um tipo definido, à variável de uma classe.
- Objetos instanciados: carro, avião.

```
Veiculo carro = new Veiculo( );

Veiculo aviao = new Veiculo( );
```

Métodos Acessores e Modificadores

- Diferenças conceituais entre os métodos get, set e add:
  - get somente busca o estado do objeto e o relata.
  - set e add modificam o estado do objeto.

- Métodos que modificam campos de instâncias são chamados de métodos modificadores.

- Métodos que somente acessam campos de instâncias sem os modificar são chamados de métodos acessores.

Abstração

- Consiste na habilidade de ignorar os aspectos não relevantes de um assunto para o propósito em questão, tornando possível uma concentração maior nos assuntos principais.
- Conceitos de OO aplicados na abstração:
  - Encapsulamento;
  - Herança;
  - Polimorfismo.

Mensagens

- A comunicação entre objetos ocorre exclusivamente por meio de mensagens.
- A troca de mensagens ocorre entre objetos diferentes, dentro do mesmo objetos ou quando é solicitado métodos estáticos de uma classe.
- Uma mensagem aciona um método do objeto.

Exemplo:

```
carro.acelerar( );

auxDep.addEmpregado(auxEmp);
```

Encapsulamento

- O objetivo do encapsulamento é restringir a visibilidade da informação, além de esconder como funcionam os métodos.
- Também chamado de ocultamento dos dados (Data Hiding), impede que dados sejam lidos ou modificados por outra classe. A única maneira de acesso é chamando os métodos da classe, se estiverem disponíveis (public).
- Descreve a junção de métodos e dados dentro do objeto de forma que o acesso aos dados seja permitido somente por meio dos próprios métodos do objeto.
- Encapsular é fundamental para que seu sistema seja suscetível a mudanças.
- O conjunto de métodos públicos de uma classe também é chamado de interface da classe.

Encapsulamento

- Encapsulamento dá ao objeto o comportamento de “caixa preta”.
- Uma classe pode mudar totalmente a forma como ela armazena seus dados, mas enquanto continuar a usar os mesmos métodos para manipular os dados, nenhum outro objeto saberá nem se importará com isso

Herança

- Mecanismo que permite a uma classe adquirir as propriedades de outra, permitindo a reutilização de especificações comuns.
- Dois conceitos importantes:
  - Generalização;
  - Especialização.
- Quais são as SEMELHANÇAS entre classes?

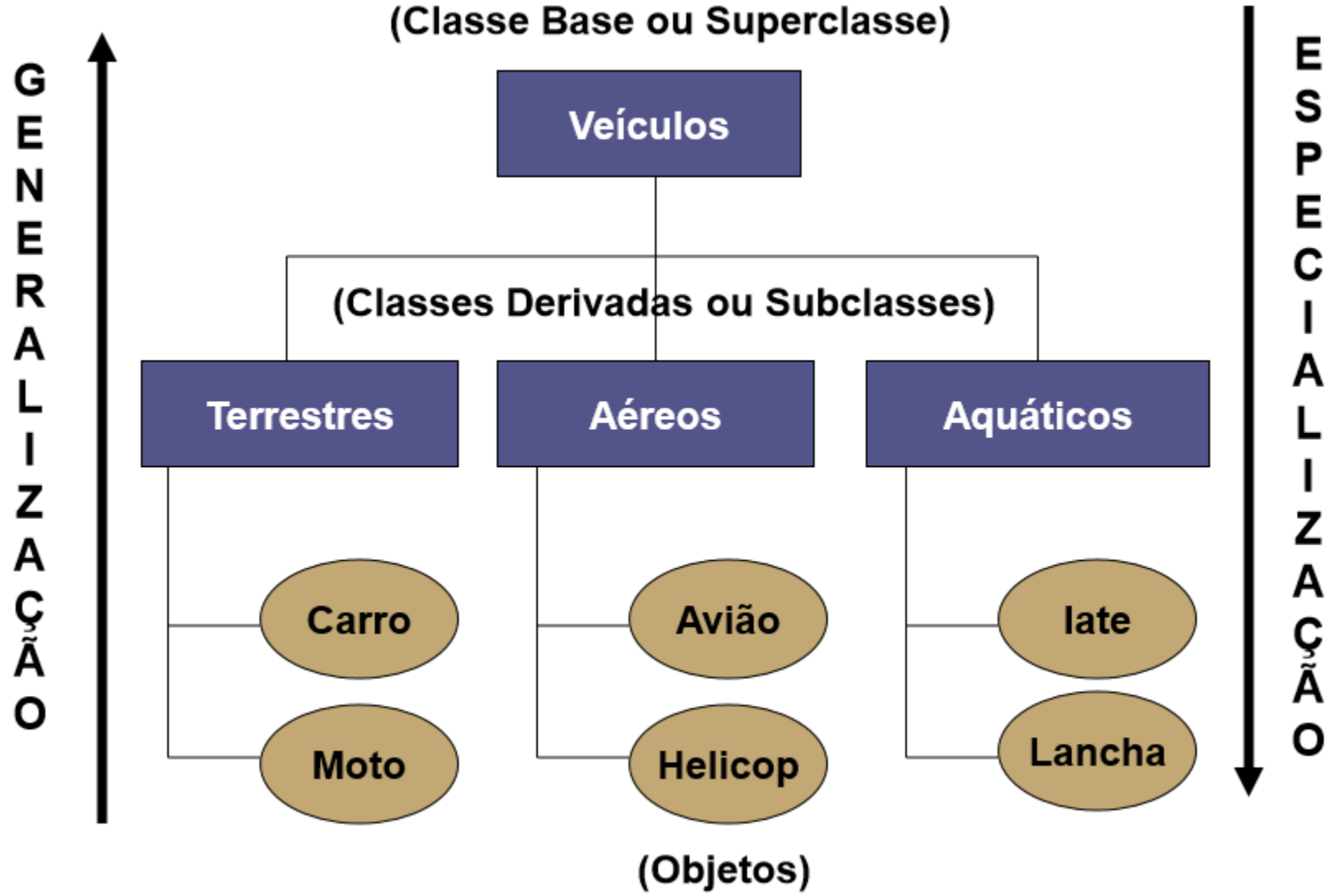
  GENERALIZAÇÃO
- Quais são as DIFERENÇAS entre classes?

  ESPECIALIZAÇÃO

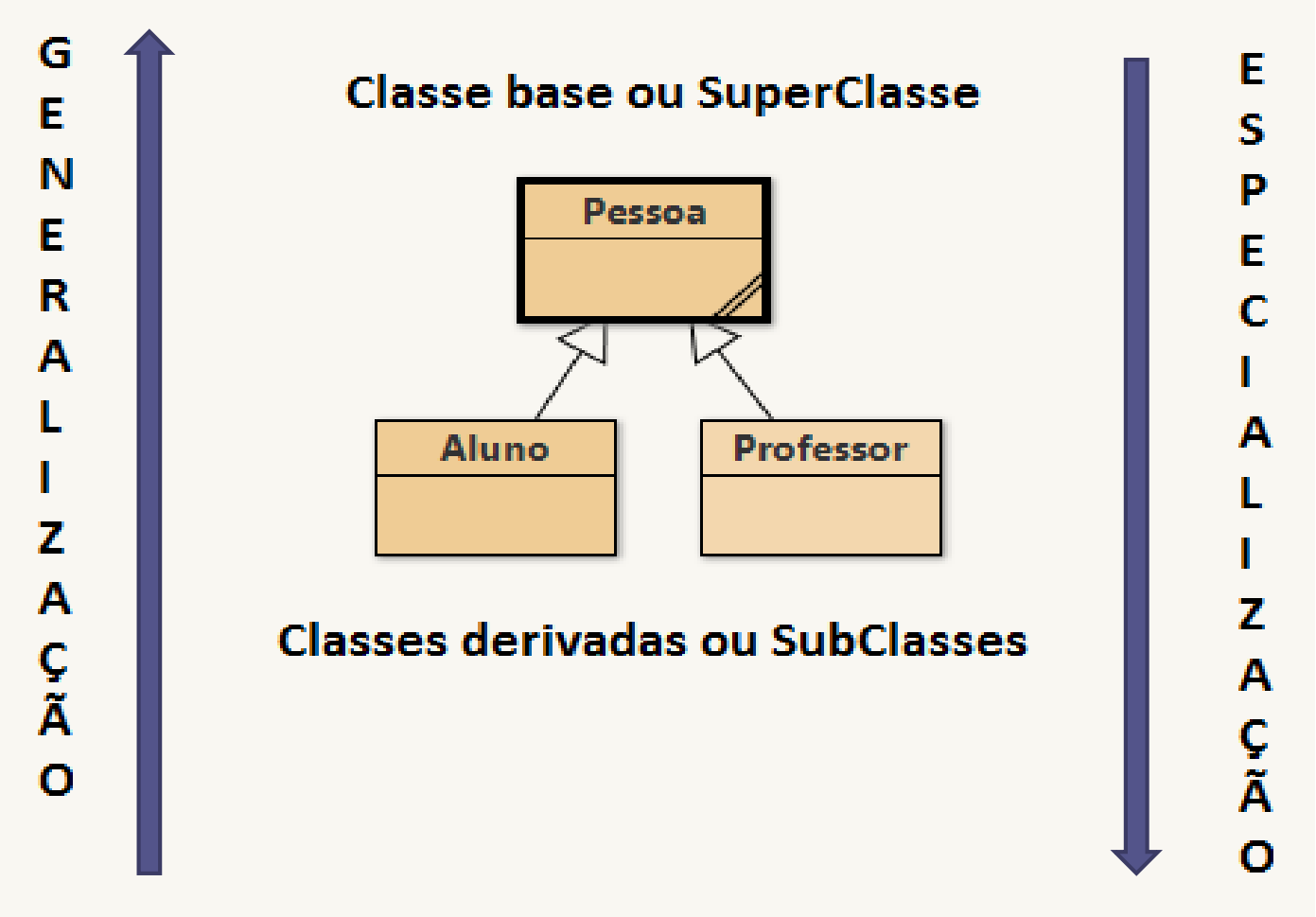
Herança

- As classes são organizadas em hierarquias de classes que tem nos níveis mais altos as classificações mais gerais.
  - A classe mais geral é chamada de **classe base** ou **superclasse**;
  - As classes mais específicas são chamadas de **classes derivadas** ou **subclasses**.

Herança


(Classe Base ou Superclasse)
Veículos
(Classes Derivadas ou Subclasses)
Terrestres
Aéreos
Aquáticos
Carro
Moto
Avião
Helicop
Iate
Lancha
(Objetos)
GENERALIZAÇÃO
ESPECIALIZAÇÃO

Herança


GENERALIZAÇÃO
Classe base ou SuperClasse
Pessoa
Aluno
Professor
Classes derivadas ou SubClasses
ESPECIALIZAÇÃO

Herança

```
public class Pessoa {
        private String nome;
        private String endereco;

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

Herança

```
        public String getEndereco() {
                return endereco;
        }
        public void setEndereco(String endereco) {
                this.endereco = endereco;
        }
}
```

Herança

```
public class Aluno extends Pessoa {
        private double notas[] = new double[8];

        public double[] getNotas() {
                return notas;
        }

        public void setNotas(int bimestre, double notas) {
                this.notas[bimestre] = notas;
        }
```

Herança

```
    public double calcularMedia(){
        double soma = 0;
        int contador = 0;
        for(int i=0; i<notas.length; i++){
            if(notas[i] != -1){
                soma = soma + notas[i];
            }
        }
        return soma / contador;
    }
}
```

Herança

```
public class Professor extends Pessoa {
        private String departamento;
        private double salario;
        public String getDepartamento() {
                return departamento;
        }
        public void setDepartamento(String departamento) {
                this.departamento = departamento;
        }
```

Herança

```
        public double getSalario() {
                return salario;
        }
        public void setSalario(double salario) {
                this.salario = salario;
        }

        public double calcularSalarioLiquido(){
                return salario * 0.05;
        }
}
```

Herança

```
public class Teste  {
    public static void main(String args[])      {
        Professor professor = new Professor();
        professor.setNome("Edson"); //método da classe Pessoa
        professor.setSalario(2000); //método da classe Professor
    }
}
```

Polimorfismo

- Polimorfismo na POO significa “muitas formas”, é a capacidade de um programa discernir, dentre os métodos homônimos, aquele que deve ser executado.
- Várias versões diferentes de um método podem receber o mesmo nome, entretanto dependendo do tipo de dado que está sendo tratado, uma versão específica é executada.
- Um objeto pode se comportar de maneira diferente ao receber uma mensagem. No polimorfismo temos dois tipos:
  - Polimorfismo estático ou sobrecarga.
  - Polimorfismo dinâmico ou sobreposição.

Polimorfismo

- Polimorfismo Estático - quando temos o mesmo método implementado várias vezes na mesma classe. A escolha de qual método será chamado depende da assinatura dos métodos sobrecarregados.
- Polimorfismo Dinâmico - quando a subclasse sobrepõe o método original. Agora o método escolhido se dá em tempo de execução e não mais em tempo de compilação. A escolha de qual método será chamado depende do tipo do objeto que recebe a mensagem.