Java Iniciação

Sessão 16-07-21

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
ASSOCIAÇÃO PORTUGUESA DAS EMPRESAS DO SECTOR ELÉCTRICO E ELECTRÓNICO
PO ISE
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

Tecnologia JAVA

Modelos Básicos

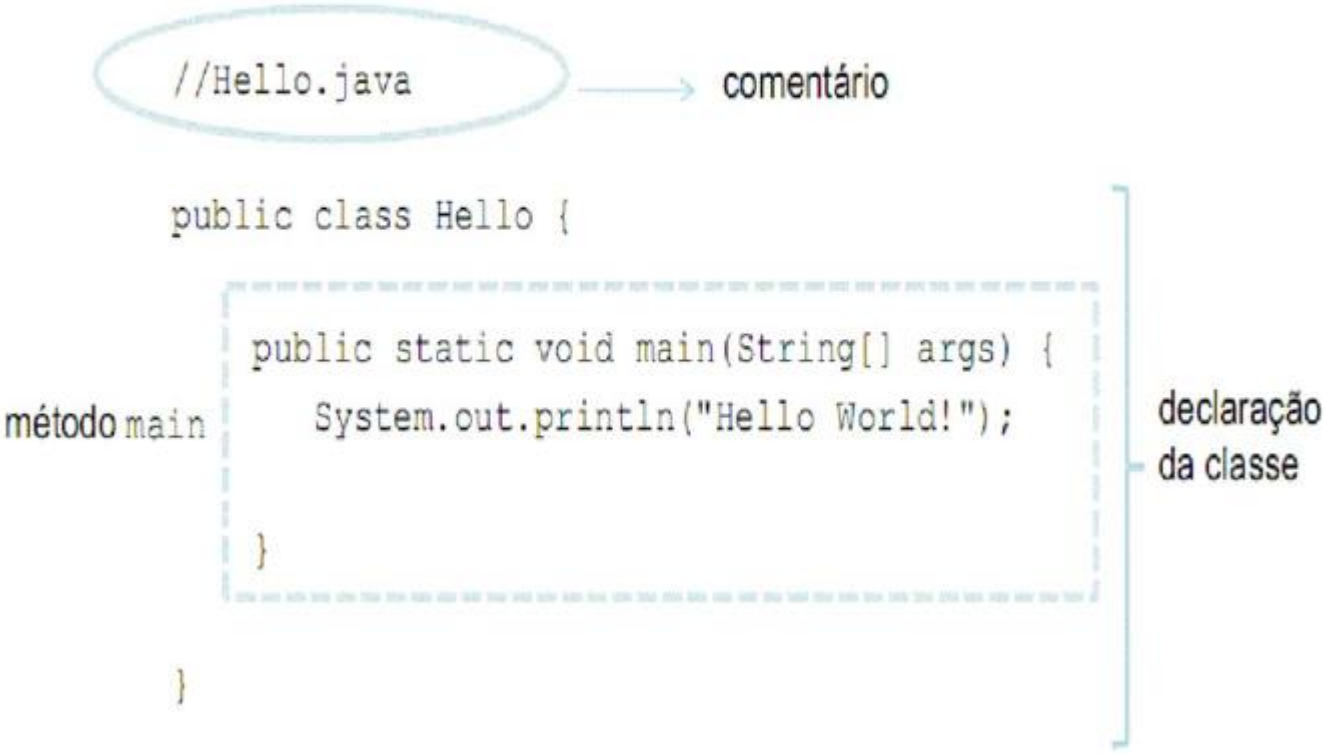
Celina Lopes

Sumário

- Introdução;
- Classes;
- Objetos;
- Métodos;
- Instância;
- Tipo de dados;

INTRODUÇÃO

Estrutura base de um programa em Java



Declaração da Classe

```
public class ClassName
{
    Corpo da classe.
}
```

- **public** e **class** são duas palavras reservadas no Java
- Cada ficheiro contém a definição de pelo menos uma classe
  - Se a classe for declarada como **public**, o nome do ficheiro tem de ser igual ao nome da classe
  - Um ficheiro só poderá ter, no máximo, uma classe declarada como **public**

Método *Main*

```
public static void main (String[] args) {
    // Corpo do método
}
```

- Ponto de entrada do programa Java
  - Ao executamos um programa Java, a JVM invoca o método main para iniciar a aplicação.

Convenções

- Existe um conjunto de convenções (boas práticas) que devem ser seguidas durante o desenvolvimento de aplicações
- Uma boa prática é a produção de código bem formatado e limpo:
  - Facilita a sua manutenção;
  - Aumento da legibilidade
    - Muitos programadores podem contribuir para o desenvolvimento de uma aplicação.

Objectos, classes e instâncias

- Classes
  - Representam tipos ou categorias de "coisas"
  - Uma classe descreve as características partilhadas de um tipo de coisa
    - As características que fazem com que uma coisa seja o que é
  - Por exemplo, a classe Carro representa os carros genéricamente
    - Têm quatro rodas, um motor, …
- Objectos
  - Representam 'coisas' específicas de um tipo de coisa
    - Um item/elemento/indivíduo específico de um tipo de coisa
  - Por exemplo, um objecto representa o carro vermelho ali no parque de estacionamento
- Os objectos são criados a partir de classes.
  - A classe descreve o tipo de objecto
  - Os objectos representam **instâncias** individuais da classe

Atributos e Estado de um Objecto

- Um objecto tem **atributos**, ou campos, que representam características do objecto e que têm valores associados.
- A classe define quais os atributos que um objecto tem, mas cada objecto armazena o seu próprio conjunto de valores para os seus atributos (**o estado do objecto**).

Múltiplas instâncias de uma classe

- Múltiplas **instâncias** podem ser criadas a partir de uma única classe.
  - Todas as instâncias (objectos) têm os mesmos atributos, mas em cada instância os atributos podem ter valores diferentes
    - Diferentes estados

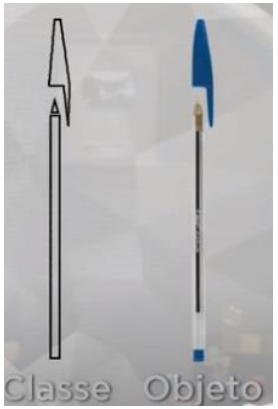
O que é um Objeto ?

Um compromisso é um objeto?
Um aluno é um objeto?

Definição de Objeto:
é uma coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamentos e estado atual.

Uma caneta é um objeto ?

Posso ter outros objetos do tipo caneta?



A caneta é o **objeto**
O molde serve para classificar- o molde da caneta é a **classe**

Uma caneta é um objeto ?

Atributos

Modelo
Ponta
Cor
Quantidade de tinta
Tapada

Métodos

Pintar
Escrever
Destapar
Tapar

Estado

A caneta está:
quase sem tinta
destapada
é azul
escreve fino

Classe caneta

Identificar os atributos da classe

Modelo : string
Cor : string
Quantidade de tinta : double
Ponta : string
Tapada : boolean

metodo pintar()
se (tapada)
escreve("não escreve")
senão escreve ("pode pintar")

método tapar()
tapada = verdadeiro

Criar o objeto a partir da classe

Já conseguimos ver o estado do objeto neste momento

Modelo : Bic cristal
Cor : azul
Quantidade de tinta : 0.5
Ponta : fina
Tapada : verdadeiro

Metodos

Pintar
Escrever
Destapar
Tapar

Estado

A caneta está:
quase cheia
tapada
azul
escreve fino

Criar uma instância classe

Quando consigo criar um objeto através da classe

Instanciar

nome do objeto = nova caneta

C1 = nova Caneta -- tenho um objeto chamado C1 que é uma nova caneta

A palavra nova serve para instanciar um objeto a partir de uma classe;
O que está a seguir ao nova é a classe que euvou utilizar;
O que vem antes do **nova** é o nome do objeto que vai existir;

Métodos

- Objectos têm operações que podem ser invocadas, que se chamam **métodos**.
- O cabeçalho do método chama-se a sua **assinatura**
- Fornece a informação necessária para invocar o método

Valores de retorno

- Todos os métodos podem retornar um resultado, através de um valor de retorno.
- Quando não têm nenhum valor para retornar, retornam o tipo `void`.

Interacção entre objectos

- Objectos podem criar outros objectos
- Objectos podem comunicar invocando métodos uns dos outros
- Num programa Java podem existir centenas ou milhares de objectos
  - O utilizador cria um objecto inicial e todos os restantes objectos são depois criados por este, de forma directa ou indirecta

Terminologia revisitada - Classe

Unidade que contém um conjunto de métodos que realizam as tarefas da classe.

- No mundo real encontramos, frequentemente, diversos objetos do mesmo tipo
- Exemplo:
  - existem diversas bicicletas, todas do mesmo fabricante e modelo, onde cada uma dessas bicicletas foi construída a partir do mesmo projeto e assim contêm os mesmos componentes.
  - recorrendo à terminologia da programação orientada a objetos dizemos que a minha bicicleta é uma instância (ou objeto) da classe Bicicleta

- A classe é o modelo a partir do qual são criados os objetos individualmente.
- Ao definir a classe é necessário declarar:
  - Campos/Atributos (estrutura de dados): características numa classe cujos valores diferem de objeto para objeto.
  - Métodos: funções dentro de uma classe que atuam sobre as suas instâncias. Representam, na sua generalidade, comportamentos.

Exemplos

- Classes
  - Veiculo;
  - Bola;
  - Computador.
- Atributos da classe Automóvel
  - Cor;
  - Número de Portas;
  - Motor.
- Métodos da classe Automóvel
  - Ligar;
  - Travar;
  - Acelerar.

Classes

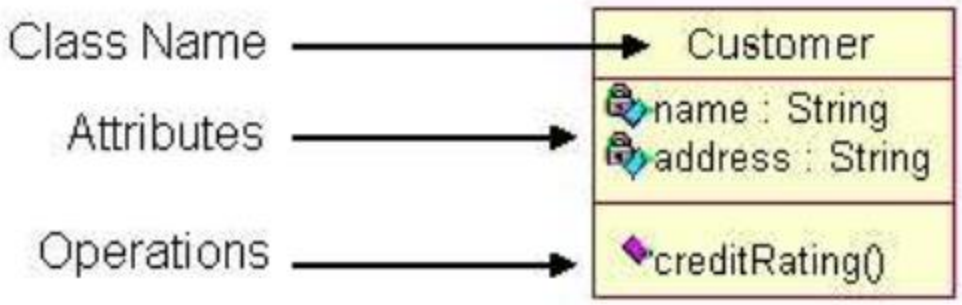


Figura : Representação UML de uma classe

Forma geral de uma classe

```
class NomeDaClasse {

    //variáveis de instância - atributos
    int var1;
    int var2;
    int var3;

    //declaração de métodos
    void metodo1 (int parametros){
        //corpo do método
    }

    void metodo2 (int parametros){
        //corpo do método
    }

    void metodo3 (int parametros){
        //corpo do método
    }

}
```

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

Quais são as características / atributos que um carro pode ter?

Definição da classe Carro e seus atributos

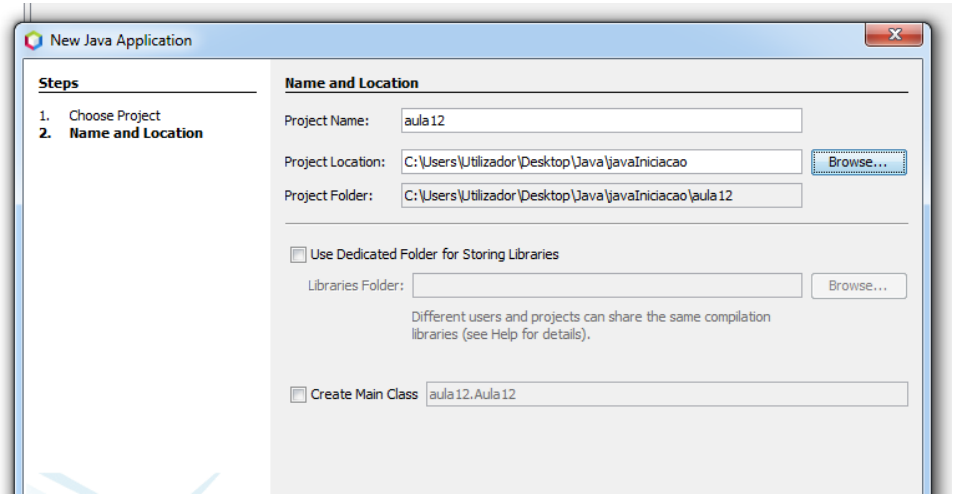
```
class Carro {

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

}
```

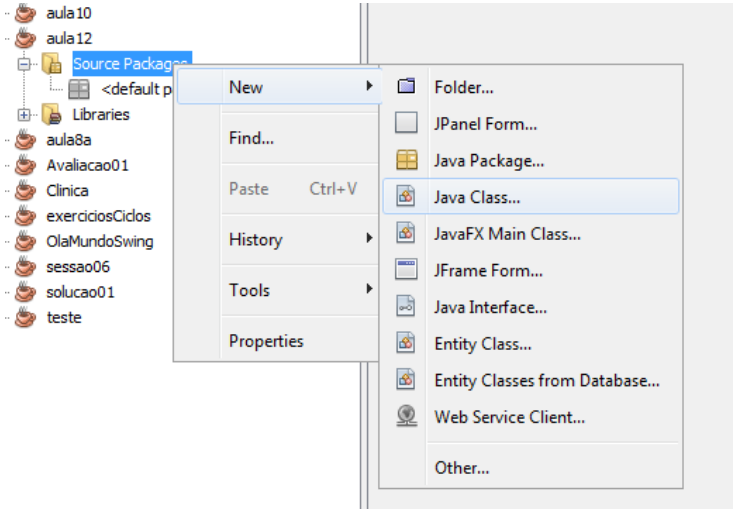
Criar esta classe no netbeans

Nome da classe: Carro



Criar esta classe no netbeans

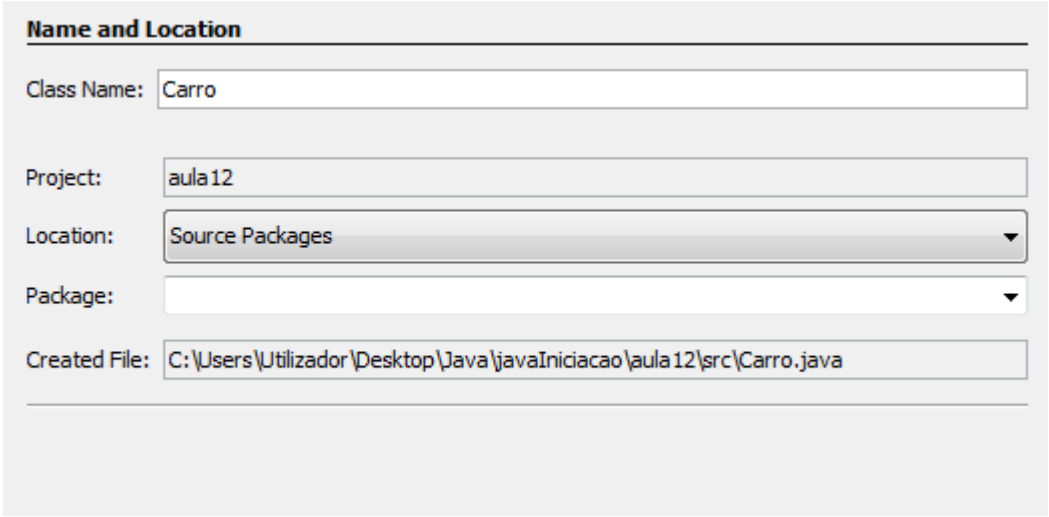
Nome da classe: Carro

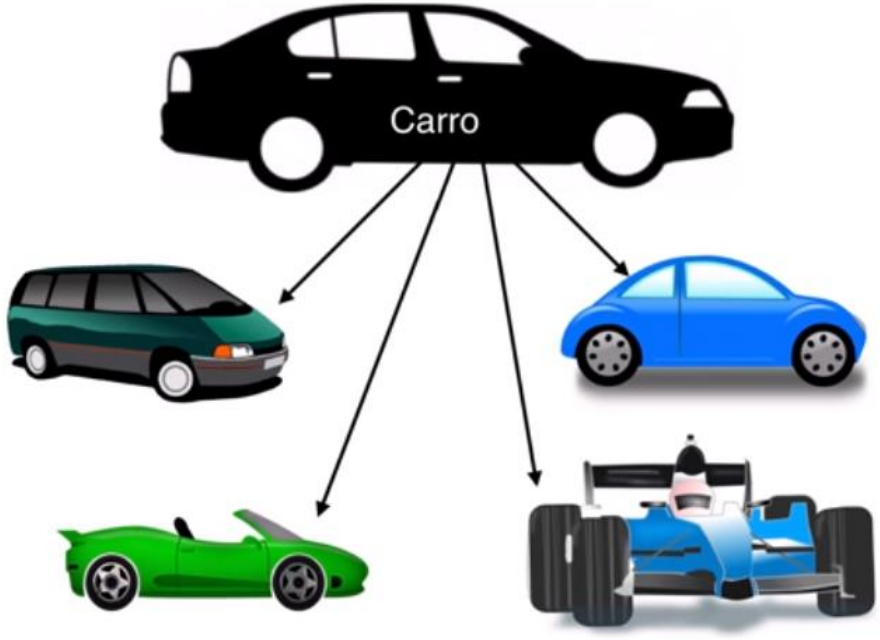


Sem o método main, porque só vamos Criar e declarar a classe

Criar esta classe no netbeans

Nome da classe: Carro





Temos a classe Carro com 5 Atributos.

Como fazemos para os usar?

Vamos criar o objeto

Posso ter pelo menos 4 carros <u>diferentes</u> (4 objetos diferentes)

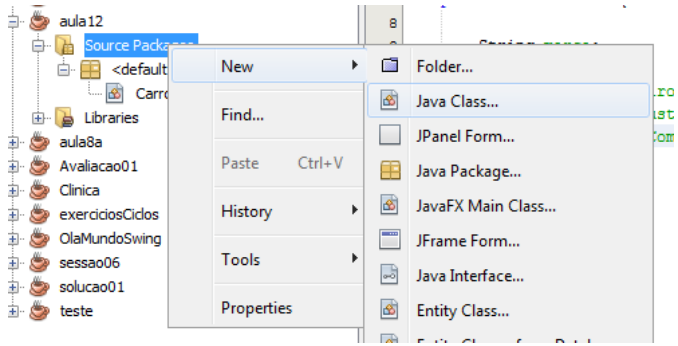
Criação dos objetos

Instanciamos (criamos uma instância da classe) através da palavra new

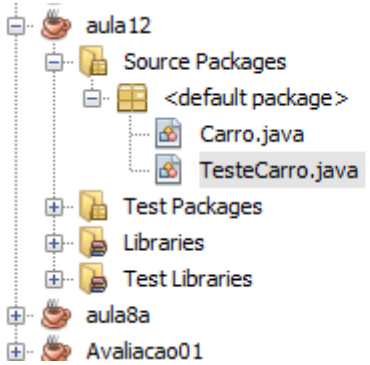
Instanciamos (criamos uma instância da classe) através da palavra new

Criação dos objetos

Criar uma classe de teste : para testar a criação dos nossos objetos da classe Carro



Criar a classe de teste com método main




```
8  /**
9   *
10  * @author Celina
11  */
12 public class TesteCarro {
13     public static void main(String[] args) {
14
15     }
16
17 }
```

Objetos

Terminologia revisitada - Objeto

similar a um objeto real e que consiste num estado e num comportamento relacionado.

- Objetos/Instâncias: concretização de uma classe numa entidade particular; as instâncias de uma classe têm todas as mesmas operações, as mesmas estruturas de dados, mas valores próprios.
- É relativamente simples encontrar exemplos de objetos reais: um cão, uma secretária, uma televisão, uma bicicleta, etc.. Os objetos reais partilham duas características: estado e comportamento.
  - Os cães têm um estado (nome, raça, cor, fome, etc.) e comportamento (ladrar, ganir, abanar a cauda, etc.).
  - As bicicletas têm também um estado (mudança atual, cadência atual do pedal, velocidade atual) e comportamento (mudar mudança, mudar cadência do pedal, travar).

Exercícios

Exercícios

1. Implemente, utilizando a linguagem Java, duas instâncias da classe Automóvel
2. Implemente, utilizando a linguagem Java, duas instâncias para cada uma das seguintes classes.
   - Livro
   - Mesa
   - Árvore

Métodos

Terminologia revisitada - método

Alberga as instruções que executam uma tarefa.

- Os métodos especificam o comportamento que um objeto pode ter ou seja, os métodos, em Java, determinam as mensagens que um objeto pode receber
  - para que o objeto A peça um serviço ao objeto B, A precisa de conhecer a linguagem que deve utilizar para comunicar com B.

Declaração de um método I

[NivelAcesso][[static]|[abstract]][final][native][synchronized]
TipoRetorno NomeMetodo (ListaArgumentos) [throws exception]

- **NivelAcesso** – definição dos níveis de acesso ao método;
  - **private** – o nível de acesso mais restritivo. Um método declarado como private só pode ser acedido dentro da própria classe onde é declarado;
  - **package** – nível de acesso atribuído quando não há declaração explicita do nível de acesso. Este tipo de acesso permite que todos os métodos de todas as classes da mesma package acedam ao método;

Declaração de um método II

- **protected** – Um método declarado como protected pode ser acedido por métodos da mesma classe, por métodos de classes da mesma package e por métodos das subclasses da classe onde está declarado;
- **public** – o nível de acesso menos restritivo. Um método com este nível de acesso poderá ser chamado a partir de qualquer método de qualquer classe.

- **static** – modificador para declarar um método de classe;
- **abstract** – modificador que declara um método sem implementação e que deve ser membro de uma classe abstrata;
- **final** – declara um método que não pode ser reescrito nas subclasses da classe onde está declarado;

Declaração de um método III

- **native** – declara um método que está implementado noutra linguagem;
- **synchronized** – *threads* a correr de forma concorrente por vezes chamam métodos que operam sobre os mesmos dados. A utilização deste modificador permite que as diversas *threads* acedam à informação de forma segura.
- **TipoRetorno** – tipo de dados do valor devolvido ou void se o método não retornar um valor;
- **NomeMetodo** – Nome do método;
- **ListaArgumentos** – Lista de parâmetros de entrada, separados por vírgulas;
- **throws exception** – Devolve uma Exceção caso ocorra um erro.

Assinatura e invocação

- O nome de um método e o tipo de parâmetros constituem a assinatura do método, que identifica univocamente esse método
- Os métodos só podem ser criados integrados numa classe.
- Um método apenas pode ser invocado por um objeto;
- A chamada de um método é efetuada da seguinte forma:
  - <NomeObjeto>.<nomeMetodo> (arg1, arg2, . . .);

Retorno

- um método retorna ao código que o invocou quando:
  - completa todas as instruções do método;
  - encontra uma instrução return;
  - lança uma exceção.
- o tipo de dados de retorno é definido na declaração do método;
- métodos declarados como void não retornam qualquer valor e não necessitam que seja utilizada a instrução return;
- O tipo de dados do valor de retorno tem de ser igual ao tipo de dados declarado para o valor de retorno.

Definição da classe Carro e seus atributos

```
class Carro {

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

}
```

Método simples: sem retorno e/ou parâmetro

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

Método com retorno

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

Método com parâmetro

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