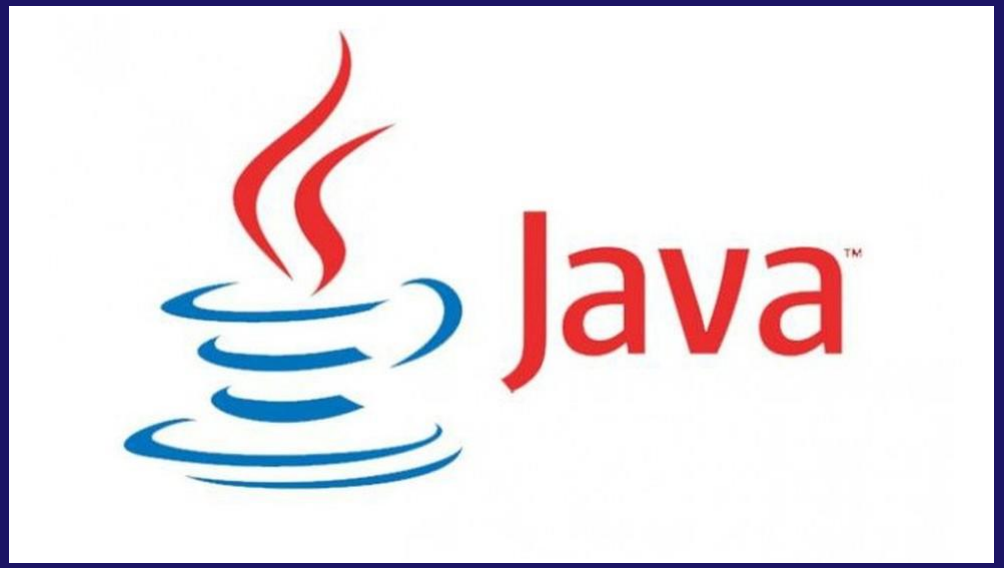
Curso Técnico em Desenvolvimento de Sistemas

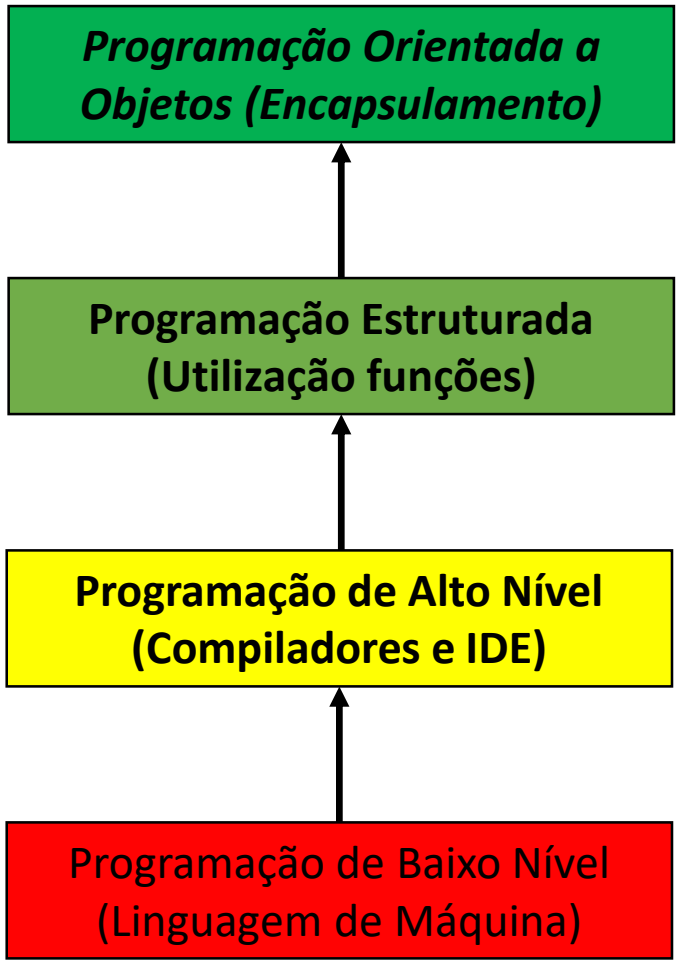
UCR8 – Desenvolver Interface Gráfica Java

Aula 07 – Revisão de Orientação a Objetos



Programação Orientada a Objeto - POO

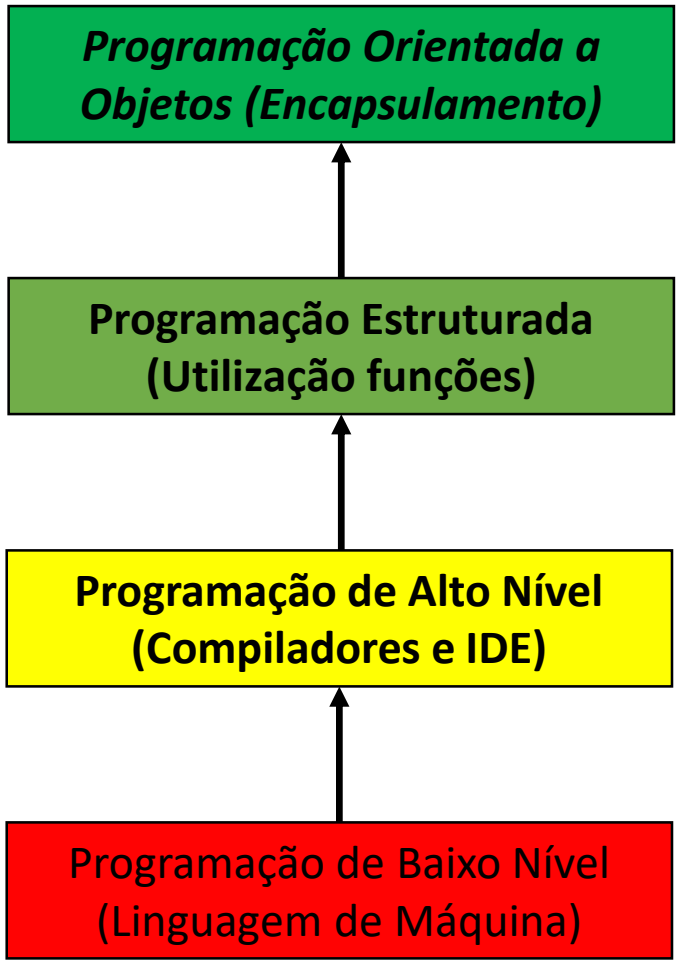
- Aproximar o mundo digital do mundo real.



- Programação em linguagem de máquina;
- Muito difícil, enfadonha.

Programação Orientada a Objeto - POO

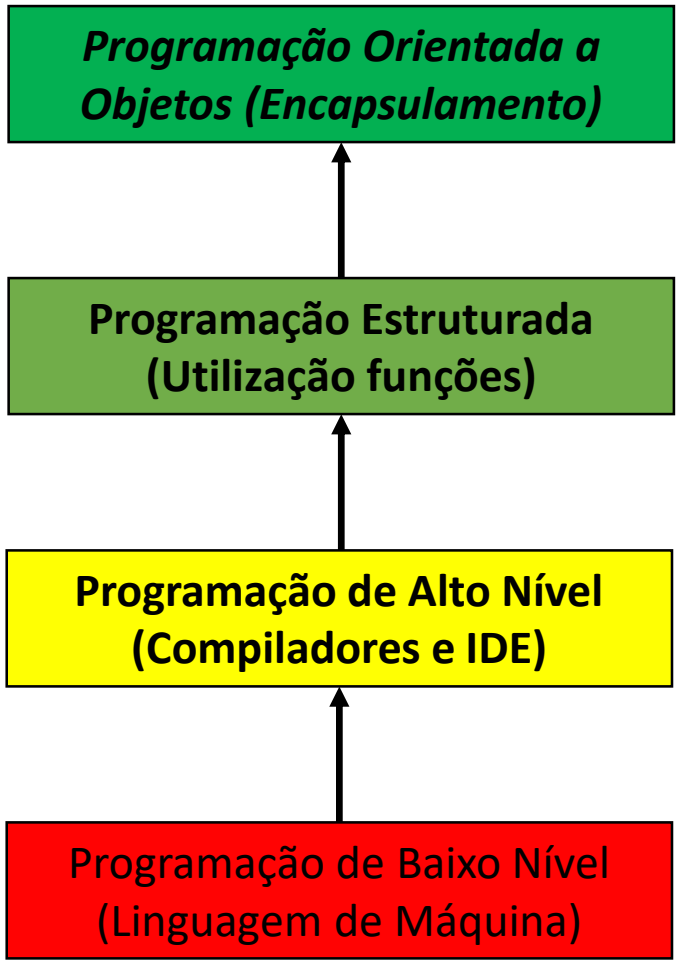
- Aproximar o mundo digital do mundo real.



- Programação em linguagem entendida por humanos;
- Execução operações sequenciais;
- Dificuldade em manutenção e execução dos programas.

Programação Orientada a Objeto - POO

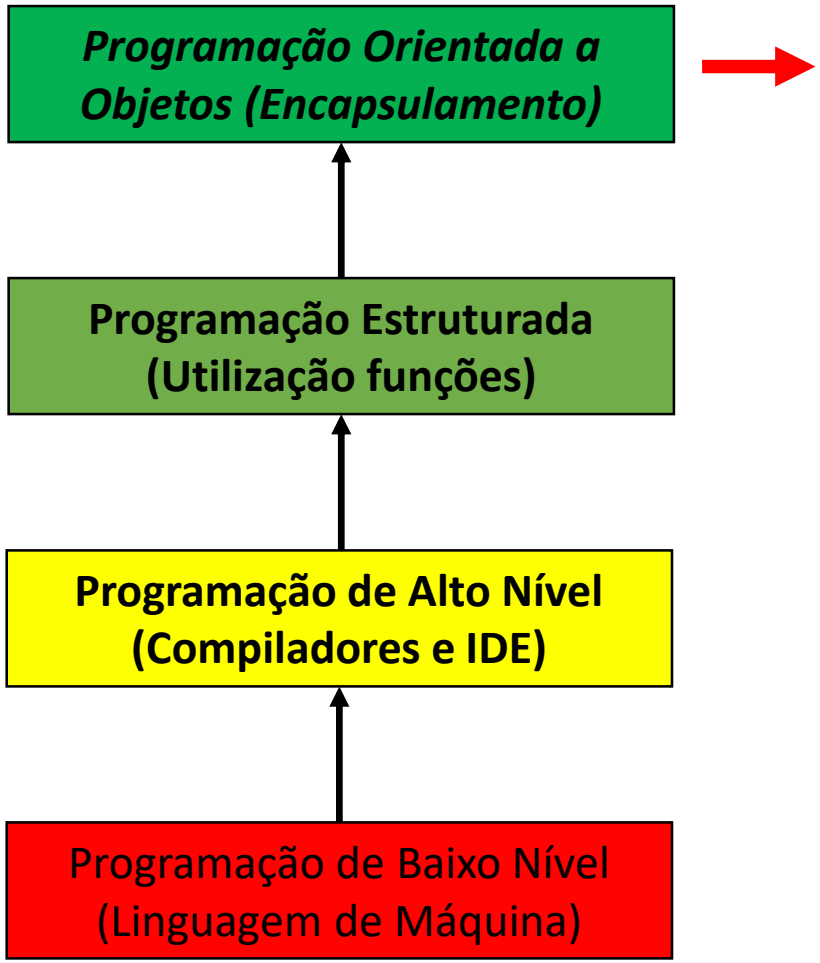
- Aproximar o mundo digital do mundo real.



- Modularização dos programas;
- Reuso do código;
- Pequenas estruturas processando dados.

Programação Orientada a Objeto - POO

- Aproximar o mundo digital do mundo real.

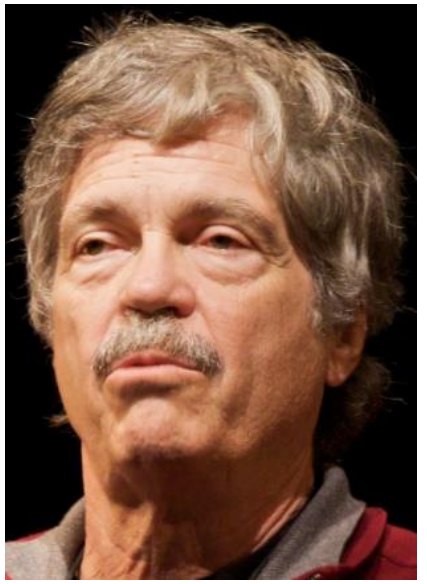


- Programas maiores;
- Desenvolvimento modular;
- Encapsulamento das rotinas, procedimentos, funções em programas maiores.

Programação Orientada a Objeto - POO

- Evolução da programação Estruturada:
  Orientação à Objetos.

"Computador ideal, deve funcionar como organismo vivo, onde uma célula se relaciona com outras, a fim de alcançar um objetivo, mas cada célula funciona de forma autônoma. As células poderiam se reagrupar para resolver um determinado problema, ou desempenhar outras funções".



Alan Kay,
EUA - 1940

Programação Orientada a Objeto - POO

- Ideia central: processar coisas menores... ***OBJETOS!***
- 2ª ideia central: funções processam um objeto em particular.
- 3ª ideia central: otimização de troca de informações entre funções.

- ***Exemplo:***

Programação não orientada.

Preocupação com os circuitos, componentes, funcionalidades e como estes itens estão inter-relacionados.

Programação orientada.

Preocupação com a funcionalidade dos botões; ações executadas, caso botões seja pressionados.

Programação Orientada a Objeto - POO

- ***Confiabilidade:*** *isolamento de pequenos trechos de programa não afeta o programa como um todo;*

- ***Oportuno:*** *programa pode ser dividido em várias partes, podendo ser desenvolvidas em paralelo.*

- ***Fácil Manutenção:*** *Facilidade para atualização do software, pequena modificação beneficia todos que utilizarem um módulo do programa.*

- ***Extensibilidade:*** *Permite a criação de novas funcionalidades.*

- ***Reutilização:*** *Mesmo código utilizado em diversos contextos;*

- ***Naturalidade:*** *Foco total nas funcionalidades do sistema, não detalhes de implementação.*

Programação Orientada a Objeto - POO

- ***Carro pode ser um objeto?***

Pode ser descrito por meio de características, comportamentos e estados!

CARACTERÍSTICAS	COMPORTAMENTO	ESTADO
Marca	Acelerar	Estacionado
Modelo	Desacelerar	Tanque cheio
Ano	Mover	Parado
Cor	Transportar	Estragado

Programação Orientada a Objeto - POO

- ***Posso ter vários automóveis com as mesmas características?***

Objetos seguem um mesmo formato, mesma classificação.

Todos objetos pertencem a uma mesma ***CLASSE****!*

Modelo para classificar um carro!

Programação Orientada a Objeto - POO

- Classe serve para realizar classificações de um objeto.
- Criação de um objeto, necessita da definição de uma classe.
- ***Toda Classe Responde a 3 perguntas:***

O que eu TENHO?	**Que coisas eu FAÇO?**	**Como estou AGORA?**
CARACTERÍSTICAS	**COMPORTAMENTO**	**ESTADO**

Programação Orientada a Objeto - POO

ATRIBUTOS

- Que coisas que um carro tem? Marca, modelo, cor, motor, ano.

MÉTODOS

- Que coisas um carro faz? Acelerar, transportar, estacionar, mover, parar.

ESTADO

- Como estou agora? Parado, estacionado, freado, desligado.

- ***Todo objeto vem de um MOLDE!***
- Criação de um objeto, necessita da definição de uma classe.

Programação Orientada a Objeto - POO

- ***Como criar uma classe?***

  - 1. Identificar os atributos da classe;

  - *2. Identificar os métodos da classe;*

  - *3. Identificar o estado atual da do objeto.*

```
Classe Automovel
marca: caractere
modelo: caractere
ano: inteiro
estacionamento: falso

 metodo acelerar()
      se (estacionamento) então
            escreva(“Carro estacionado!”)
      senao
            escreva(“Acelerando!”)
      fim se
 fim metodo acelerar()
 metodo estacionar()
       Estacionamento = verdadeiro
 fim metodo estacionar()

 fim Classe automovel
```

Programação Orientada a Objeto - POO

- ***Como criar Classe e instanciar (criar) um objeto?***

Classe principal programa_carro

```
Automovel carro1 = new Automovel
carro1.marca = “Toyota”
carro1.modelo = “Corolla”
carro1.cor = “Branco”
carro1.estacionamento= falso;
Carro1.estacionar()

Automovel carro2 = new Automovel
carro2.marca = “Jeep”
carro2.modelo = “Compass”
carro2.cor = “Amarelo”
carro2.estacionamento= true;
carro2.estacionar()
```

fim classe principal programa_carro

Classe Automovel

```
marca: caractere
modelo: caractere
ano: inteiro
estacionamento: falso

metodo acelerar()
    se (estacionamento) então
        escreva(“Carro estacionado!”)
    senão
        escreva(“Acelerando!”)
    fim se
fim metodo acelerar()
metodo estacionar()
    Estacionamento = verdadeiro
fim metodo estacionar()
```

fim Classe automovel

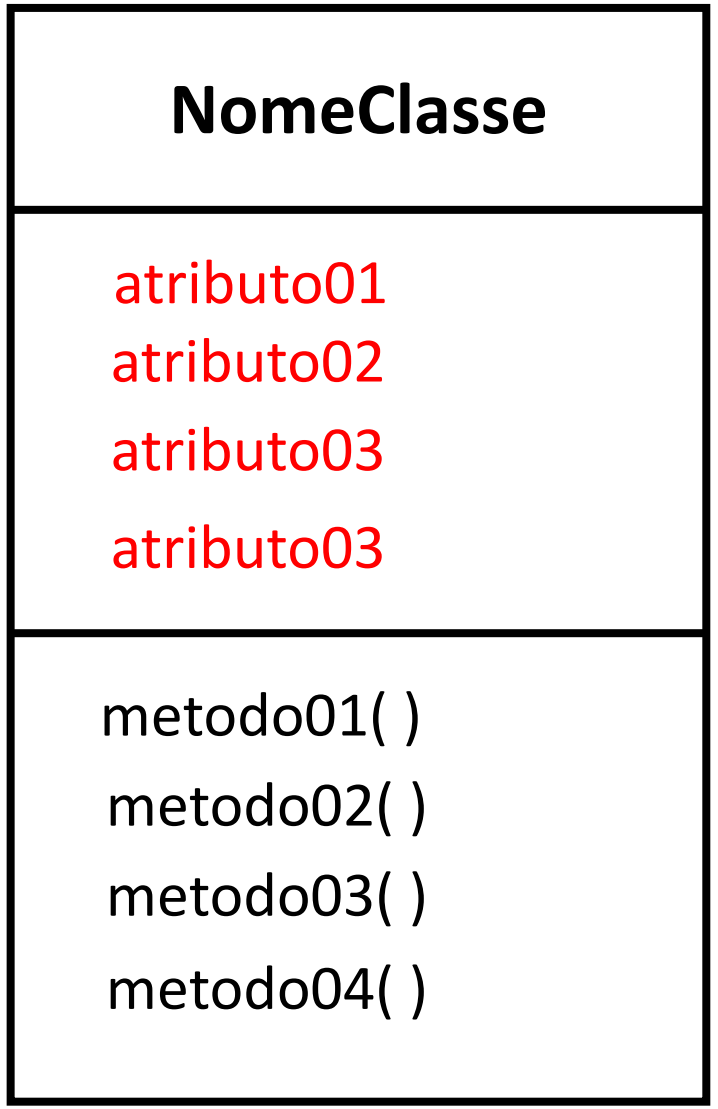
Diagrama de Classes

UML - Linguagem de Modelagem Unificada: forma de organizar a criação de uma classe.

Nome da classe (negrito) sempre devem começar com letra maiúscula.

Atributos da classe
(atributo não tem parênteses)

Métodos da classe
(parênteses depois do nome da classe)



Como identificar atributos, métodos e classes?

- Classes sempre devem começar com letra maiúscula
- Métodos e atributos começam com letra minúscula
- Método são identificados pelos parênteses após o nome do método.
- Atributo não tem parênteses.

Diagrama de Classes

Como identificar atributos, métodos e classes?

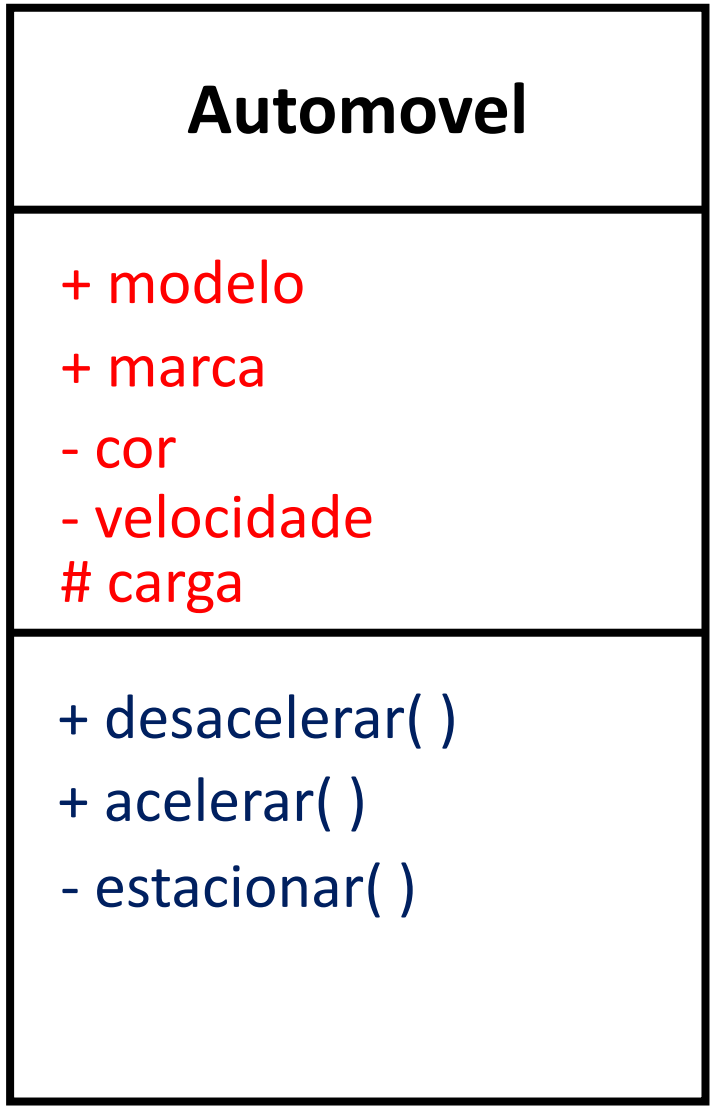
- Classes sempre devem começar com letra maiúscula
- Métodos e atributos começam com letra minúscula
- Método são identificados pelos parênteses após o nome do método.
- Atributo não tem parênteses.

Modificadores de Visibilidade

Classe

Atributos

Métodos



Métodos Getter, Setter e Construtor

- ***Métodos Assessores (métodos tipo getter)*** conseguem acessar um certo atributo, mantendo a segurança de acesso à ele.

- ***Métodos Modificadores (métodos tipo setter)*** modificam coisas que estão dentro do objeto - garantindo total segurança do atributo.

- ***Métodos Construtores:*** são utilizados para executar ações e inicializar objetos com definição de atributos, no instante em que o objeto é criado.

Método Construtor

Métodos Construtor é responsável pela criação de um objeto da classe que está sendo implementada; iniciando com valores e seus atributos ou realizando outras funções que possam vir a ser necessárias.’

Um métodos construtor deve possuir o mesmo nome da classe, (inclusive com correspondência de sintaxe) e não deve ter retorno; seguido de parênteses “()”.

Quando usamos a palavra-chave ***new*** estamos passando como um parâmetro, qual construtor deve ser executado para instanciar um objeto.’

Método Construtor

```java
public class Carro{

private String cor;
private double preco;
private String modelo;

/* CONSTRUTOR PADRÃO */
public Carro(){

}

/* CONSTRUTOR COM 3 PARÂMETROS */
public Carro(String cor, String modelo, double preco){
this.cor = cor;
this.modelo = modelo;
this.preco = preco;
}
}
```

Criação de um construtor padrão em Java.

Criação de um construtor que recebe parâmetros em Java.

Passagem de parâmetros; criação do objeto.

```java
public static void main(String[] args) {
//Construtor sem parâmetros
Honda hondaFitPreto = new Honda("2.0 Flex", "Honda Accord", "60000");
}
```

Encapsulamento

- ***Encapsulamento*** é umas das bases da Poo;
- Utilizado para proteger as informações sensíveis ou sigilosas, contidas nos códigos;
- Em Java é necessário controlar o acesso ao código, de modo que não fiquem suscetíveis à ações indevidas;

Encapsular significa: *evitar que os dados sofram acessos indevidos*.

Em POO, utilizamos os métodos ***getters*** e os ***setters*** – *estes métodos são conhecidos como métodos de acesso e modificação.*

São utilizados para proteger os dados, especialmente na criação das classes.

Métodos Getter e Setter

- ***Métodos Assessores (métodos tipo getter)*** conseguem acessar um certo atributo, mantendo a segurança de acesso à ele.
- ***Métodos Modificadores (métodos tipo setter)*** modificam coisas que estão dentro do objeto - garantindo total segurança do atributo.

Método *getter* retorna o valor do atributo.

O método *setter* recebe um parâmetro e o coloca no atributo.

```java
public class Vehicle {
  private String color;

  // Getter
  public String getColor() {
    return color;
  }

  // Setter
  public void setColor(String c) {
    this.color = c;
  }
}
```

```java
public static void main(String[] args) {
  Vehicle v1 = new Vehicle();
  v1.setColor("Vermelho");
  System.out.println(v1.getColor());
}

// O resultado é "Vermelho"
```

Quando os métodos getter e setter estiverem definidos, os utilizamos no main (programa principal).

Interfaces

- ***A interface é um recurso muito utilizado em Java, bem como na maioria das linguagens orientadas a objeto, para "obrigar" a um determinado grupo de classes a ter métodos ou propriedades em comum para existir em um determinado contexto,***

```java
public interface RadioComum {

    public abstract void play();
    public abstract void stop();

}
```

```java
public class RadioBolso implements RadioComum {

    @Override
    public void play() {

    }

    @Override
    public void stop() {

    }

}
```