PROGRAMAÇÃO ORIENTA A OBJETOS

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

HABNER FABRÍCIO BOESING
habner.boesing@unoesc.edu.br

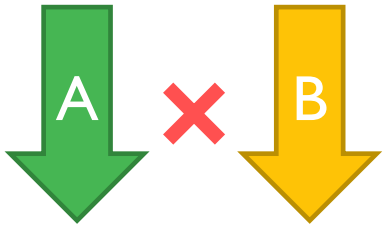
OBJETIVOS

- Paradigmas de Programação:
  - Estruturada
  - Orientada a objetos
- Composição de uma classe
- Implementação de uma classe
- Instanciação de uma classe
- Método construtor
- Abstração

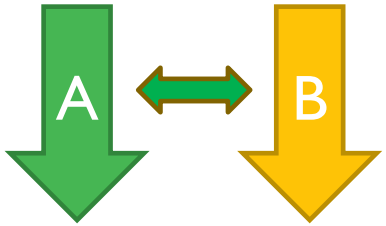
PARADIGMAS DE PROGRAMAÇÃO

Existem 2 tipos principais de paradigmas de programação diferentes:

- Estruturada:
  - Código executado em uma única sequência dentro do próprio escopo.
  - Estruturas básicas de controle: sequência, condição e repetição.
  - Programa orientada a procedimentos.
  - Procedimentos não podem ser reaproveitados em outros programas.
  - Possui melhor desempenho de processamento



- Orientada à objeto:
  - Estruturas relacionais: objetos e métodos
  - Procedimentos podem ser reaproveitados dentro de outros programas.
  - Possui melhor entendimento de linguagem, em detrimento do desempenho de processamento.
  - Reaproveitamento de código



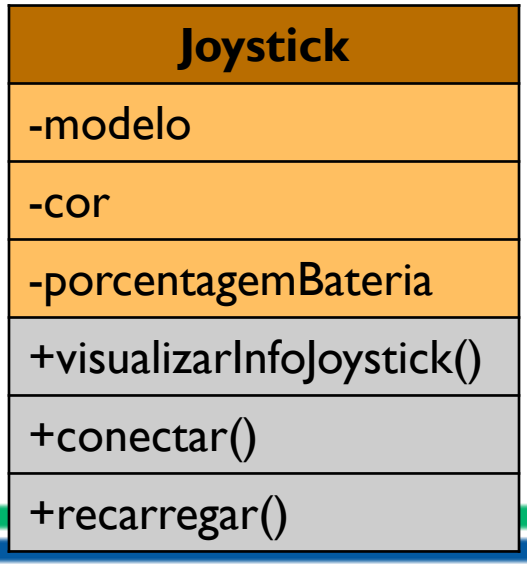
PARADIGMAS DE PROGRAMAÇÃO

Estruturada:

Orientada à objeto:

CLASSE, ATRIBUTOS E MÉTODOS

- A Programação Orientada a Objetos (POO) possui **3 elementos** principais:
- **Classe**: classe é um conjunto de características e comportamentos que definem o conjunto de objetos. Basicamente é a estrutura da qual são derivados os objetos. O nome da classe sempre começa em maiúsculo.
- **Atributo**: definem as características que o objeto pode possuir. Os nomes dos atributos são sempre escritos em camelCase.
- **Método**: define as ações que o objeto pode realizar. O nome dos métodos são sempre escritos em camelCase, seguidos de ().



CLASSE - EXEMPLO

- Exemplo de implementação de uma classe em Java:

Representação

Joystick
-modelo
-cor
-porcentagemBateria
+visualizarInfoJoystick()
+conectar()
+recarregar()

Implementação

```java
public class Joystick {
    private String modelo;
    private String cor;
    private int porcentagemBateria;

    public void visualizarInfoJoystick(){
        //código para visualizar informações do joystick
    }
    public void conectar() {
        //código para conectar o joystick ao vídeo-game
    }
    public void recarregar(){
        //código para recarregar o controle
    }
```

UNOESC

INSTANCIAR UMA CLASSE

- Uma das formas de relacionar os elementos (atributos e métodos) de uma classe para outra é realizar a instanciação da classe por meio de um objeto.
- Exemplo:

Classe Joystick – Estrutura do Objeto

```
public class Joystick {

    String modelo;
    String cor;
    int porcentagemBateria;

    void visualizarInfoJoystick(){...}
    void conectar() {...}
    void recarregar(){...}
}
```

Classe Principal – Criação do Objeto

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

        Joystick js1 = new Joystick();
        js1.modelo = "PS4";
        js1.cor = "Preto";
        js1.porcentagemBateria = 100;

        js1.visualizarInfoJoystick();
        js1.conectar();
        js1.recarregar();
```

Desta forma, na classe principal, por meio do objeto **js1** será possível atribuir valores aos atributos criados na classe joystick, assim como executar os métodos da classe joystick.
Em resumo a classe joystick representa apenas a estrutura do objeto joystick e na classe principal criamos de fato o objeto joystick, dando "vida" ao objeto a atribuindo suas características.

INSTANCIAR UMA CLASSE – MÉTODO CONSTRUTOR

- Ao instanciar uma classe é possível já definir parâmetros que queremos atribuir ao objeto por meio de um método construtor.
- Exemplo:

Classe Joystick – Estrutura do Objeto

```
public class Joystick {
    private String modelo;
    private String cor;
    private int porcentagemBateria;

    public Joystick() {
    }

    public Joystick(String m, String c, int pb) {
        this.modelo = m;
        this.cor = c;
        this.porcentagemBateria = pb;
    }
```

Classe Principal – Criação do Objeto

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

        Joystick js1 = new Joystick();
        js1.setModelo("PS4");
        js1.setCor("Preto");
        js1.setPorcentagemBateria(100);
        System.out.println("Modelo: " + js1.getModelo());

        Joystick js2 = new Joystick( m: "PS5",  c: "Branco",  pb: 0);

        js1.visualizarInfoJoystick();
        js2.visualizarInfoJoystick();
```

Desta forma, escolher se queremos criar o objeto js1 padrão, ou seja, sem modelo, cor e porcentagemBateria definidos, ou então podemos criar o objeto já passando estas informações por meio do método construtor da classe Joystick. Lembrando que é possível ter mais de um método construtor na classe.

POO - CARACTERÍSTICAS

- As 4 características principais que caracterizam uma POO (Programação Orientada a Objeto) são:
- **Abstração**
- **Encapsulamento**
- **Herança**
- **Polimorfismo**

ABSTRAÇÃO - CRIAÇÃO DE MÚLTIPLOS OBJETOS

- A partir da instanciação de uma classe é possível criar múltiplos objetos daquele mesmo tipo, permitindo inclusive que cada objeto receba características diferentes, criando assim a sua própria identidade. A esta características damos o nome de *abstração*.

Classe Principal

Criação de Diversos Objetos a partir da classe Joystick

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

        Joystick js1 = new Joystick();
        js1.modelo = "PS4";
        js1.cor = "Preto";
        js1.porcentagemBateria = 100;

        Joystick js2 = new Joystick();
        js2.modelo = "PS5";
        js2.cor = "Branco";
        js2.porcentagemBateria = 0;
```

Objeto js1

Objeto js2

ATIVIDADE PRÁTICA

- 1) Crie uma classe chamada Pessoa, com os atributos **nome** e **idade**. Crie um método construtor completo que requisite estes atributos. Crie também o método **exibirInfo()**, o qual deve mostrar na tela o nome e idade da pessoa.

Na classe principal, crie 2 objetos do tipo pessoa e depois execute o método exibirInfo() para cada uma delas.

ATIVIDADE PRÁTICA

- 2) Crie uma classe chamada **Veiculo**, com os atributos **ano**, **modelo**, **cor** e **quilometragem**. Crie um método construtor completo que requisite estes atributos. Crie também os métodos:
- **verificarManutencao()** – Este método irá verificar a quilometragem do veículo. Se o veículo possuir menos de 25 mil km, deve-se exibir a mensagem "Tudo ok!", caso possua entre 25 mil e 75 mil km, deve-se exibir a mensagem "Realizar revisão parcial!". Caso possua mais de 75 km deve-se exibir a mensagem "Realizar revisão completa!".
- **mudarCor(String cor)** – Este método deverá receber por parâmetro uma cor e esta deverá sobrescrever a cor anterior presente no atributo do objeto.
- **exibirCor()** – Este método deverá exibir na tela o valor presente no atributo cor do objeto.

Na classe principal crie um objeto do tipo Veiculo, e execute os métodos deste objeto na seguinte ordem:

- verificarManutencao()
- exibirCor()
- mudarCor()
- exibirCor()