

TI-220 Java Orientado a Objetos

ANTONIO CARVALHO - TREINAMENTOS

Disciplina

TI-220 Linguagem de Programação Java Orientado a Objetos

Objetivo : Ensinar conceitos de programação orientada a objetos, assim como princípios e alguns padrões de projetos

Carga horária : 20 horas

Professor

Antonio Rodrigues Carvalho Neto
antoniorcn@hotmail.com

Ao enviar emails favor colocar no cabeçalho:
<TURMA>-<MATRICULA>-<NOME>-<Assunto>

Ementa

Orientação a Objetos - Introdução

- Classes
- Objetos
- Métodos
- Construtores

Estrutura da Memória

Orientação a Objetos - Dependências

- Associação
- Composição
- Herança
- Sobrecarga

Orientação a Objetos – Polimorfismo

- Sobrescrita
- Interfaces

Modificadores de Acesso

Recursos Utilizados

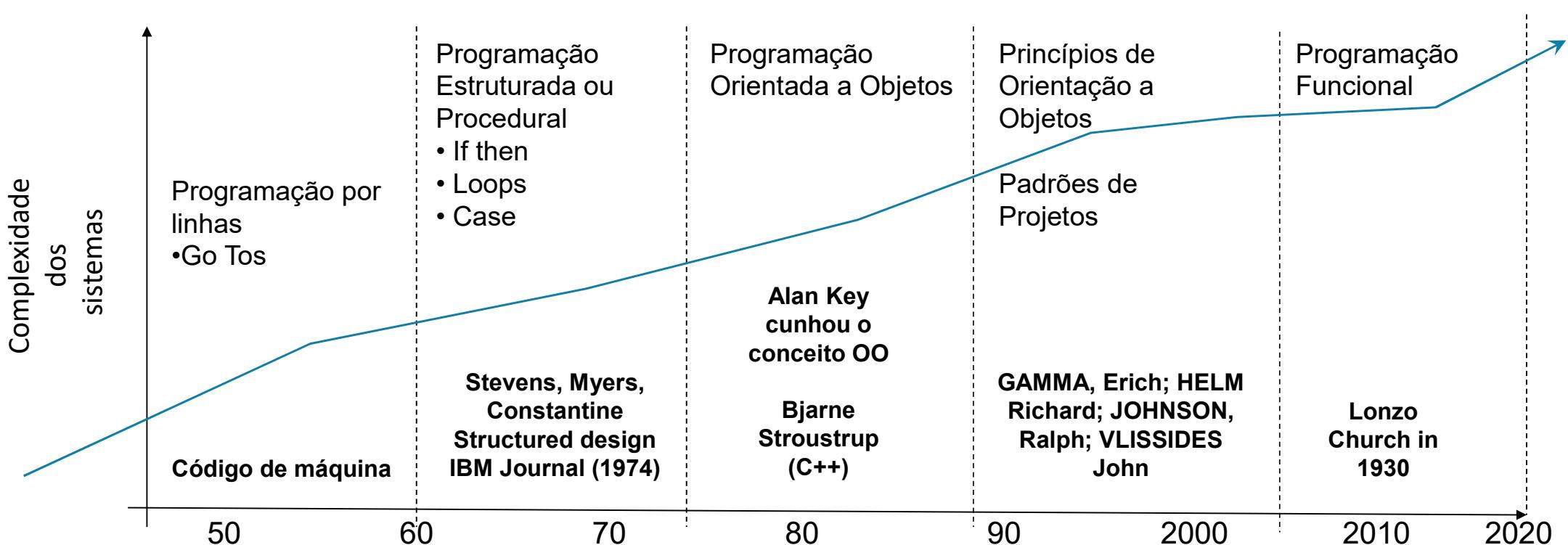
Para baixar o **Eclipse** acesse o site --> www.eclipse.org

Para baixar o **Java** acesse o site -->
<http://www.oracle.com/technetwork/java/javase/downloads/java-se-jdk-7-download-432154.html> procure a versão 7 do JDK

Java Orientação a Objetos

Introdução

Evolução dos paradigmas de programação



O que é Orientação a Objetos

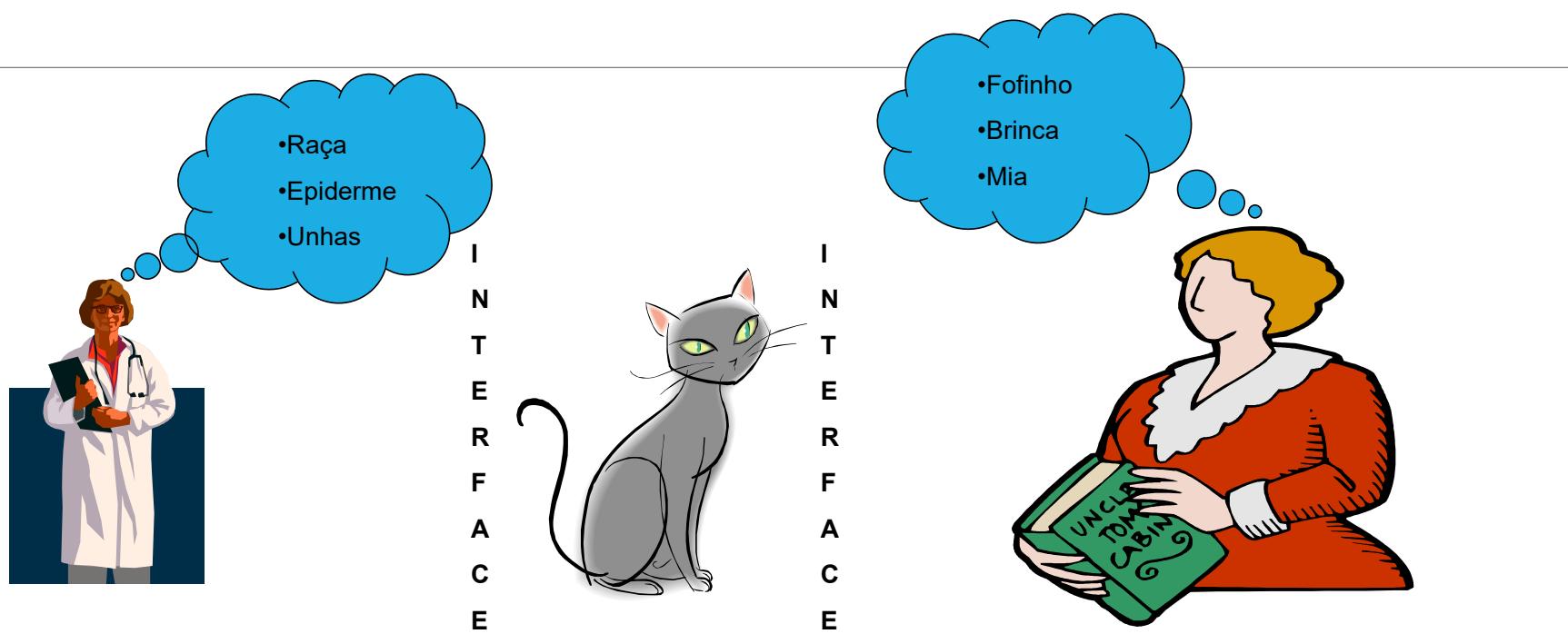
A orientação a Objetos visa abstrair características de entidades concretas e abstratas do mundo real, criando padrões ou classificações que posteriormente serão transformados em objetos virtuais.

O que é Orientação a Objetos

Todo objeto é composto por :

- Características, também conhecidas como atributos ou estado.
As características de um objeto modificam seu estado.
- Comportamentos, também conhecidos como métodos, funções, ações. Os comportamentos são ações que o objeto pode executar.

O que é Abstração



Nota : Duas ou mais pessoas podem ver características distintas em um mesmo objeto. O objeto comporta ambas as características porém cada uma das pessoas visualiza aquilo que mais atendem a sua realidade, mediante aos seus próprios filtros internos. Isso chama-se abstração.

O que é Abstração

“Uma abstração é qualquer modelo que inclui os aspectos mais importantes, essenciais de alguma coisa, ao mesmo tempo em que ignora os detalhes menos importantes” (EDUARDO BEZERRA)

Através das abstrações podemos concentrar nosso foco, naquilo que realmente importa, ignorando as demais características e comportamentos.

Objetos

São elementos do universo, sejam físicos ou abstratos, reais ou imaginários. São únicos e guardam seu estado, mesmo que possam possuir estados idênticos ainda sim são únicos.

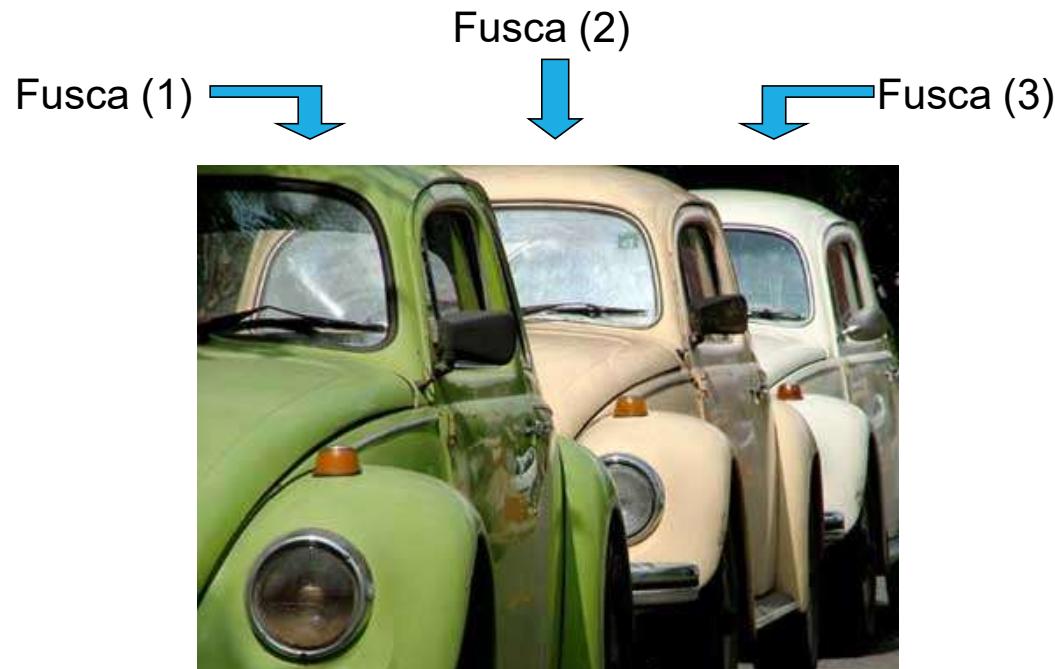
Os objetos podem ser representados por classes, que visam abstrair suas características e comportamentos do mundo real.

Embora únicos os objetos da mesma classe possuem os mesmos tipos de estados e comportamentos.

Nota: o resultado do comportamento pode ser diferente em cada objeto, devido ao seu estado.

Objetos

Objeto é um elemento concreto de um tipo de classe.



Classe

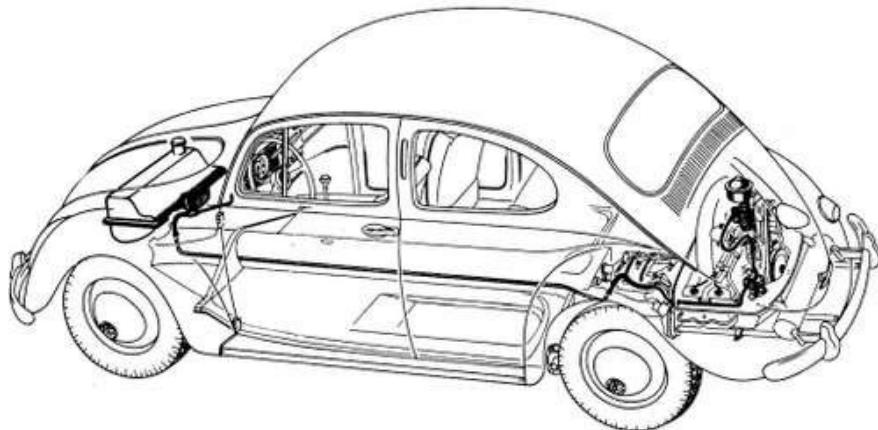
Classe é uma especificação dos tipos de estado e comportamentos dos objetos suportados. (SIERRA, 2008 – Tradução adaptada pelo Autor)

A classe é uma abstração do objeto (seja físico ou abstrato), através dela é possível descrever o objeto sob a ótica das características e comportamentos.

Classe

Através da classe é possível descrever os atributos e comportamentos dos objetos.

SISTEMA DE ALIMENTAÇÃO



Classe - Sintaxe

A sintaxe para criar uma classe é :

```
<modificador> class <nome da classe> {  
    <modificadores> <tipo> <nome do atributo1>;  
    <modificadores> <tipo> <nome do atributo2>;  
    ..  
    <modificadores> <tipo> <nome do atributoN>;  
  
    <modificadores> <tipo de retorno> <nome do método 1> ( <parâmetros> ) {  
        <Código a ser executado linha 1>;  
        <Código a ser executado linha 2>;  
        ..  
    }  
    <modificadores> <tipo de retorno> <nome do método N> ( <parâmetros> ) {  
        <Código a ser executado linha 1>;  
        <Código a ser executado linha 2>;  
        ..  
    }  
}
```

Classe - Exemplo

Gato

+tamanho : float
+peso : float

+brincar() : void
+miar() : void

```
public class Gato {  
  
    public float tamanho;  
  
    public float peso;  
  
    public void brincar() {  
        System.out.println("Estou brincando");  
    }  
  
    public void miar() {  
        System.out.println("miau ...");  
    }  
}
```

Classe - Exercício

A partir de objetos **comuns**, crie 3 classes, abstraindo ao menos 5 características e 3 comportamentos para cada uma.

Crie o código em Java para estas classes

Classe - Instanciando

- Instanciar uma classe é o ato de criar um objeto a partir dela.
- A classe contém os **tipos** das características e os comportamentos que os objetos vão possuir.
- Mas cada objeto possuirá **conteúdos** diferentes em suas características, e o comportamento de cada um irá variar conforme estes conteúdos.

Classe - Instanciando

- A sintaxe para instanciar uma classe é:

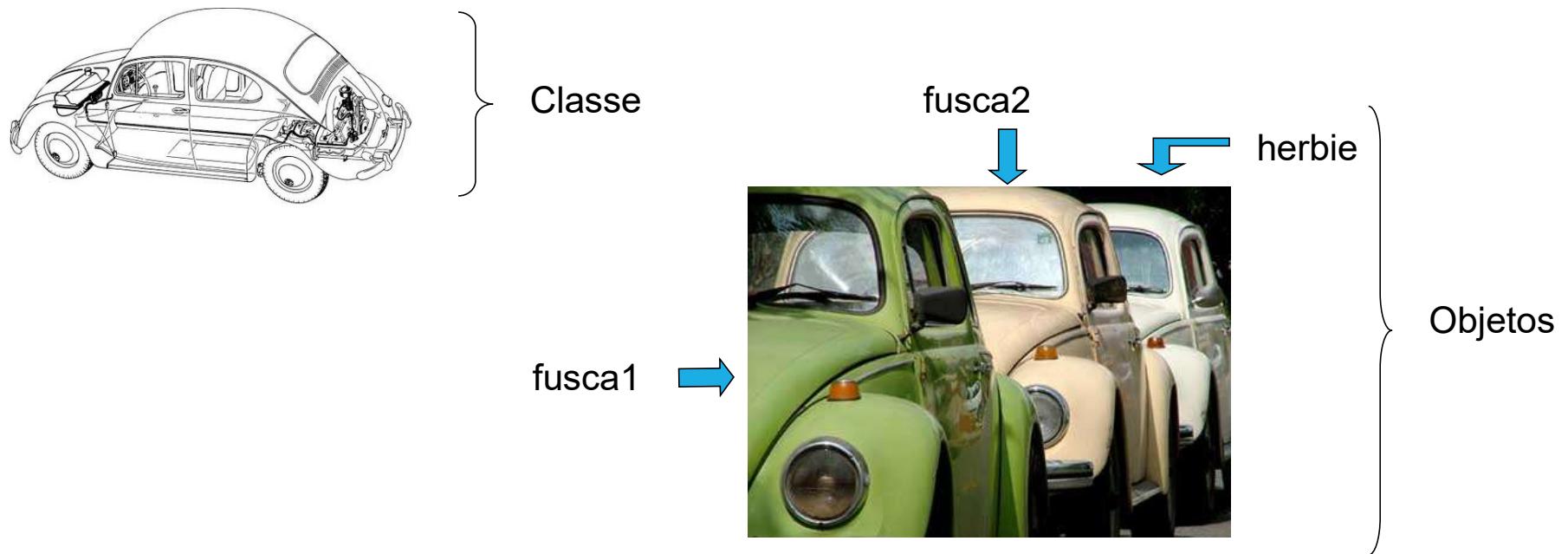
```
<tipo da variável> <nome da variável> = new <nome da classe>(<parâmetros>);
```

- Exemplo:

// Para instanciar um objeto do tipo Gato usa-se :

```
Gato felix = new Gato();
```

Classe - Instanciando



A partir de uma mesma classe é possível criar diversos objetos.

Classe - Instanciando

A classe carro mostrada na página anterior é apenas a planta (o desenho), mostrando como os objetos instanciados serão e como se comportarão:

```
public class Carro {  
    float autonomia;  
    int maxKmHora;  
    String marca;  
    String modelo;  
    int ano;  
    int velocidade;  
  
    public void acelerar() {  
        System.out.println(" Acelerando ... ");  
        velocidade = velocidade + 10;  
    }  
  
    public void frear() {  
        System.out.println(" Freando ... ");  
        velocidade = velocidade - 10;  
    }  
}
```

Para se criar os objetos a partir da classe é preciso instanciá-los, conforme no exemplo abaixo:

```
// Para instanciar um objeto do tipo Carro usaremos :  
Carro fuscal = new Carro ();  
Carro fusca2 = new Carro ();  
Carro herbie = new Carro ();
```

Objetos Instanciados

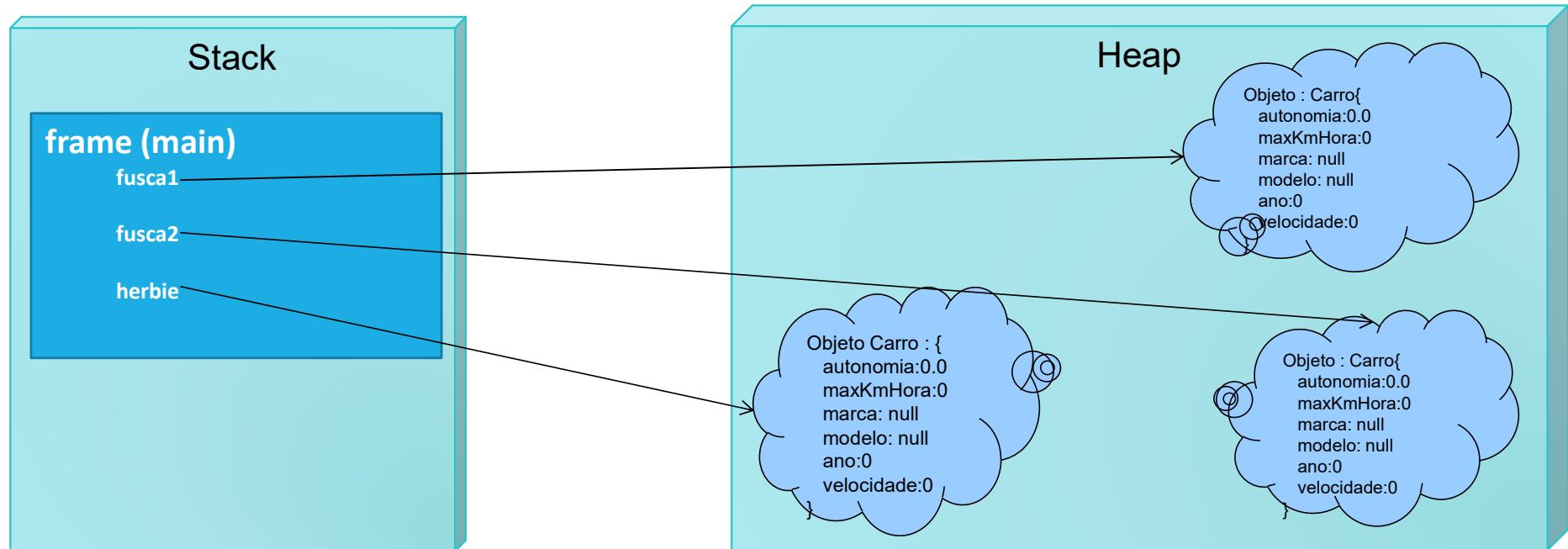
A partir do momento em que a classe for instanciada, o Java criará mais um objeto na memória, que poderá conter **informações** diferentes em suas **características** (atributos).

Para alterar o conteúdo de uma **característica** do objeto é preciso referenciar seu atributo e atribuir uma nova informação.

Lembrem-se atributos e características são a mesma coisa.

Objetos Instanciados

Objetos criados na memória a partir do momento em que são instanciados



Objetos Instanciados

A sintaxe para atribuir uma informação para uma característica do objeto é :

<nome do objeto>.<nome do atributo> = <novo valor>;

Exemplo:

```
herbie.autonomia = 398.5f;
```

Para utilizar a característica de um objeto basta mencioná-la conforme abaixo :

<nome do objeto>.<nome do atributo>

Exemplo:

```
System.out.println( herbie.autonomia );
```

Objetos Instanciados

Quando o código abaixo for executado, os objetos instanciados na memória sofrerão alteração no estado.

As variáveis de instância que não forem inicializadas, receberão valores padrões conforme o tipo:

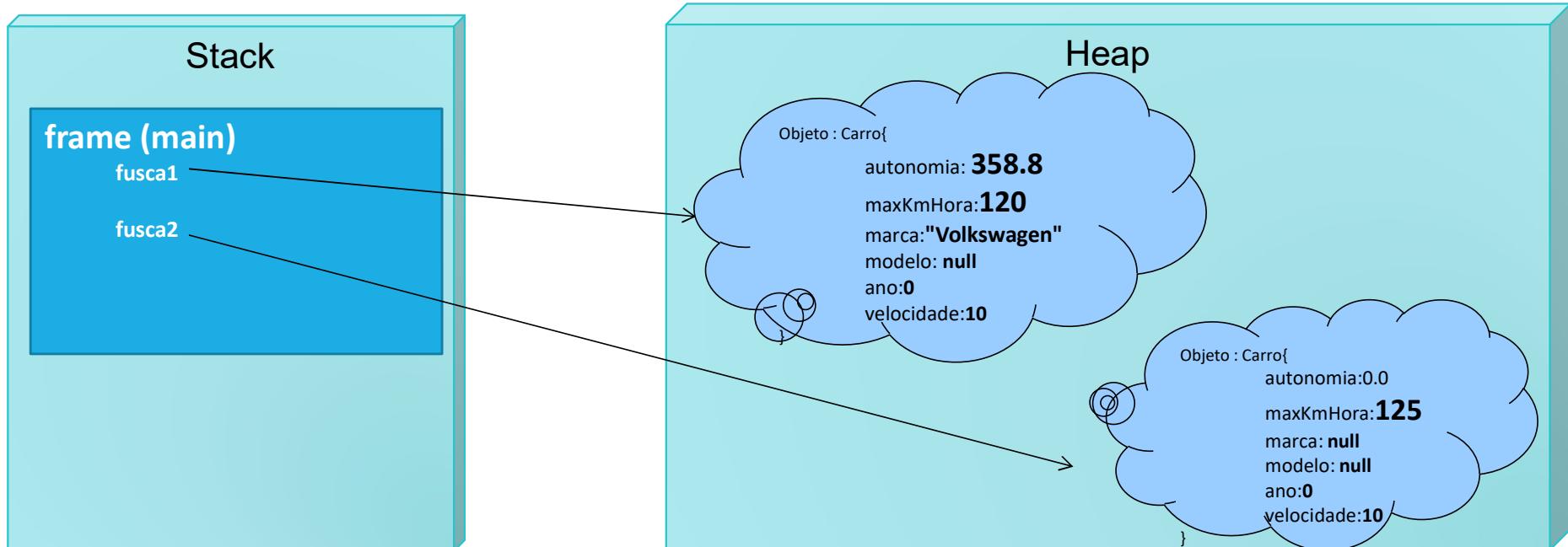
- byte, short, int, long, float e double receberão o valor **0**
- boolean receberá **false**
- char receberá o caractere '**\u0000**'
- demais objetos receberão **null**

Código:

```
Carro fusca1 = new Carro();  
Carro fusca2 = new Carro();  
fusca1.autonomia = 358.8f;  
fusca1.maxKmHora = 120;  
fusca2.maxKmHora = 125;  
fusca1.marca = "Volkswagen";  
fusca1.acelerar();  
fusca2.acelerar();
```

Objetos Instanciados

Objetos são criados na memória a partir do momento em que são instanciados



Objetos - Exercício

Com base nas classes criadas no exercício anterior, instancie **2** objetos para cada classe, e escreva códigos para modificar suas características e executar seus comportamentos.

Dúvidas



Quiz Time

Kahoot

POO – Orientação a Objetos 1

Acessar o link

<https://play.kahoot.it/v2/?quizId=b7cfe37d-8d7d-4b57-b897-75ed64bac870>

Métodos

Métodos

- Os métodos são rotinas ou funções que possuem acesso a instância do objeto, podendo alterar suas características e invocar outros comportamentos. Através do uso de métodos é possível dividir e organizar o programa em rotinas, facilitando a compreensão do código.
- Os métodos são identificados por um nome e podem receber e retornar informações .
- No Java os métodos:
 - Podem ou não retornar objetos
 - Possuem *namespace* próprio portanto as variáveis criadas dentro do método fazem parte de um escopo local

Métodos

Os métodos no Java declaram o tipo de informação a ser retornada assim como em algumas outras linguagens tipadas.

O tipo da informação retornada, assim como o nome do método e seus parâmetros são conhecidos como assinatura

Métodos

Sintaxe:

```
[modificadores] <tipo de informação retornada> <nome do método> ([<Tipo paramentro1> <Nome Parametro1>,] ... [<Tipo parmentroN> <Nome paramentroN>])  
{  
    return [<valor>];  
}
```

Exemplo :

```
boolean matricularDisciplina( String disciplina ) {  
    System.out.println("Matricula efetivada em " + disciplina);  
    return (true);  
}
```

Métodos

O método pode ser invocado pelo nome e um par de parenteses contendo ou não parâmetros, obedecendo a forma como foi declarado.

```
Aluno a = new Aluno();  
boolean b = a.matricularDisciplina( "Matematica" );  
System.out.println( b );  
System.out.println( a.matricularDisciplina("POO") );
```

Métodos

- Os parâmetros passados para a função são atribuídos aos identificadores através da ordem em quem foram passados.
- Exemplo:

```
void fazAlgo( int numero1, int numero2, float numero3 ) {  
    System.out.println( "Este é o 1o parametro informado :" + numero1 );  
    System.out.println( "Este é o 2o parametro informado :" + numero2 );  
    System.out.println( "Este é o 3o parametro informado :" + numero3 );  
}  
  
ClasseExemplo c = new ClasseExemplo();  
c.fazAlgo(10, 10, 4.5);
```

- Nesta caso o número float foi para a 3^a variável e os números inteiros para a 1^a e 2^a variável.

Métodos que usam Lista Variável de Argumentos (var-args)

Através do uso do var-args é possível passar um número variável de argumentos no momento de chamada de um método.

parâmetro : informação solicitada pelo método no momento em que é declarado, compõem sua assinatura

EX :

```
class Funcionario {  
    float salario;  
    public void adicionarBeneficios( float a, float b ) {  
        salario += a;  
        salario += b;  
    }  
}
```

argumento : informação passada ao método no momento em que ele é chamado.

EX :

```
Funcionario f = new Funcionario();  
f.adicionarBeneficios( 100.0, 200.0 );
```

Métodos que usam Lista Variável de Argumentos (var-args)

Sintaxe :

```
[modificador de acesso] [outros modificadores] <tipo> <nome> ([ [<tipo1> <parametro1>],  
    [<tipo2> <parametro2>],  
    [<tipo N-1> <parametroN-1>],  
  
    [<tipo var-args>••• <parametro var-args>]  
]);
```

Tipo var-args pode ser um tipo primitivo ou uma objeto

Regras:

1. Pode haver outros parâmetros em um método que espera um var-args, porém o parâmetro var-args precisa ser o último parâmetro esperado.
2. So pode haver um parâmetro var-args por método

Métodos que usam Lista Variável de Argumentos (var-args)

Exemplo de métodos que utilizam var-args

```
public int somar (int... valores) { }
public int somareElevar(int exp, int... valores){ }
public String concatenar(String... textos){ }
```

O parâmetro do tipo var-args é recebido pelo método como sendo um vetor

```
class Funcionario {
    float salario;
    public void adicionarBeneficios( float ... beneficios ) {
        for (float bene : beneficios) {
            salario += bene;
        }
    }
}
```

Métodos - Exercício

Nosso calendário atual é Gregoriano.

Faça um método que receba como argumento o dia, mês e ano do nosso calendário gregoriano, e retorne o número do dia correspondente no calendário Juliana

Fórmula :

$$\begin{aligned} \text{data_juliana} = & (1461 * (\text{ano} + 4800 + (\text{mes} - 14) / 12)) / 4 + \\ & (367 * (\text{mes} - 2 - 12 * ((\text{mes} - 14) / 12))) / 12 - \\ & (3 * ((\text{ano} + 4900 + (\text{mes} - 14) / 12) / 100)) / 4 + \text{dia} - 32075 \end{aligned}$$

Para testar informe para a função a data de inicio das olimpíadas do Brasil 05 de Agosto de 2016. O resultado deverá ser : 2457606

Dúvidas

