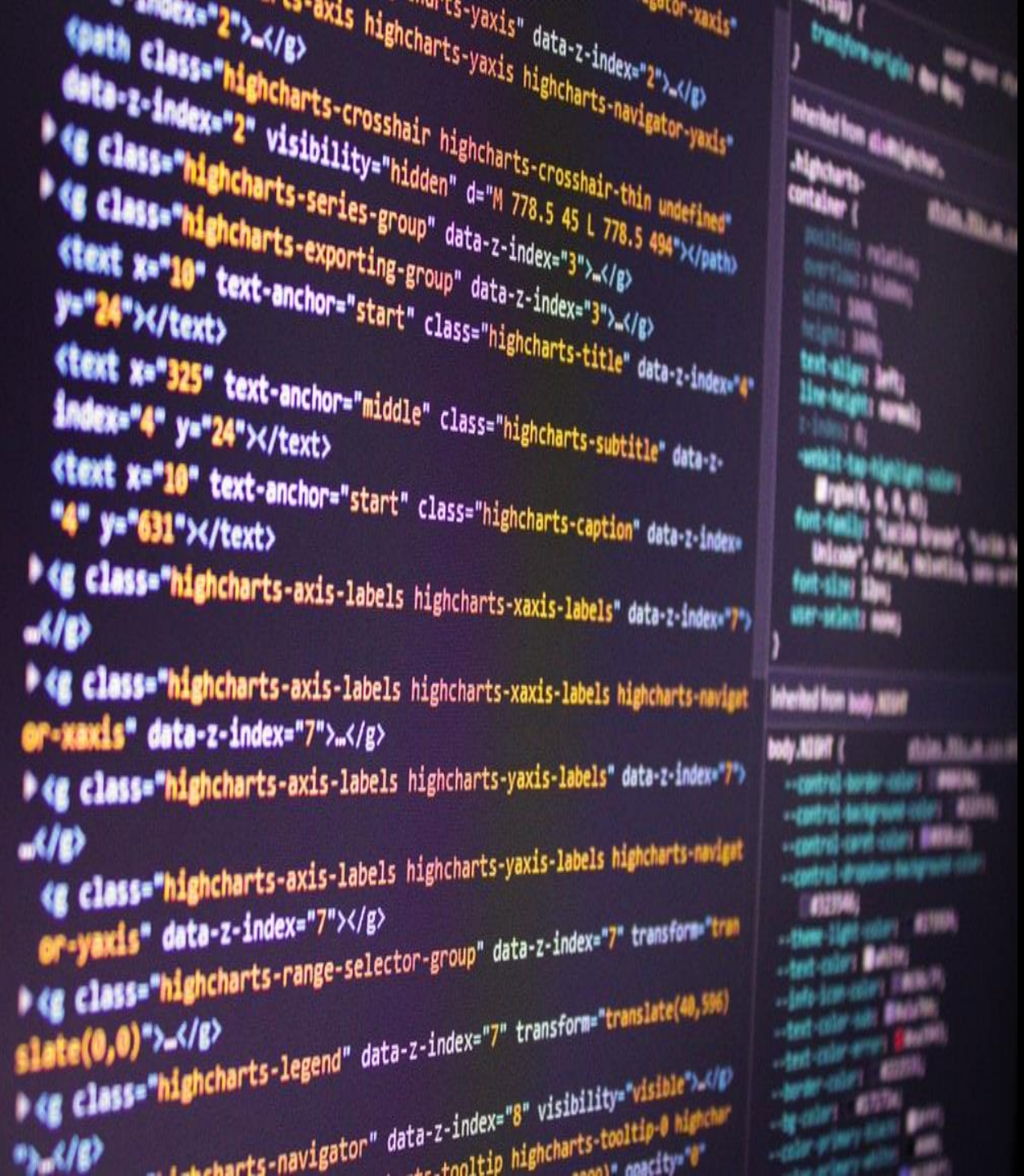


Projeto Integrador II

PROF. DR. FERNANDO T. FERNANDES

Senac

Orientação a Objetos

- Introdução

O que é Orientação a Objetos

De maneira geral, a orientação a objetos permite **descrever, por meio de trechos de código, objetos do mundo real e suas relações**.

Linguagem mais próxima a como pensamos e nos comunicamos !

Algumas linguagens com suporte a OO

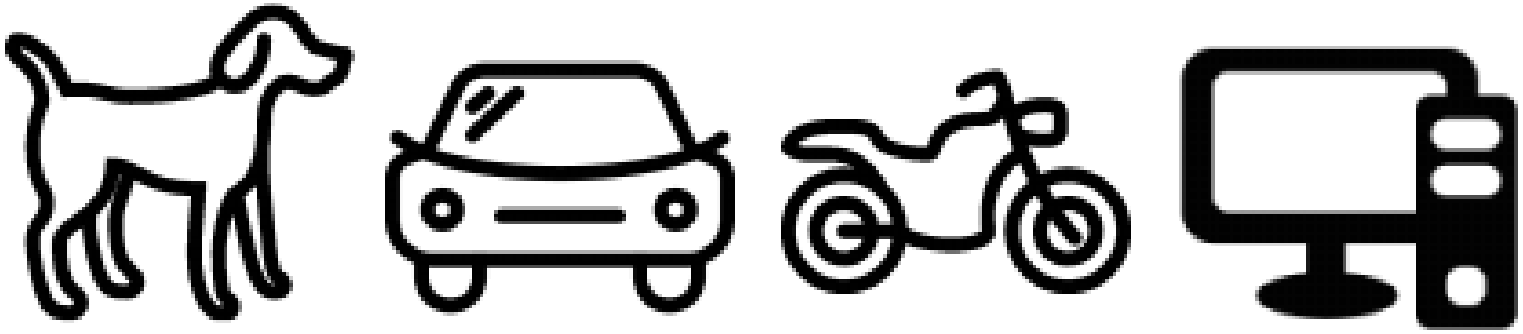
Java	C#	Python	C++
Javascript	PHP	Kotlin	Outras

Classes e Objetos

Classe

Representação da ***estrutura***, ***propriedades*** , ***comportamentos e ações*** de um objeto real ou abstrato em um código.

- Ex: Veículo, Computador, Animal



Se fossemos Henry Ford, que implementou a linha de montagem para automóveis, como especificaríamos um projeto de um Carro? O que qualquer carro a ser produzido **deveria possuir**?

- Todo carro **possui**:
  - Motor
  - Rodas
  - Volante
  - Portas
  - Pedal
  - Marca
  - Modelo
  - Cor

Classe

Classe

- Quais **ações** um carro pode fazer?

- Acelerar
- Frear
- Virar

- Uma classe define um **modelo conceitual** de um objeto **real**.
  - Exemplo: em uma classe Carro, é definido **como** um carro pode ser construído e **o que** ele pode fazer**.**

Classe
Projeto

- Ações = **Métodos**
  - Acelerar, Frear
- Partes do Carro = **Atributos**
  - Rodas, Cor, Modelo

Objetos
Construção

Carros reais com cores, motores e modelos diferentes

Projetando uma classe Imóvel

- Quantos cômodos terá cada apto?
- Qual a metragem?
- Quantas suítes?

Classe

Imóvel

Características:

número de cômodos
suítes
banheiros
metragem
preço

Comportamentos:

Verificar data e disponibilidade de locação
Verificar se está ocupado
Obter número de quartos

Classe

Projeto

Características: número de cômodos, banheiros, suítes metragem, etc.

Métodos: verificar disponibilidade

Objetos

Construção

Apartamentos do mesmo tipo construídos e com diferentes moradores.

Definindo uma Classe

Classe

Projeto

- Ações = Métodos
  - Acelerar, Frear
- Partes do Carro = Atributos
  - Rodas, Cor, Modelo

Implementação em Java

```
//Nome do Projeto
public class Carro {

//Atributos
   private static String marca =  “Ford";

   private String[] Rodas;
   private String volante;
   private String cor;
   private String modelo;
   private int velocidadeAtual;

//Ações
public void acelelar(){
}

public void acelelar(int numVelocidade){
}

public void frear(int numVelocidade){
}

} // Fim da classe
```

Classe - Construtores




Definem como construir um objeto de uma classe

Podem especificar um comportamento padrão ao **criar** os objetos da classe.

Ex: Construam carros na cor preta!

Classe - Construtores

Sintaxe - public NomeDaClasse()

- Ex: Toda vez que eu construir um carro, criarei na cor preta.

```
public Carro() {
    this.cor = “preto”;
}
```

É possível ainda passar parâmetros ao criar um novo objeto. Ex: Criar um objeto com um modelo especificado

```
public Carro(String nomeModelo) {
    this.cor = “preto”;
    this.modelo = nomeModelo;
}
```

Classe

Projeto

- ✓ Construir na cor Preta

Implementação em Java

```
//Nome do Projeto
public class Carro {

   //Atributos
   private String cor;

   //Construtor Padrão – sem parâmetros
   public Carro(){
       this.cor= “preto”;
   }

   //Sobrecarga de construtor – com parâmetro
   public Carro(String nomeCor){
       this.cor = nomeCor;
   }

} //término da definição da classe
```

Classe - Métodos

- Definem as **ações** da classe
- Ações a serem executadas após o **OBJETO** da classe ser construído.

```
//Acelerar com uma velocidade constante
 public void acelerar(){
     this.velocidadeAtual += 10;
 }

//Frear a uma velocidade constante
 public void frear(){
     this.velocidadeAtual -= 10;
 }
```

- Obs: Também podem especificar o comportamento padrão de uma **CLASSE** com o uso da palavra-chave **static.**

Classe - Métodos

Classe

Projeto

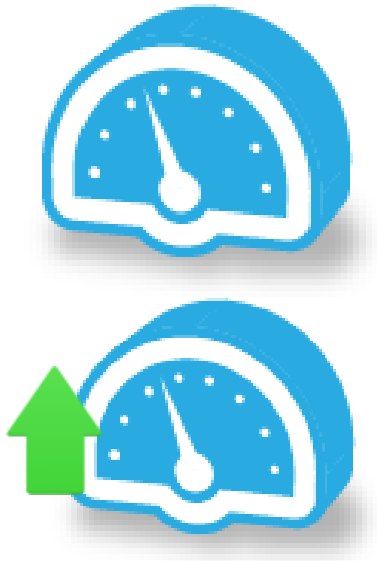
- ✓ Construir na cor Preta
- ✓ Carro pode acelerar e frear constantemente

Implementação em Java

```
public class Carro {

   //Atributos
   private String cor;
   private int velocidadeAtual;

   //Construtor Padrão – sem parâmetros
   public Carro(){
       this.cor= “preto”;
   }

   //Acelerar com uma velocidade constante
   public void acelerar(){
       this.velocidadeAtual += 10;
   }

   //Frear a uma velocidade constante
   public void frear(){
       this.velocidadeAtual -= 10;
   }
} //término da definição da classe
```

Classe – Métodos - Sobrecarga

- Pode ocorrer das ações serem executadas de diferentes maneiras
  - Ex: O carro poderá ter aceleração constante **E** também aceleração esportiva definida pela intensidade do pedal



Classe – Métodos - Sobrecarga

- Métodos podem não receber nenhum parâmetro ou receber um ou mais parâmetros

```
public void acelerar(){
    this.velocidadeAtual += 10;
}

public void acelerar(int numVelocidade) {
     this.velocidadeAtual += numVelocidade;
 }

public void acelerar(int numVelocidade, int valorMaximo) {
   if(velocidadeAtual<valorMaximo){
       this.velocidadeAtual += numVelocidade;
   }
}
```

Não recebe parâmetros

Sobrecarga que recebe um valor inteiro a acelerar

Sobrecarga que recebe um valor inteiro e limita a aceleração

Classe – Métodos – Tipos de Retorno

- Podem não retornar nada após a execução do método;

```
public void acelerar(){
        this.velocidadeAtual += 10;
   }
```

- Ou podem retornar valores após sua execução

```
public boolean isCarroParado(){
        return (velocidadeAtual == 0);
    }
```

Classe - Variáveis de Classe vs Variáveis do Objeto/Instância

- Atributos (ou variáveis) de classe são comuns a todos os objetos a serem construídos

```
//Variáveis da Classe
private static String marca =  “Ford";
```

  - Ex: Todo carro que será construído na minha fábrica será da marca “Ford”.

- Variáveis da Instância são particulares ao objeto

```
//Variáveis da Instância
private String cor;
private String modelo;
```

  - Ex: Cada carro que eu construir poderá ter cores e modelos diferentes.

Classe - Resumo

Classe

Projeto

- ✓ Carro da marca Ford
- ✓ Construir na cor Preta
- ✓ Carro pode acelerar e frear constantemente ou sob demanda

Implementação em Java

```
public class Carro {

  //Variáveis da Classe
   private static String marca =  “Ford";

   //Atributos
   private String cor;
   private int velocidadeAtual;

   public Carro() {
      this.cor = “preto”
   }

   public void acelerar(){
      this.velocidadeAtual+=10;
   }

   public void acelerar(int numVelocidade){
       this.velocidadeAtual+=numVelocidade;
   }

}
```

Construindo objetos de uma classe

Objetos

- Objetos são **construções** da classe

- Também chamados de **instância** da classe

Objetos

- Não andamos em conceitos, andamos em carros...
- Para poder andar em um carro precisamos **construir** um carro.
- Quando construímos carros, podemos construir carros com diferentes modelos e propriedades como, por exemplo, carros com motores 1.0 ou 2.0, de cores preta, branca, prata, etc.
- Então, **um objeto é uma construção de uma classe**, que contém propriedades reais**.**
  - Ex: Carro **objMeuCarro**= **new** Carro();
    objMeuCarro.setCor("prata");

Métodos Acessores (*Getters)* e Modificadores *(Setters)*

- Servem para controlar o acesso aos campos (propriedades da classe ou objeto).

```
//Retornar a cor do carro
public String getCor() {
    return cor;
}

//Definir a cor do carro
public void setCor(String cor) {
    this.cor = cor;
}
```

```
//Retornar a velocidade do Carro
public int getVelocidade() {
    return velocidadeAtual;
}

//Alterar a velocidade do carro
public void setVelocidade(int novaVelocidade) {
    this.velocidadeAtual = novaVelocidade;
}
```

- Exemplo:

```
//Instancio um objeto da classe Carro
Carro meuCarro = new Carro();

//Faço a pintura preta
meuCarro.setCor("Preta");

//Consulto a cor do carro
String cor = meuCarro.getCor();
```

Métodos Acessores (*Getters)* e Modificadores *(Setters)*

- Também podem executar um tratamento prévio antes de retornar um valor.

```
//Defino um método acessor que tranforma o modelo do carro em letras maíusculas
public String getModelo() {
    return modelo.toUpperCase();
}
```

- Exemplo: Instanciar um objeto da classe carro, definir seu modelo e exibir seu modelo

```
Carro meuCarro = new Carro();
meuCarro.setModelo("gol");                                  //Defino o Modelo

String modeloExibir = meuCarro.getModelo();                 //Consulto o Modelo

JOptionPane.showMessageDialog(this, modeloExibir );  //Exibo o modelo
```

Palavra-chave *this*

- Serve para acessar os **atributos** ou métodos de um objeto construído.

```
public Carro() {
    this.cor = “preto”;
 }
```

Objetos - Traduzindo para o mundo Java...

Descrição	Código em Java
Ok. Terminei o projeto. Construa meu carro!	Carro carroPrincipal = new Carro();
Pinte meu carro na cor branca	carroPrincipal.setCor(“Branco”).
Construa outro carro!	Carro carroReserva = new Carro();
Pinte meu carro na cor preta	carroReserva.setCor(“Preto”).
Qual a cor do meu carro principal?	String cor = carroPrincipal.getCor();

Padrão de Nomenclatura

Por que usar um padrão de Nomenclatura?

Facilitar a leitura do código e a manutenibilidade

Diferencia uso de classes e objetos

A própria linguagem Java usa padrão de nomenclatura – Java Naming Convention

Ex:JButton,JTextField

Padrão de Nomenclatura

Nome	Padrão
Pacotes	Usar letras minúsculas. Pacotes internos nomeaplicacao. Ex: aula3 Pacotes Públicos Nome invertido do site da empresa Ex: http://www.sp.senac.br br.senac.sp.nomeaplicacao
Classes	Inicia com uma letra maiúscula e segue em letras minúsculas. Se tiver mais de uma palavra, cada palavra iniciará com uma letra maiúscula. Se for uma sigla, também usar letra maiúscula. Ex: Carro, JMenuBar, JOptionPane
Métodos	Inicia com letras minúsculas. Para mais de uma palavra, cada palavra subsequente inicia em letra maiúscula. – conhecido como Camel-Case Usar na primeira palavra um Verbo Ex: acelerar() acelerarVeiculo()

Padrão de Nomenclatura

Nome	Padrão

Construtor	Mesmo nome da Classe
	
	public class Carro{
	public Carro(){
	
	}
	}

Campos (Variáveis de classe ou de objeto)	Camel-Case Mesmo padrão de métodos Ex: String volante Int velocidadeAtual
Parâmetros	Camel-Case acelerar(int velocidade)
Variáveis e objetos	Camel-Case Ex: JFrame telaInicial = new JFrame(); String nome = “”;

Lab – 2 – Introdução a Orientação a Objetos

Clone o **projeto** chamado “lab2” do GitHub e siga as instruções

Para cada tarefa, adicione um comentário, faça o *commit* e suba (*push*) as alterações para o GitHub.

Tipo de tarefa: **Individual**

Tempo Estimado: **30 minutos**

ADO 1 – Introdução a Orientação a Objetos

1. Crie um projeto **java desktop no NetBeans** chamado “seunomeSobrenomeADO1”.

   Ex: JoaoSilvaADO1

2. Escolha um animal que inicie com a mesma letra que seu nome (ex: Fernando – F):

   Foca, Formiga, etc.

3. Defina **uma propriedade nome** e mais **pelo menos duas** propriedades de seu animal (O que o seu animal possui) **além do nome**

4. Crie os métodos assessores (Getters e setters) para as propriedades criadas

5. Crie um construtor vazio

6. Crie ao menos um construtor que receba parâmetros e defina uma propriedade ao criar um objeto desta classe

7. Crie pelo menos dois métodos referentes ao que o seu animal pode fazer.

8. Instancie dois objetos da sua classe e exiba o nome dos animais criados.

9. **Exporte o projeto para zip e suba ao BlackBoard**

Tipo de tarefa: **Individual**

Tempo Estimado: **30 minutos**

Obrigado!

proffernandotfernandes

@proffernandotfernandes