Programação orientada a objetos em Java

Profª. Heloisa Moura

Programação orientada a objetos em Java

O que vamos ver:

- Pacotes e import;
- Introdução a OOP;
- Objetos
- Classes
- Criação de objetos
- Métodos

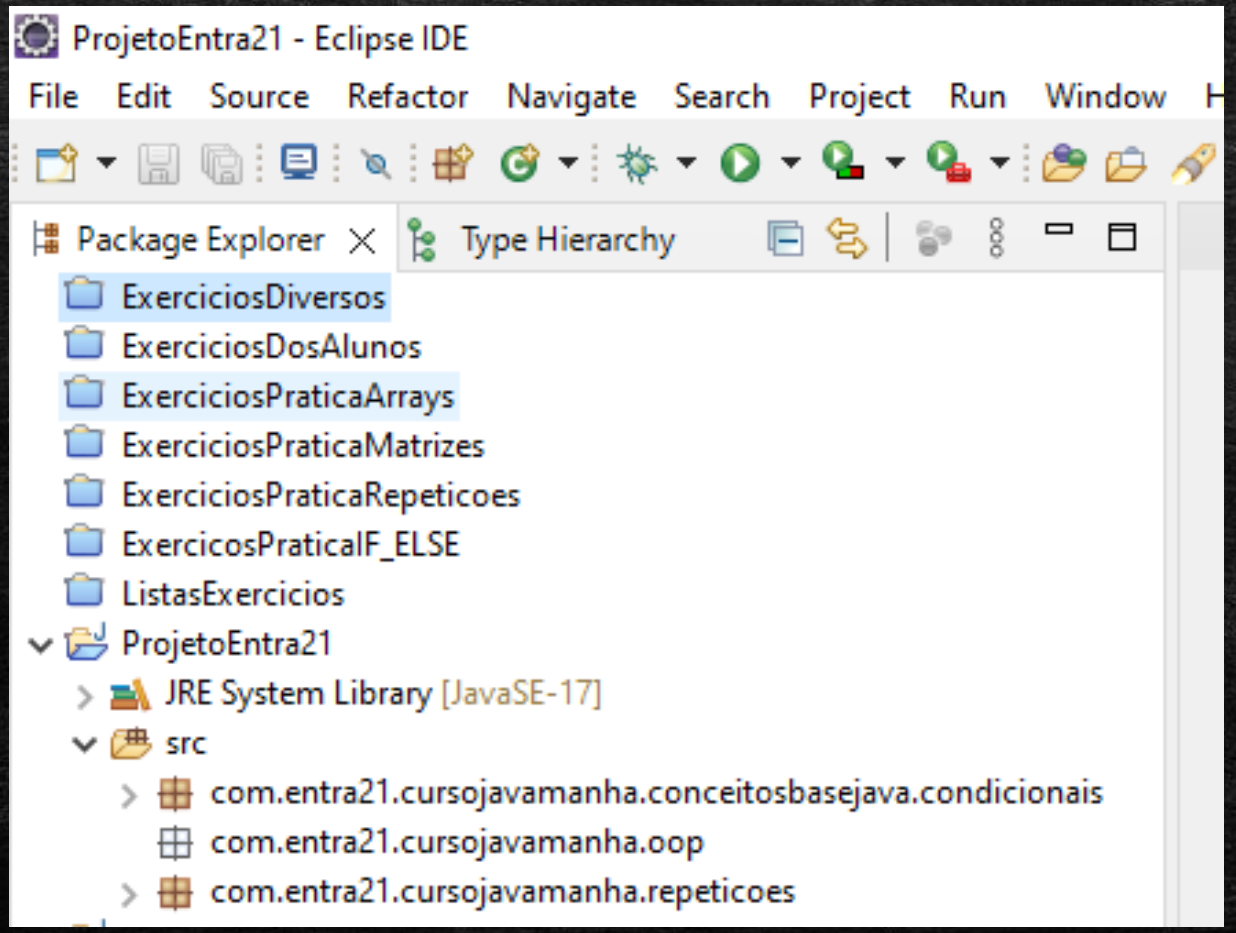
Programação orientada a objetos em Java
Pacotes e import

Pacotes - Convenção Java

- SEMPRE em lowercase(letra minúscula)
- Domínio da empresa ao contrário + Nome do projeto + (opcional)
- Pastas para organizar

Programação orientada a objetos em Java
Pacotes e import

Convenção Java



Programação orientada a objetos em Java
Pacotes e import

Import Java

- A instrução **import** do **Java** tem a finalidade de permitir em uma classe, de um determinado pacote, o acesso a demais classes que estejam em pacotes diferentes. Há duas formas de realizar a importação de uma classe usando a instrução **import** , a forma explícita e a forma implícita.
- Import java.net.*: Importa todas as classes do pacote java.net.
- Import java.net.URL: Importa apenas a classe URL do pacote java.net.

Programação orientada a objetos em Java
Pacotes e import

Import Java

Forma explicita: Import java.net.URL

Forma implícita: Import java.net.*

Conflitos de classes com o mesmo nome

Podemos ter problemas na importação de classes de pacotes diferentes, mas que tenham o mesmo nome. Por exemplo, se em um determinado código fosse necessário o uso das classes List do pacote java.awt e java.util, como o compilador saberia qual classe está sendo usada em determinado momento de sua declaração?

O compilador não vai conseguir identificar qual importação usar para a classe List e então irá gerar um erro de compilação do tipo:

Programação orientada a objetos em Java
Pacotes e import

Import Java

Forma explicita: Import java.net.URL

Forma implícita: Import java.net.*

O compilador não vai conseguir identificar qual importação usar para a classe List e então irá gerar um erro de compilação.

Conflitos de classes com o mesmo nome

Podemos ter problemas na importação de classes de pacotes diferentes, mas que tenham o mesmo nome. Por exemplo, se em um determinado código fosse necessário o uso das classes List do pacote java.awt e java.util, como o compilador saberia qual classe está sendo usada em determinado momento de sua declaração?

O compilador não vai conseguir identificar qual importação usar para a classe List e então irá gerar um erro de compilação do tipo:

Programação orientada a objetos em Java
Pacotes e import

Docs Java

https://docs.oracle.com/javase/8/docs/api/

Programação orientada a objetos em Java
Introdução

Orientação a Objetos

Programação Orientada a Objetos - POO (Object –Oriented Programming – OOP) é um paradigma de programação que usa tipos de dados personalizados.
Em vez de operar apenas com tipos de dados primitivos , podemos construir novos tipos de dados, conforme nossa necessidade .
Esses novos tipos de dados, que chamamos de **classes**, podem conter estruturas semelhantes a funções , denominadas **métodos,** e variáveis internas , chamadas de **atributos**.

Programação orientada a objetos em Java
Introdução

Principal objetivo da Programação orientada a objetos

Representar elementos do mundo real, para um objeto dentro do seu sistema. Para um computador entender o que é um objeto é necessário descrever as características(**atributos**) e também as ações (**métodos**) que podem serem executadas por esse objeto.

Programação orientada a objetos em Java
Introdução

Vantagens da Orientação a Objetos

- Fornece uma estrutura modular para a construção de programas;
- Objetos podem ser reutilizados em aplicações diferentes;
- O software se torna mais fácil de manter;
- O desenvolvimento é mais rápido, devido ao reuso de código;
- Encapsulamento: não é necessário conhecer a implementação interna de um objeto para poder usá-lo.

Programação orientada a objetos em Java
Introdução - Objetos

Objetos

Representam entidades do mundo real, tangível ou intangível. Entenda-se por tangível como algo que podemos manipular fisicamente, como o mouse que usamos por exemplo, e intangível como sendo algo que existe, mas não podemos manipular fisicamente, como é o caso da dívida contraída pelo João da Silva no dia 15/01/2022, por exemplo.
Um objeto possui características próprias (**atributos**) e executa determinadas ações(**métodos**). Sendo esses atributos e metódos provenientes da classe que o origina.

Programação orientada a objetos em Java
Introdução - Objetos

Objetos

Não confunda classe de objetos, com o objeto propriamente dito. Isso porque é comum encontrarmos exemplos de classes sendo dados como objeto, como, é o caso de chamar carro de objeto. Carro é uma classe de objetos, um objeto dessa classe seria carro vermelho de placa AAA, da marca AA, ano xxxx e modelo mmmm.

Programação orientada a objetos em Java
Introdução - Objetos

Objetos

Alguns exemplos de objetos.

Carro do João
Marca: BMW
Cor: Amarela
Ano: 00
Modelo: 00

Cachorro da Maria
Nome: Rex
Cor: Marrom
Idade (anos): 02
Raça: Vira-lata

Programação orientada a objetos em Java
Introdução - Objetos

Alguns exemplos de objetos.

Bola do Manoel
Marca: PB
Cor: Preta e branca
Uso: Futebol

Notebook da Marta
Marca: NONONO
Processador: AAA
Tamanho da tela: 14”
Peso: 3,5 KG

Máquina da Cláudia
Marca: Nikon
Cor: Preta
Tipo: Digital
Tela visor: 4,5”

Programação orientada a objetos em Java
Introdução - Objetos

Objetos

As figuras acima, mostra que todas as entidades são casos particulares, ou seja, tem características próprias. Quando nos referimos a objetos, não estamos nos referindo ao gênero e sim a casos particulares. Por exemplo, o carro do João é um exemplar de um tipo genérico carro, por isso ele é considerado um objeto.

Programação orientada a objetos em Java
Definição de Classe

Conceito fundamental: Classe

É uma representação de objetos, que são descritos com as mesmas características (atributos) e as mesmas operações. Representa uma ideia ou conceito do mundo externo e agrupa objetos que tenham propriedades similares.

As classes são os blocos de construção mais importantes dos sistemas orientados a objetos, e devem possuir responsabilidades bem definidas na aplicação.

Programação orientada a objetos em Java
Definição de Classe

Classe

É um tipo personalizado e abstrato de dados, “molde” para a criação de objetos. Cada classe criada se torna um novo tipo disponível para declarar atributos, métodos e criar objetos

Por convenção, começamos o nome de uma classe com a primeira letra maiúscula. Caso o nome tenha mais de uma palavra, todas deverá ter a primeira letra maiúsculas e as palavras estarem juntas.

As classes são codificadas em arquivos com a extensão .java, com o mesmo nome da classe. Usamos as classes para criarmos objetos.

Programação orientada a objetos em Java
Definição de Classe

Em resumo - Classe

O que temos que ter em mente é que uma classe, por si só não é "consumível", ou seja, não posso usar uma classe, mas sim um exemplar desta. Veja uma classe como uma **especificação** do que determinados objetos devem ter, dessa forma, eu não posso consumir essa especificação e sim um objeto criado de acordo com esta. Como, por exemplo, você não pode comer uma receita de bolo (classe), mas pode comer um bolo **feito** (objeto) com as especificações constantes na receita. Ainda, você não pode morar em uma planta de uma casa (classe), mas pode morar em uma casa **construída** (objeto) de acordo com as especificações constantes nessa (planta da casa). Note, no parágrafo anterior, o destaque para **feito** e **construída**, isso significa dizer que a partir de uma classe eu posso construir (fazer) objetos, quantos eu queira.

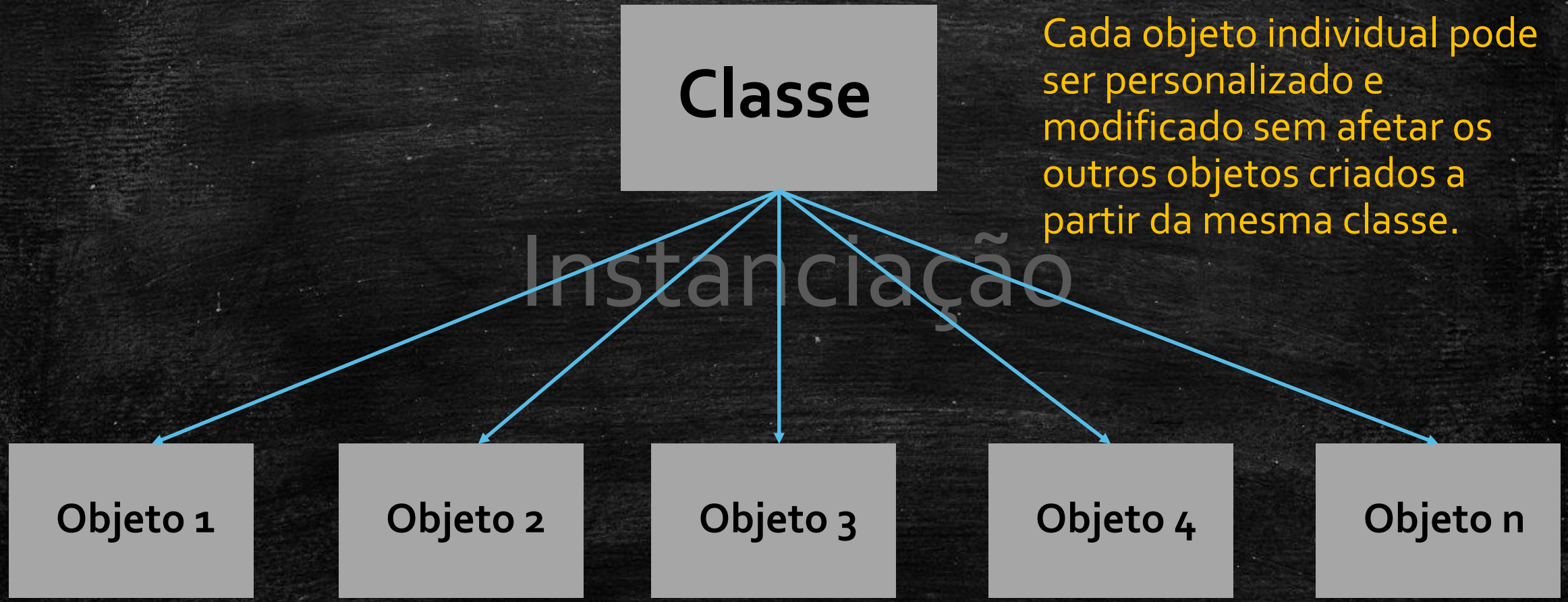
Programação orientada a objetos em Java
Definição de Classe

Forma geral de uma Classe

Boas praticas de programação para construção de classes

```
1
2   class NomeDaClasse {
3
4       //variáveis de instância - atributos
5       int var1;
6       int var2;
7       int var3;
8
9
10      // declaracao de metodos
11      void metodo1 (int paramentro) {
12          // corpo do metodo
13      }
14
15      public static void metodo2 (int paramentro) {
16          // corpo do metodo
17      }
18
19      void metodo3 (int paramentro) {
20          // corpo do metodo
21      }
22
23  }
```

Programação orientada a objetos em Java
Classe - Objeto



Cada objeto individual pode ser personalizado e modificado sem afetar os outros objetos criados a partir da mesma classe.

Programação orientada a objetos em Java

Atributos, Operações e Métodos

Um objeto tem características e comportamentos. Em se tratando da orientação objetos, dizemos que essas características são **atributos** e os comportamentos são as **operações**.

- Tendo em mente que uma classe é uma especificação do que um objeto deve ter para pertencer a essa (classe), podemos dizer que a classe define que atributos os objetos pertencentes a ela deve ter e como ele deve se comportar.
- Antes de detalhar mais o que é atributo e o que é operação, é importante entender que operação e métodos não são as mesmas coisas. É comum confundir os dois termos.

Programação orientada a objetos em Java
Atributos, Operações e Métodos

Atributos

São características dos objetos (não da classe, pois a classe especifica as características que os objetos devem ter), como por exemplo:

- A máquina da Cláudia tem a marca Nikon, cor preta e tipo digital. Pense assim, a classe máquina fotográfica especificou que um objeto pertencente a ela deveria ter esses atributos, por isso esse exemplar (máquina da Cláudia) possui tais características.

Programação orientada a objetos em Java
Atributos, Operações e Métodos

Atributos

Há dois tipos principais de atributos:

- Estáticos - Mantém o mesmo valor durante toda a sua existência.
  Exemplo: data de nascimento de uma pessoa. Mantem o mesmo valor durante toda a existência do objeto instanciado.

- Dinâmicos - Podem ter valores que variam com o passar do tempo.
  Exemplo: idade de uma pessoa. Varia de ano em ano.

Programação orientada a objetos em Java
Criando Classe

Representar uma entidade do mundo real: Um Carro

- Cor
- Marca
- Modelo
- Número de passageiros
- Capacidade do tanque de combustível
- Consumo de combustível por km

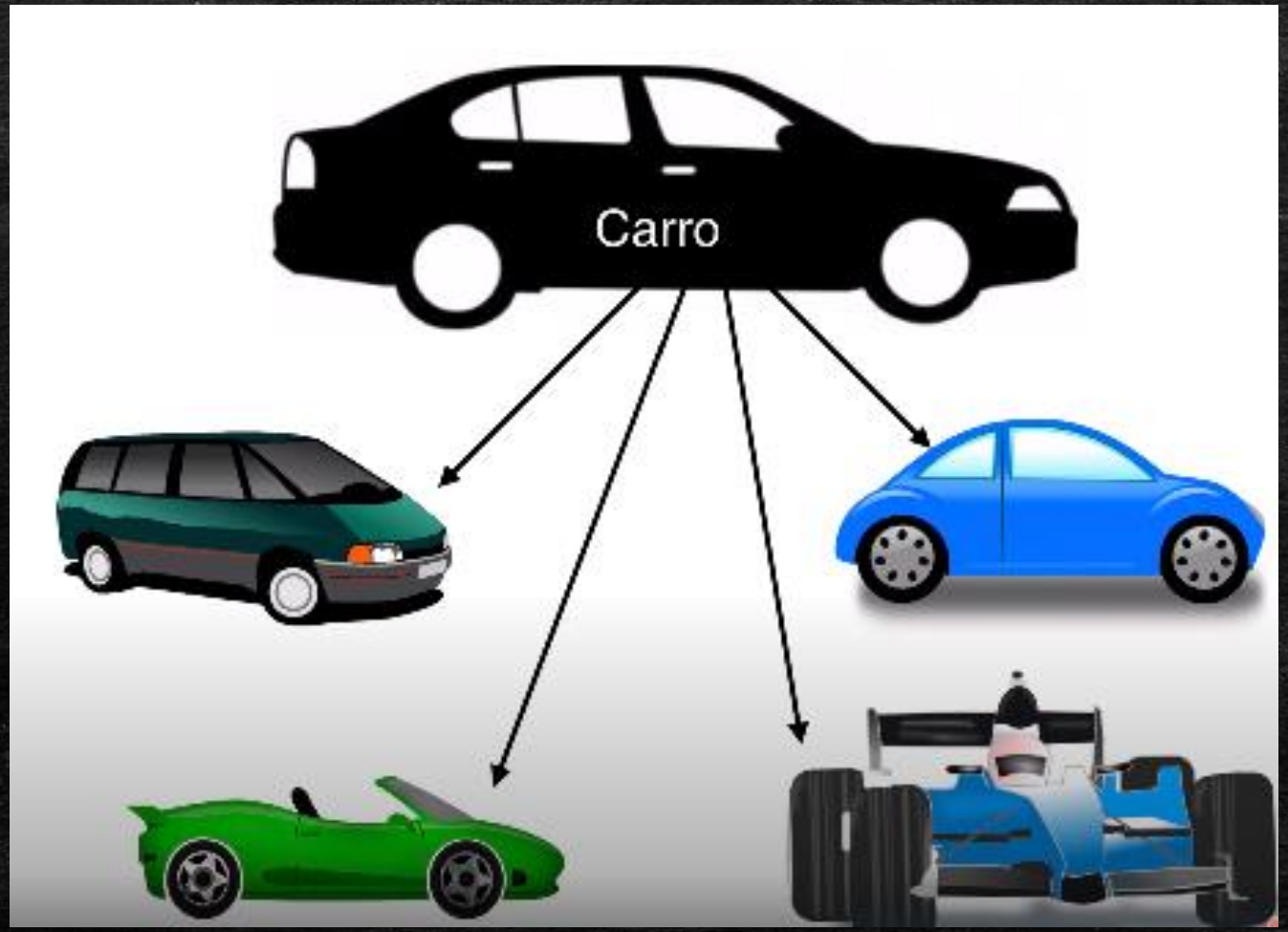
Programação orientada a objetos em Java
Criando Classe

Definição da classe Carro e seus atributos

Traduzindo as informações (características) da entidade Carro para código Java.

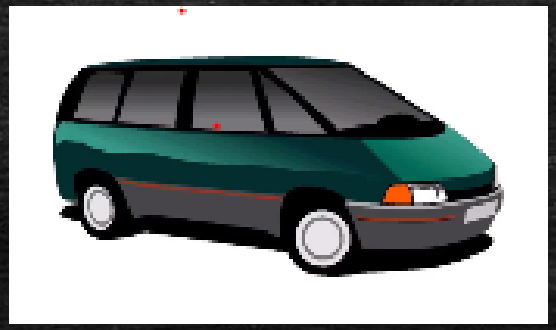
```
class Carro{

  String marca;
  String modelo;
  int numPassageiros;  // numero de passageiros
  double capCpmbustivel; // capacidade do tanque de combustível
  double consumoCombustivel; // consumo de combustível por km

}
```

Programação orientada a objetos em Java
Criando Classe



Programação orientada a objetos em Java
Criando Objetos

Criação de objetos



```
Carro van = new Carro();
van.marca = "Fiat";
van.modelo = "Ducato";
 van.numeroPassageiros = 10;
 van.capCpmbustivel = 100;
 van.consumoCombustivel= 0.15;
```

Instanciação da Classe Carro. Cria o objeto Carro van.

Programação orientada a objetos em Java
Criando Objetos

Criação de objetos

Instanciação da Classe Carro. Cria o objeto Carro fusca.

```
Carro fusca = new Carro();
 fusca.marca = "Volkswagen";
 fusca.modelo = "Fusca";
 fusca.numeroPassageiros = 4;
 fusca.capCombustivel = 30;
 fusca.consumoCombustivel = 0.2;
```

Programação orientada a objetos em Java
Criando Objetos

Objetos são acessados por referências

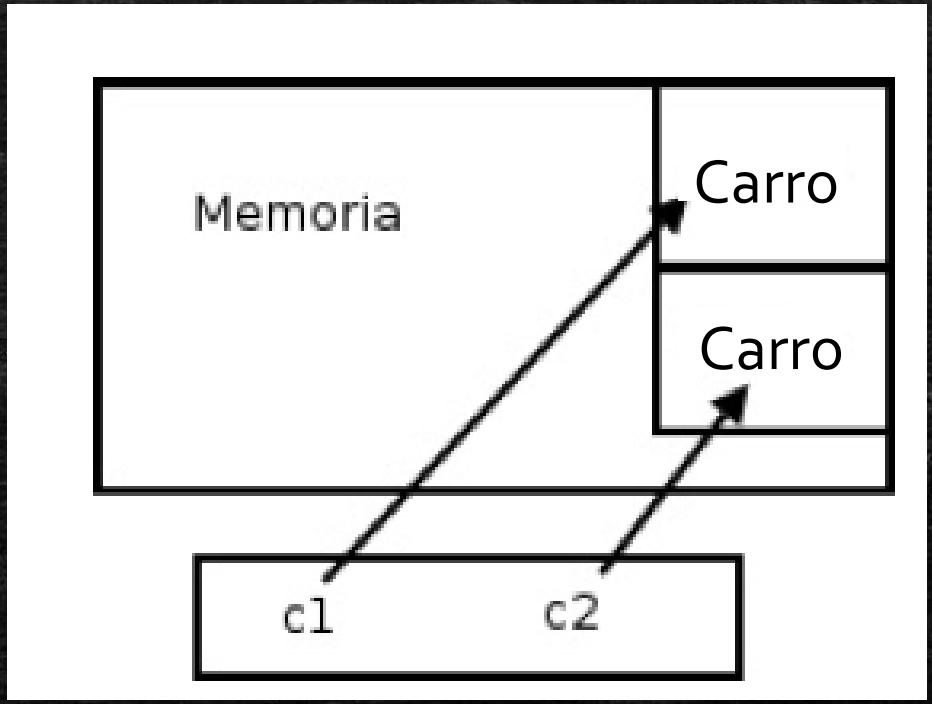
Quando declaramos um atributo para associar a um objeto, na verdade, esse atributo não guarda o objeto, mas sim, uma maneira de acessá-lo, chamada de referência. É por esse motivo que, diferente dos tipos primitivos precisamos dar new depois de declarado o atributo:

```
public static void main(String[] args) {
        Carro c1;
        c1 = new Carro();
        Carro c2;
        c2 = new Carro();
}
```

Programação orientada a objetos em Java
Criando Objetos

O correto aqui é dizer que c1 se refere a um objeto. Não é certo dizer que c1 é um objeto, pois c1 é um atributo referência. O correto é "tenho uma referência c1 a um objeto do tipo Carro“.

ilustração do código anterior



Programação orientada a objetos em Java
Criando Objetos

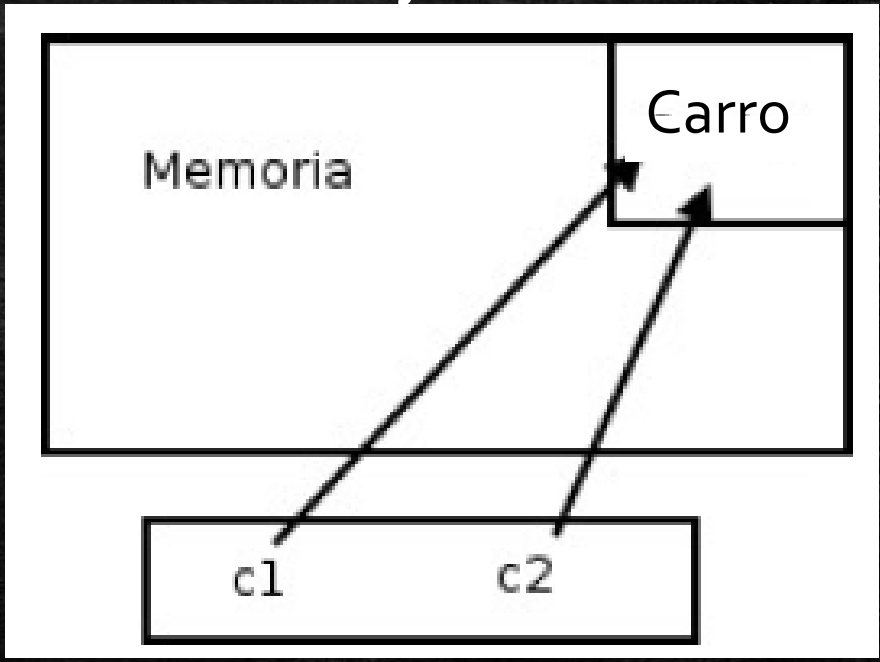
Internamente, c1 e c2 vão guardar um número que identifica em que posição da memória aquele Carro se encontra. Dessa maneira, ao utilizarmos o "." para navegar, o Java acessará o Carro que se encontra naquela posição de memória, e não uma outra.
Um outro exemplo:

```
class TestaReferencias {
        public static void main(String[] args) {
                Carro c1 = new Carro();
                c1.modelo = “fusca”;
                Carro c2 = c1; // linha importante!
                c2.modelo = “Ducato”;
                System.out.println(c1.modelo);
                System.out.println(c2.modelo);
        }
}
```

Programação orientada a objetos em Java
Criando Objetos

Qual é o resultado do código acima? O que aparece ao rodar?
O que acontece aqui? O operador = copia o valor de um atributo. Mas qual é o valor do atributo c1 ? É o objeto? Não. Na verdade, o valor guardado é a referência (endereço) ao local onde o objeto se encontra na memória principal.
Na memória, o que acontece nesse caso:

```
Carro c1 = new Carro();
Carro c2 = c1;
```



Programação orientada a objetos em Java
Criando Objetos

Quando fizemos c2 = c1 , c2 passa, nesse instante, a fazer referência ao mesmo objeto referenciado por c1 . Então, nesse código em específico, quando utilizamos c1 ou c2 , estamos nos referindo exatamente ao mesmo objeto! Elas são duas referências distintas, porém apontam para o mesmo objeto. Compará-las com " == " irá nos retornar true , pois o valor que elas carregam é o mesmo!

Outra forma de perceber isso é que demos apenas um new, logo só pode haver um objeto Carro na memória.

Programação orientada a objetos em Java
Criando Objetos

Outra situação:

```
public static void main(String[] args) {
        Carro c1 = new Carro();
        c1.marca = “Fiat";
        c1.modelo= “Ducato”;
        Carro c2 = new Carro();
        c2.marca = “Fiat";
        c2.modelo = “Ducato”;
        if (c1 == c2) {
                System.out.println(“Carros iguais");
        }
}
```

Programação orientada a objetos em Java
Criando Objetos

O operador == compara o conteúdo dos atributos, mas esses atributos não guardam o objeto, e sim o endereço em que ele se encontra. Como em cada um desses atributos guardamos carros criados diferentemente, eles estão em espaços distintos da memória, o que faz o teste if valer false . Os Carros podem ser equivalentes no nosso critério de igualdade, porém eles não são o mesmo objeto. Quando se trata de objetos, pode ficar mais fácil pensar que o == compara se os objetos (referências, na verdade) são o mesmo,. e não se são iguais

Programação orientada a objetos em Java
Criando Objetos

Para saber se dois objetos têm o mesmo conteúdo, você precisa comparar atributo por atributo.

Programação orientada a objetos em Java
Criando Objetos

Exemplo na pratica
e
Exercícios

Programação orientada a objetos em Java
Operações e Métodos

Operações

- As operações definem o que os objetos devem fazer, como por exemplo, andar, mover, imprimir. Note que neste momento estamos apenas dizendo o que o objeto tem que fazer. Seria como dizer a um veículo que ele precisa se mover e como ele irá fazer isso. A operação não.
  Em resumo, uma operação diz o que o objeto deve fazer, porém, não diz como ele deve fazer isso.

Programação orientada a objetos em Java
Operações e Métodos

Métodos

Os métodos são as formas como os objetos pertencentes a determinadas classes irão executar determinada operação. Como, por exemplo, uma operação calcular imposto (note que as operações são verbos de ação) tem como método cálculos baseados na tabela de alíquota da receita federal. Dessa forma, um método diz como um objeto deve executar a operação definida pela classe.

Programação orientada a objetos em Java
Operações e Métodos

Métodos – Outra definição

Sequencia de declarações (comandos) que possui um nome de identificação (geralmente começa com verbos).

Um método possui um corpo onde ficam os comandos que serão executados quando o método for chamado, e também podem receber dados para processamento (parâmetros) e retornar ou não informações.

Métodos são as “coisas que a classe podem fazer”

O nome de um método deve iniciar com letra minúscula e usar *Camel case.

CamelCase é a denominação em inglês para a prática de escrever as palavras compostas ou frases, onde cada palavra é iniciada com maiúsculas ou minúsculas e unidas sem espaços

Programação orientada a objetos em Java

Operações e Métodos

Método – Simples(sem retorno e/ou parâmetro)

É a maneira mais simples que a gente tem para declarar um método.
Começamos a declaração de um método pelo seu tipo de retorno.
A palavra reservada **void**, indica que o método não terá retorno de valores após ser processado.
A seguir vem o nome do método, que geralmente começa com um verbo(boas praticas de programação), seguido ou não de outra palavra que informe mais o que o método faz. Após, abrimos e fechamos parênteses, que é onde indicamos os parâmetros de entrada do método, caso tenha.

Programação orientada a objetos em Java
Operações e Métodos

Método – Simples(sem retorno e parâmetro)

Exemplo:

```
void exibirAutonomia(){
 System.out.println(“ A autonomia do carro é: “  +capCombustivel *
consumoCombustivel + “ km “ );
}
```

Programação orientada a objetos em Java

Operações e Métodos

Exemplo na pratica

Programação orientada a objetos em Java

Operações e Métodos

Método – Com retorno e sem parâmetro

Método que possui o tipo de retorno na sua declaração e a palavra reservada return, retornando no final do método o valor do tipo declarado, depois da execução do método a quem chamou.
O que é muito legal na declaração de métodos dentro de classes no Java é que dentro dos métodos nós temos acesso a informações daquela classe. Nós podemos chamar outros métodos e podemos também utilizar os atributos da classe.
No exemplo abaixo utilizamos os atributos da classe Carro dentro do método.
Exemplo:

```
double obterAutonomia(){
 return(capCombustivel * consumoCombustivel  );
}
```

O tipo de retorno pode ser tipos primitivos ou objetos.

Programação orientada a objetos em Java
Operações e Métodos

Exemplo na pratica

Programação orientada a objetos em Java
Operações e Métodos

Método – com retorno e parâmetro

Método que possui o tipo de retorno e os parâmetros ou lista de parâmetros na sua declaração. Utiliza os parâmetros declarados para executar operações dentro do método, retornando no final do método o valor do tipo declarado, a quem chamou.
Quando é que vou passar um parâmetro para o método?
Quando você precisar de uma informação que não está disponível na classe

Exemplo:

```
double calcularCombustivel(double km){
 return km/consumoCombustivel ;
}
```

Programação orientada a objetos em Java
Operações e Métodos

No exemplo anterior para calcularmos o combustível necessário para uma determinada quilometragem, precisamos da informação de km e não temos essa informação na classe. Temos o consumoCombustivel. Por isso a necessidade de passamos para o método calcularCombustivel o parâmetro km.

Programação orientada a objetos em Java

Operações e Métodos

Exemplo na pratica

Exercícios