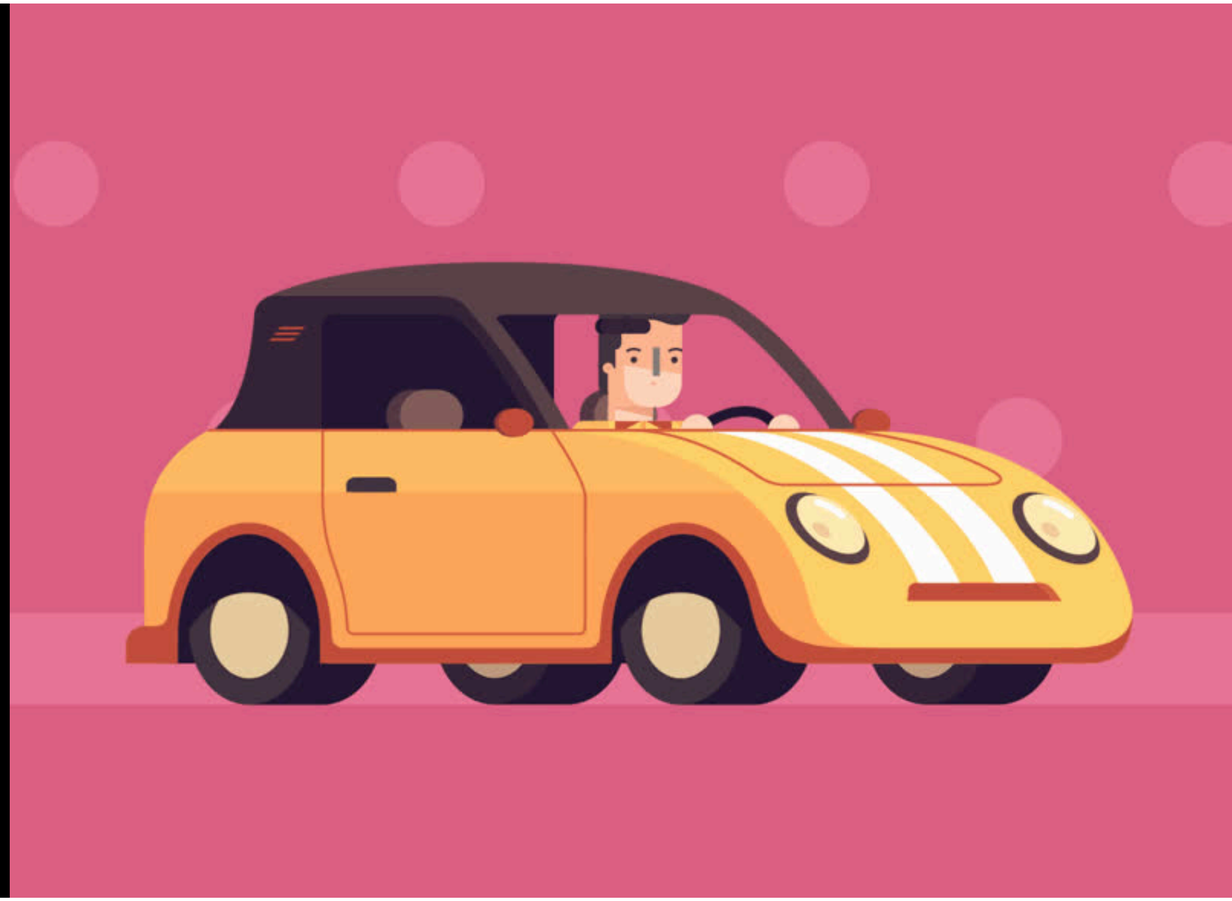
Encapsulamento

Aula 02

Professor Arthur Giangiarulo

Classes

```
public class Carro {
    private String modelo;
    private int ano;
    private String fabricante;
    private String proprietario;
    private double valor;
    private String placa;
```

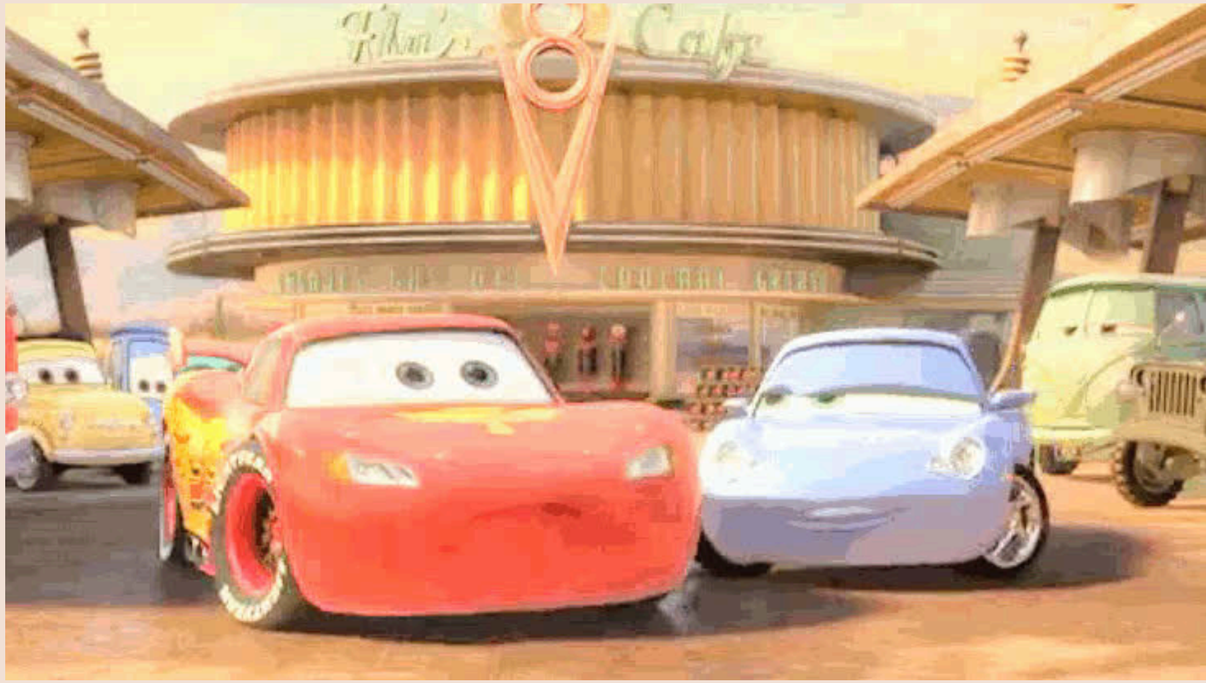


Classes

- Em Programação Orientada a Objetos, a classe é um dos principais pilares.
- Ela serve como um molde para criar objetos. Pode-se pensar em uma classe como um esboço que descreve os detalhes técnicos do que o objeto deve ser, mas sem ser o objeto em si.
- Em Java, uma classe não só define atributos (dados), mas também comportamentos (métodos).

Objeto

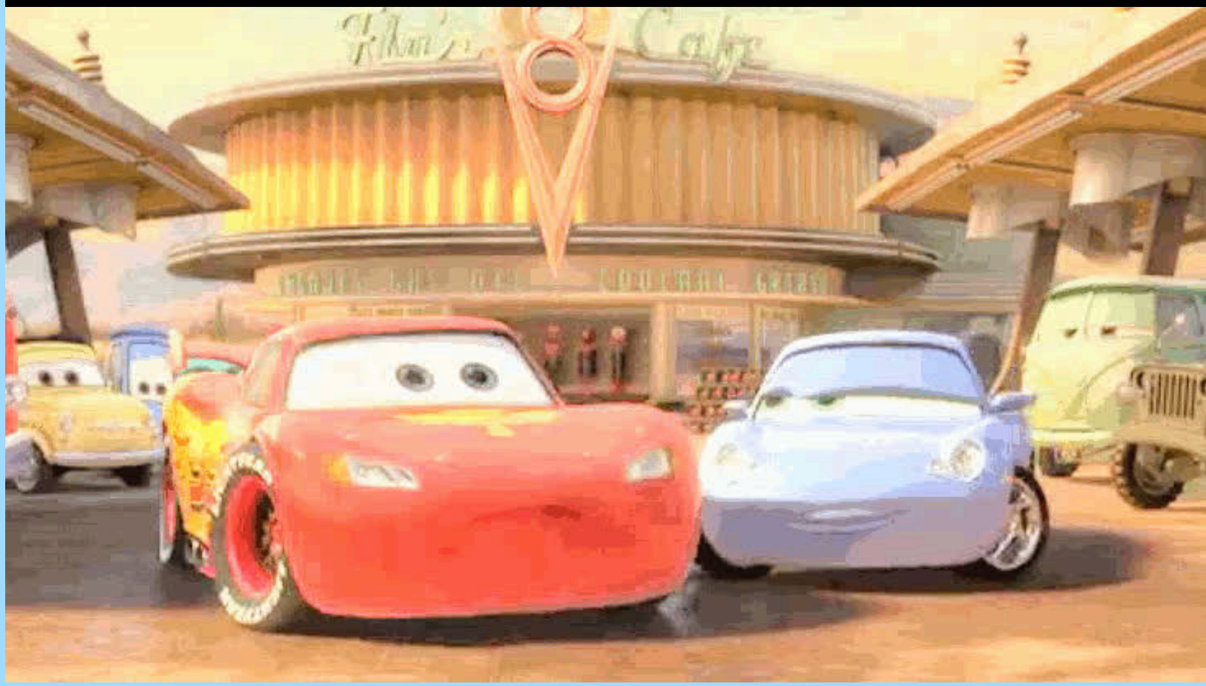
- objeto é a realização física e concreta de uma classe. Se a classe é o esboço, o objeto é a casa construída com base nesse esboço.
- Um mesmo molde (classe) pode ser usado para criar várias instâncias (objetos).



Atributos

- Também conhecidos como propriedades ou campos, são as variáveis definidas dentro de uma classe.
- Representam o estado de um objeto. No exemplo de um Carro, o estado pode incluir coisas como cor, modelo, ano, entre outros.

```
private String cor;
private String modelo;
private int ano;
```



Métodos

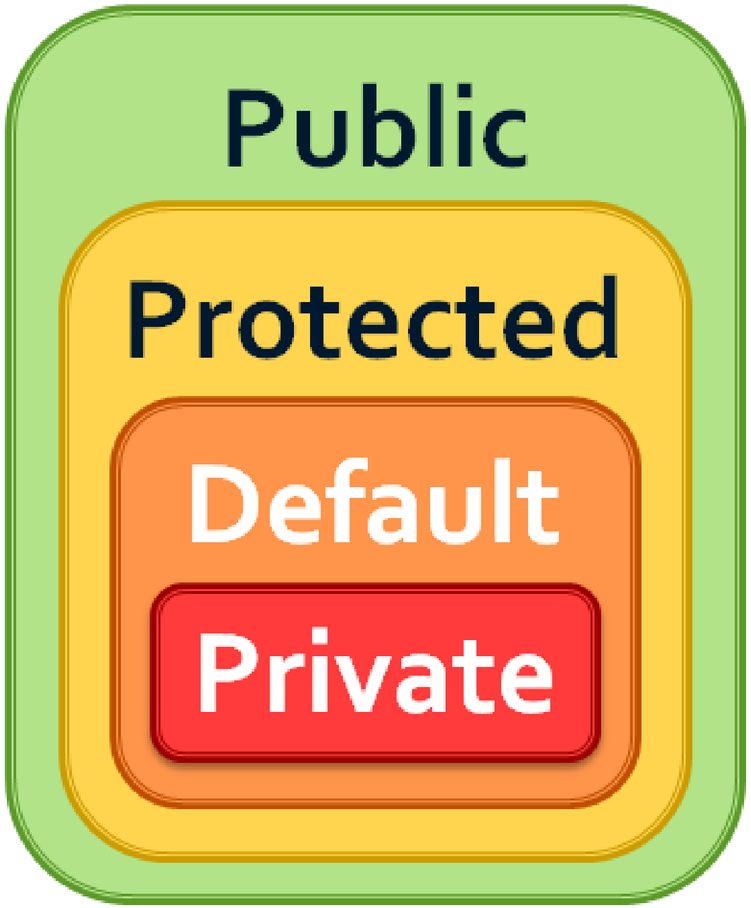
```
public double calcularIPVA() {
    return this.valor * 0.04;  // 4% do valor do carro
}
```

- Métodos são essencialmente funções definidas dentro de uma classe e representam as ações que um objeto pode executar.
- Eles operam nos dados de um objeto e podem modificar o estado de um objeto.

Encapsulamento

- Refere-se à restrição do acesso direto a alguns componentes do objeto e à provisão de métodos públicos para interagir com esses componentes.

- Em Java, isso é realizado usando modificadores de acesso como private, protected e public.



Modificadores

- **Public**
- Quando um membro de uma classe é definido como public, ele pode ser acessado por qualquer outra classe, independentemente do pacote.
- É o modificador de acesso mais aberto, sem restrições.

```
public class Pessoa {
    public String nome;

    public Pessoa(String nome) {
        this.nome = nome;
    }
}
```

Modificadores

- **Protected**
- Membros marcados como protected podem ser acessados dentro da mesma classe, por subclasses, e também por outras classes no mesmo pacote.

```
public class Veiculo {
    protected String marca;

    public Veiculo(String marca) {
        this.marca = marca;
    }
}
```

Modificadores

- **Default**
- Quando nenhum modificador de acesso é especificado, o membro tem acesso de pacote por padrão, o que significa que pode ser acessado por qualquer classe dentro do mesmo pacote, mas não por classes fora desse pacote ou por subclasses fora desse pacote.

```
class PacoteItem {
    String descricao;

    PacoteItem(String descricao) {
        this.descricao = descricao;
    }
}
```

Modificadores

- **Private**
- Quando um membro de uma classe (seja ele um atributo ou método) é marcado como private, ele só pode ser acessado diretamente dentro da própria classe em que foi definido.

```java
public class ContaBancaria {
    private double saldo;

    public ContaBancaria(double saldoInicial) {
        this.saldo = saldoInicial;
    }

    public double consultarSaldo() {
        return saldo;
    }
```

Visibilidade	public	protected	default	private
A partir da mesma classe	✓	✓	✓	✓
Qualquer classe no mesmo pacote	✓	✓	✓	⊘
Qualquer classe filha no mesmo pacote	✓	✓	✓	⊘
Qualquer classe filha em pacote diferente	✓	✓	⊘	⊘
Qualquer classe em pacote diferente	✓	⊘	⊘	⊘

Métodos getters e setters

- Para manipular atributos privados utilizamos os getters e setters.
- Getters e setters são padrões comuns em Java para encapsular o acesso aos atributos de uma classe. Eles permitem maior controle sobre como os atributos são acessados e modificados.

Métodos get e set

```java
public class Pessoa {
    private String nome; // Atributo privado

    // Método getter para o atributo nome
    public String getNome() {
        return nome;
    }

    // Método setter para o atributo nome
    public void setNome(String nome) {
        this.nome = nome;
    }
```

```java
public static void main(String[] args) {
    Pessoa pessoa = new Pessoa();

    // Utilizando o método setter para definir o nome
    pessoa.setNome("João");

    // Utilizando o método getter para obter o nome
    System.out.println(pessoa.getNome());  // Saída: João
}
```

Exercícios

- Vamos montar nossas classes com os métodos Getters e Setters