



escola
britânica de
artes criativas
& tecnologia

WORKSHOP INTENSIVO DE 3 DIAS

Descomplicando a programação com Java

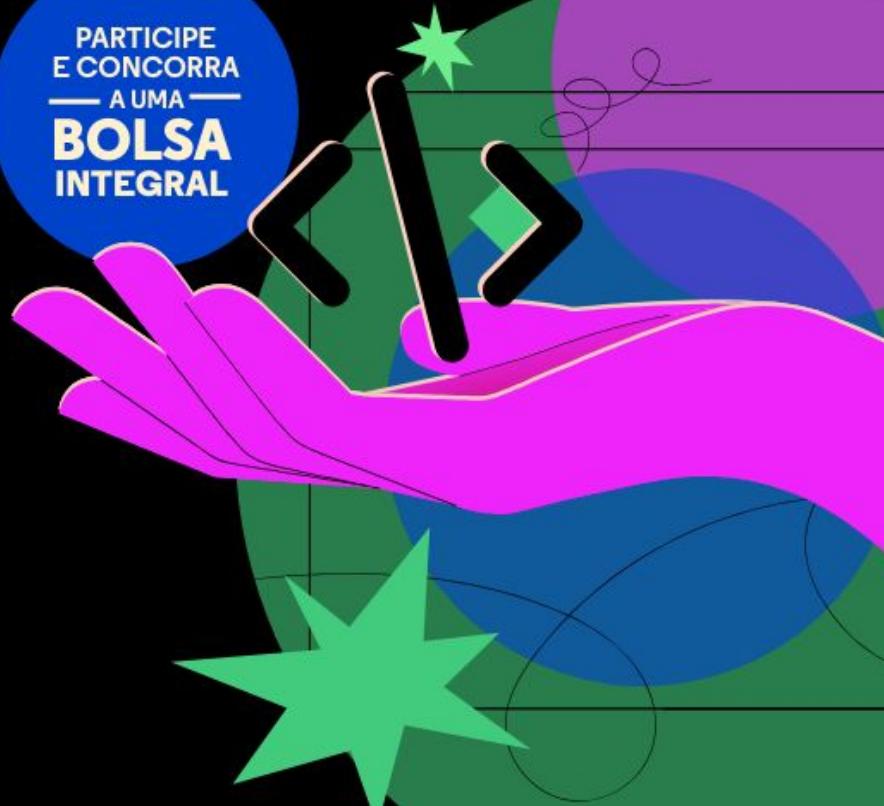
15, 16 e 17 DE JUNHO, 19h



César Lawall Silveira

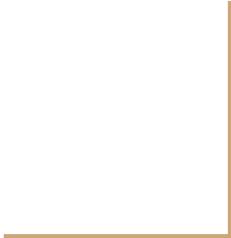
Desenvolvedor na Creditas

PARTICIPE
E CONCORRA
A UMA
**BOLSA
INTEGRAL**





Descomplicando a programação com Java



Workshop - dia 1

Agenda

Dia 1

Java e Orientação a Objetos.

Dia 2

Neste dia criaremos uma aplicação simples com acompanhamento do instrutor.

Dia 3

Faremos a correção de alguns exercícios em tempo real e responderemos dúvidas ao vivo.

O que é Java?

O que é Java?

- Linguagem de alto nível;
- Completamente Orientada a Objetos*;
- Linguagem interpretada pela JVM*;
- Criada por James Gosling, na Sun Microsystems;
- Lançada em 1995;
- Sintaxe baseada em C / C++;
- Adquirida pela Oracle como parte da Sun Microsystems em 2009-2010*;



Sun
ORACLE

Objetivos da Linguagem Java

- Simples, Orientada a Objeto e Familiar;
- Robusta e Segura;
- Com uma arquitetura neutra e portável;
- Alta Performance;
- Interpretada, *threaded* e dinâmica;



Qual a diferença de linguagem compilada para linguagem interpretada?

Receita de bolo instantâneo:

Ingredientes para 4 pessoas

Imagine que temos uma receita em
Português...

. 100 g de chocolate preto

. 50 g de manteiga

. 60 g de farinha

. 40 g de açúcar

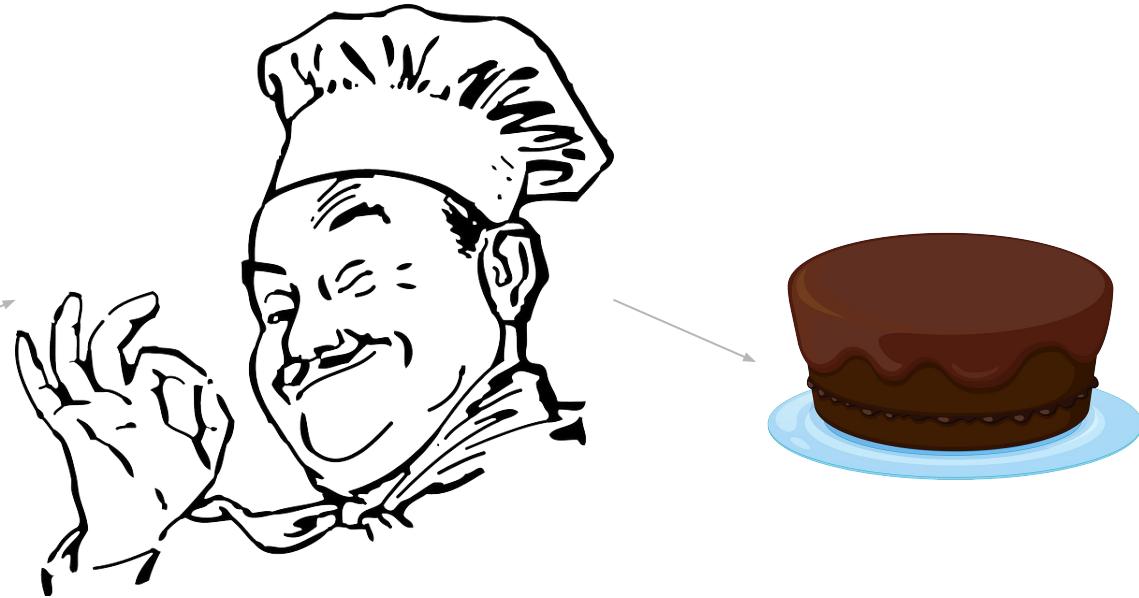
Receita de bolo instantâneo:

Ingredientes para 4 pessoas

- 100 g de chocolate preto
- 50 g de manteiga
- 60 g de farinha
- 40 g de açúcar

Como o chef entende a
receita, ele consegue
executar seus passos
Receita de bolo instantâneo.
sozinho!

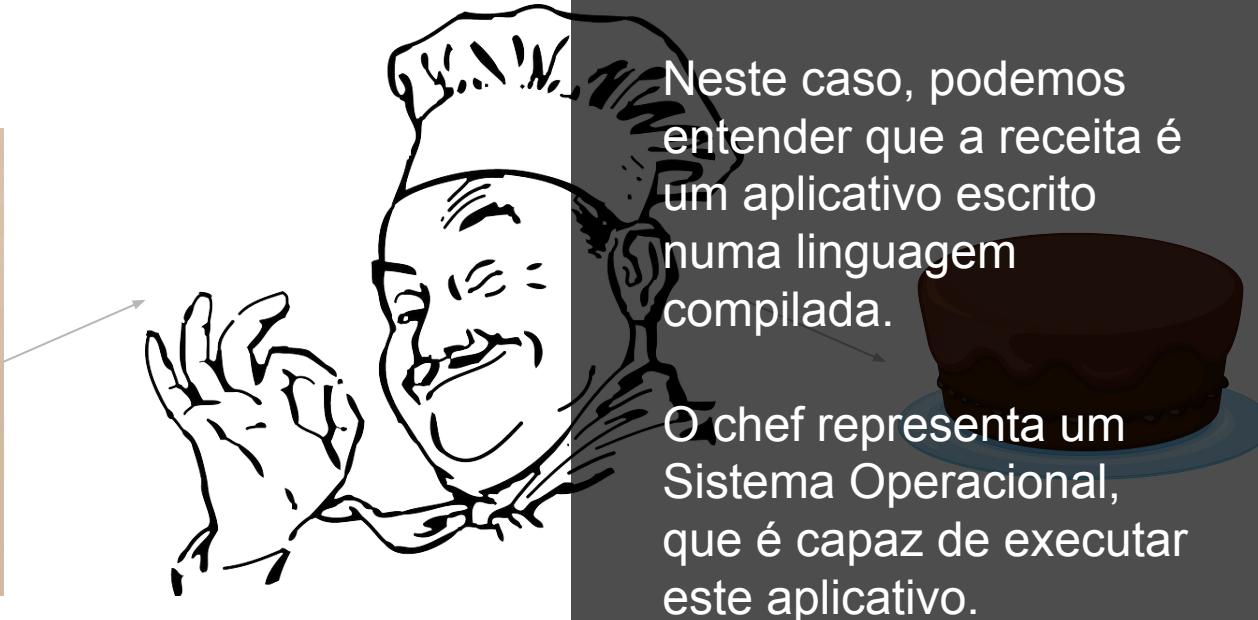
- Tempo de cozimento: 30 minutos
para 4 pessoas
- 100 g de chocolate preto
 - 50 g de manteiga
 - 60 g de farinha
 - 40 g de açúcar
 - 1/2 sachê de fermento
 - 100ml de creme de leite
 - 2 ovos



Receita de bolo instantâneo:

Ingredientes para 4 pessoas

- 100 g de chocolate preto
- 50 g de manteiga
- 60 g de farinha
- 40 g de açúcar
- 1/2 sachê de fermento
- 100ml de creme de leite
- 2 ovos



Neste caso, podemos entender que a receita é um aplicativo escrito numa linguagem compilada.

O chef representa um Sistema Operacional, que é capaz de executar este aplicativo.

Receita de bolo instantâneo:

Ingredientes para 4 pessoas

- 100 g de chocolate preto
- 50 g de manteiga
- 60 g de farinha
- 40 g de açucar
- 1 1/2 sachê de fermento
- 100ml de creme de leite
- 2 ovos



Recette : Gâteau express.

Ingrédients pour 4 personnes

Mas e se a receita estiver em francês?

- 100 g de chocolat noir
- 50 g de beurre
- 60 g de farine
- 40 g de sucre
- 1/2 sachet de levure

Recette : Gâteau express.

Ingrédients pour 4 personnes

- 100 g de chocolat noir
- 50 g de beurre
- 60 g de farine
- 40 g de sucre
- 1/2 sachet de levure

Nosso chef não fala francês, então ele não entende a receita!

Ingédients pour 4 personnes

100 g de chocolat noir

chamou um amigo para traduzir cada passo da
receita enquanto ele cozinha.

40 g de sucre

1/2 sachet de levure

ame li qui de



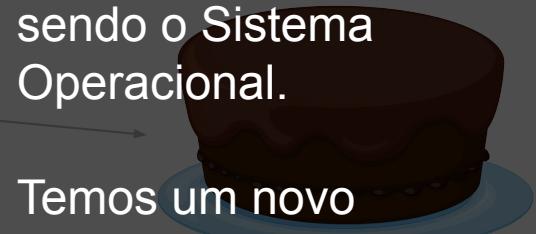
Reette : Gâteau express.

Ingrédients pour 4 personnes

- 100 g de chocolat noir
- 50 g de beurre
- 60 g de farine
- 40 g de sucre
- 1/2 sachet de levure
- 100ml de crème liquide



Neste caso, nossa receita é um aplicativo escrito numa linguagem interpretada (como Java) e o chef continua sendo o Sistema Operacional.

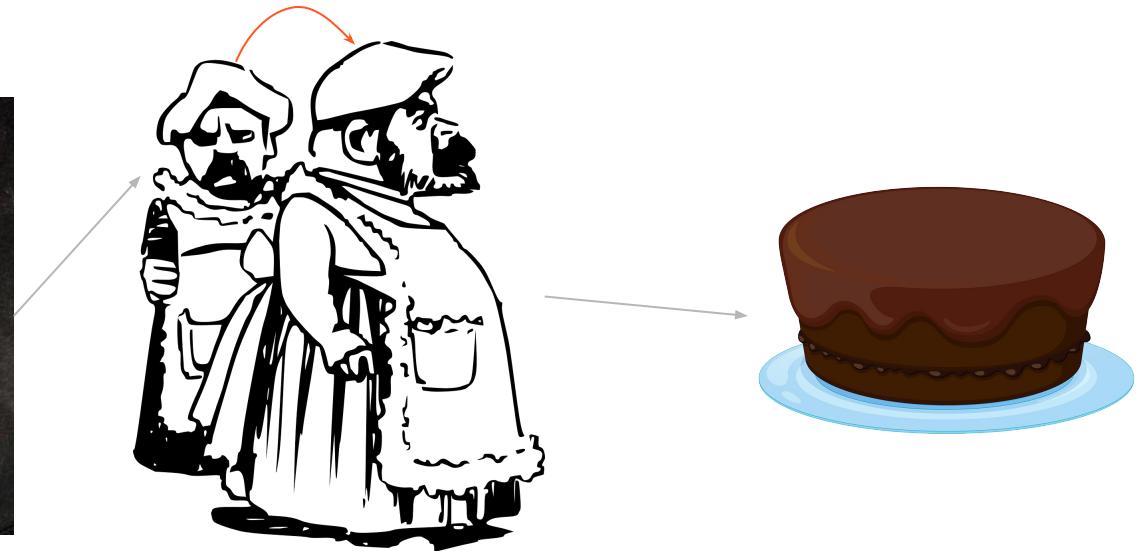


Temos um novo participante na cozinha, o interpretador: Ele traduz a receita para o chef executar!

Recette : Gâteau express.

Ingrédients pour 4 personnes

- 100 g de chocolat noir
- 50 g de beurre
- 60 g de farine
- 40 g de sucre
- 1/2 sachet de levure
- 100ml de crème liquide

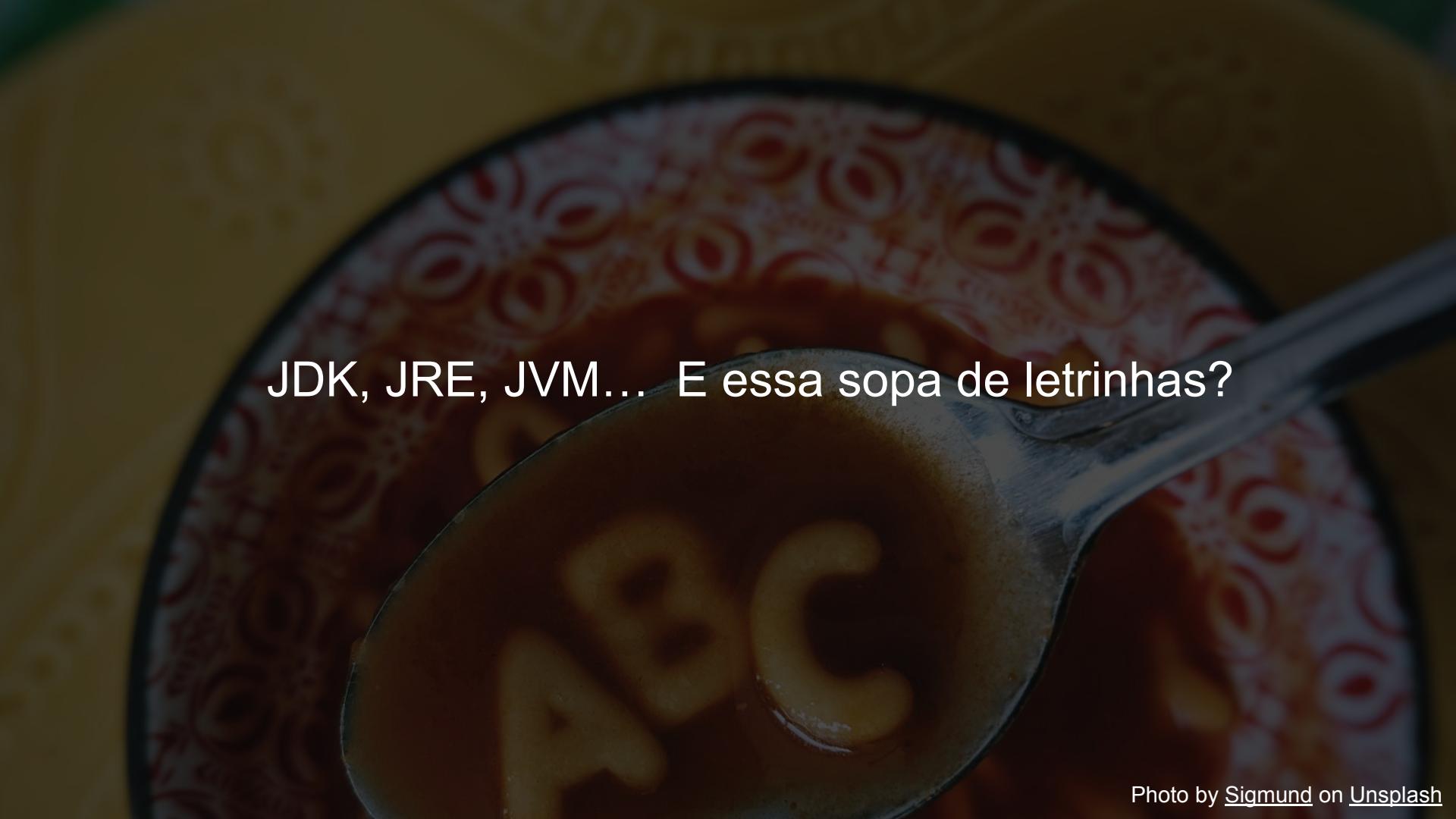


Então linguagem compilada é melhor?

Melhor não, apenas diferente!

Dentre as vantagens de linguagens interpretadas temos o lema do Java:

Write Once, Run Anywhere.

A close-up photograph of a bowl of soup. A silver spoon is resting in the bowl, which contains several floating, semi-transparent letters spelling out "JDK, JRE, JVM... E essa sopa de letrinhas?". The bowl has a red and gold patterned rim. The background is a warm, yellowish-orange color.

JDK, JRE, JVM... E essa sopa de letrinhas?

O que são JDK, JRE e JVM?

- JDK significa Java Development Kit. O kit é composto de ferramentas e bibliotecas de apoio que são instaladas no computador do desenvolvedor para auxiliar na escrita do aplicativo;
- JRE significa Java Runtime Environment. É um ambiente de execução que deve ser instalado na máquina onde será executado o aplicativo;
- JVM é a Java Virtual Machine ou Máquina Virtual Java. Ela equivale ao “amigo tradutor” do chef, sendo responsável por interpretar os comandos do aplicativo e os envia para executar no Sistema Operacional (nossa chef);

Instalação

Ao instalar a JDK, ela automaticamente
instala a JRE e a JVM.

Java SE Downloads

Java Platform, Standard Edition

Java SE 16

Java SE 16.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
 - Binary License
 - Documentation License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme

Oracle JDK

- [JDK Download](#)
- [Documentation Download](#)

Looking for Oracle OpenJDK builds?

- **Oracle Customers and ISVs targeting Oracle LTS releases:** Oracle JDK is Oracle's supported Java SE version for customers and for developing, testing, prototyping or demonstrating your Java applications.
- **End users and developers looking for free JDK versions:** Oracle OpenJDK offers the same features and performance as Oracle JDK under the GPL license .

Java SE Downloads

Java Platform, Standard Edition

Java SE 16

Java SE 16.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
 - Binary License
 - Documentation License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme



Looking for Oracle OpenJDK builds?

- **Oracle Customers and ISVs targeting Oracle LTS releases:** Oracle JDK is Oracle's supported Java SE version for customers and for developing, testing, prototyping or demonstrating your Java applications.
- **End users and developers looking for free JDK versions:** Oracle OpenJDK offers the same features and performance as Oracle JDK under the GPL license .

Java SE Development Kit 16 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Important Oracle JDK License Update

The Oracle JDK License has changed for releases starting April 16, 2019.

The new [Oracle Technology Network License Agreement for Oracle Java SE](#) is substantially different from prior Oracle JDK licenses. The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle JDK licenses may no longer be available. Please review the terms carefully before downloading and using this product. An FAQ is available [here](#).

Commercial license and support is available with a low cost [Java SE Subscription](#).

Oracle also provides the latest OpenJDK release under the open source [GPL License](#) at [jdk.java.net](#).

See also:

[Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.

[Java Developer Day hands-on workshops \(free\)](#) and other events

[Java Magazine](#)

[JDK 16.0.1 checksum](#)

Java SE Development Kit 16.0.1

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	144.87 MB	jdk-16.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	160.72 MB	jdk-16.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	146.16 MB	jdk-16.0.1_linux-x64_bin.deb
Linux x64 RPM Package	152.99 MB	jdk-16.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	170.02 MB	jdk-16.0.1_linux-x64_bin.tar.gz
macOS Installer	166.58 MB	jdk-16.0.1_osx-x64_bin.dmg
macOS Compressed Archive	167.2 MB	jdk-16.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	150.56 MB	jdk-16.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	168.78 MB	jdk-16.0.1_windows-x64_bin.zip

Java SE Development Kit 16.0.1

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	144.87 MB	 jdk-16.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive		 jdk-16.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package		 jdk-16.0.1_linux-x64_bin.deb
Linux x64 RPM Package		 jdk-16.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive		 jdk-16.0.1_linux-x64_bin.tar.gz
macOS Installer	166.58 MB	 jdk-16.0.1_osx-x64_bin.dmg
macOS Compressed Archive	167.2 MB	 jdk-16.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	150.56 MB	 jdk-16.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	168.78 MB	 jdk-16.0.1_windows-x64_bin.zip

Required

I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

[Download jdk-16.0.1_windows-x64_bin.exe](#)

Java SE Development Kit 16.0.1

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	144.87 MB	 jdk-16.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive		 jdk-16.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package		 jdk-16.0.1_linux-x64_bin.deb
Linux x64 RPM Package		 jdk-16.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive		 jdk-16.0.1_linux-x64_bin.tar.gz
macOS Installer	166.58 MB	 jdk-16.0.1_osx-x64_bin.dmg
macOS Compressed Archive	167.2 MB	 jdk-16.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	150.56 MB	 jdk-16.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	168.78 MB	 jdk-16.0.1_windows-x64_bin.zip



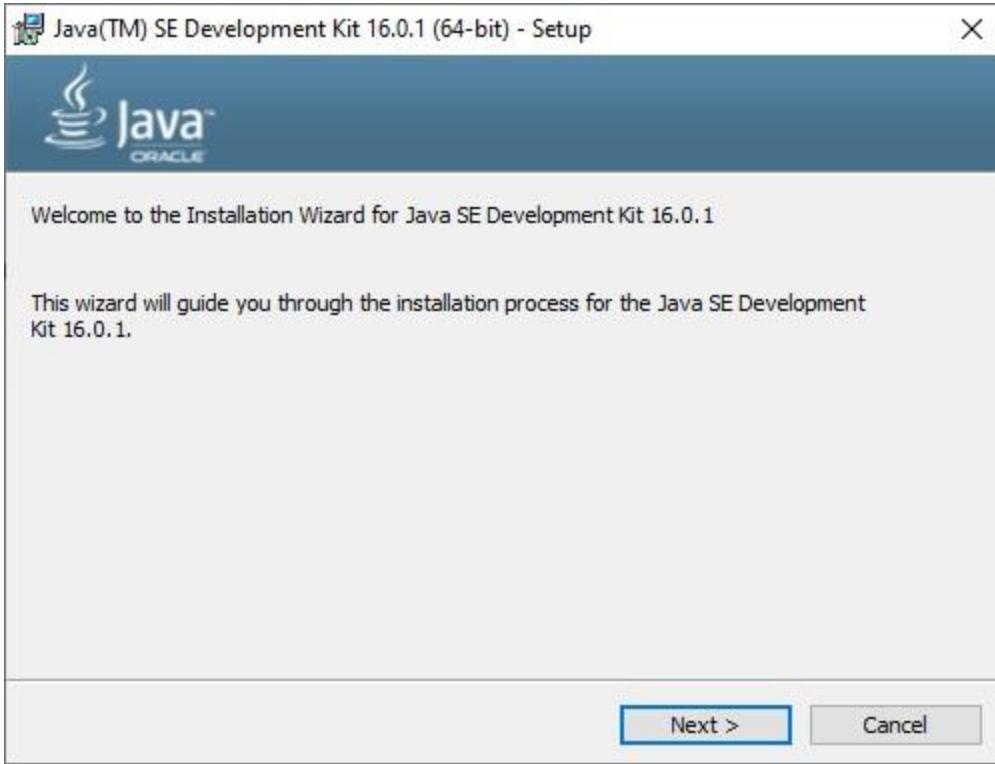
You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

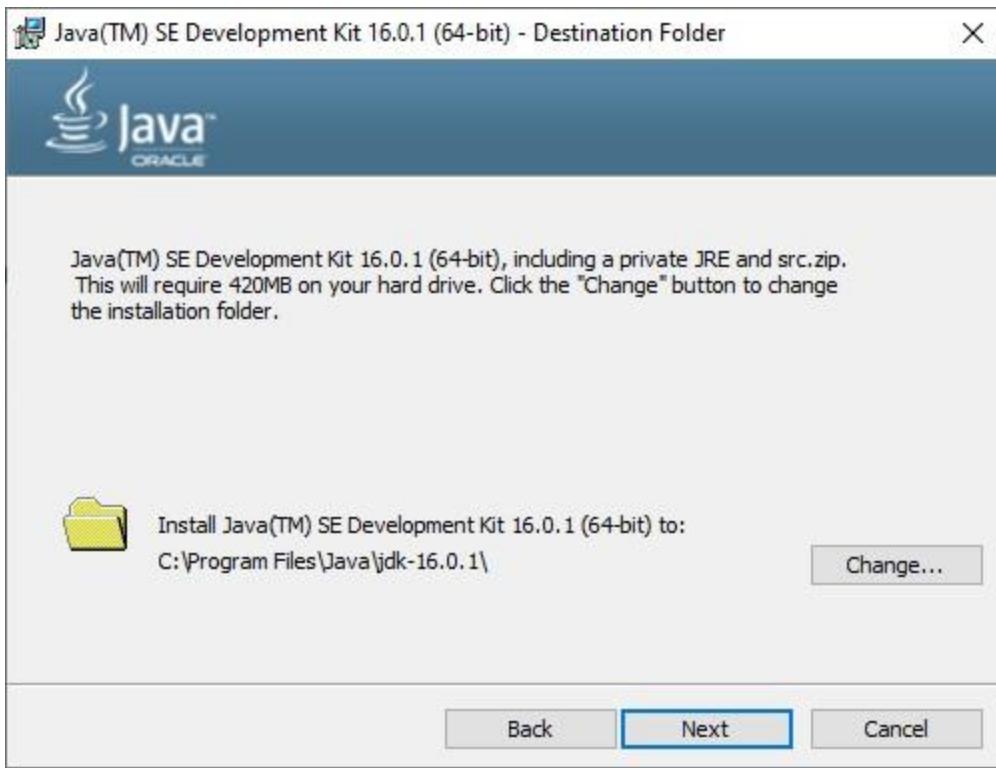
Required

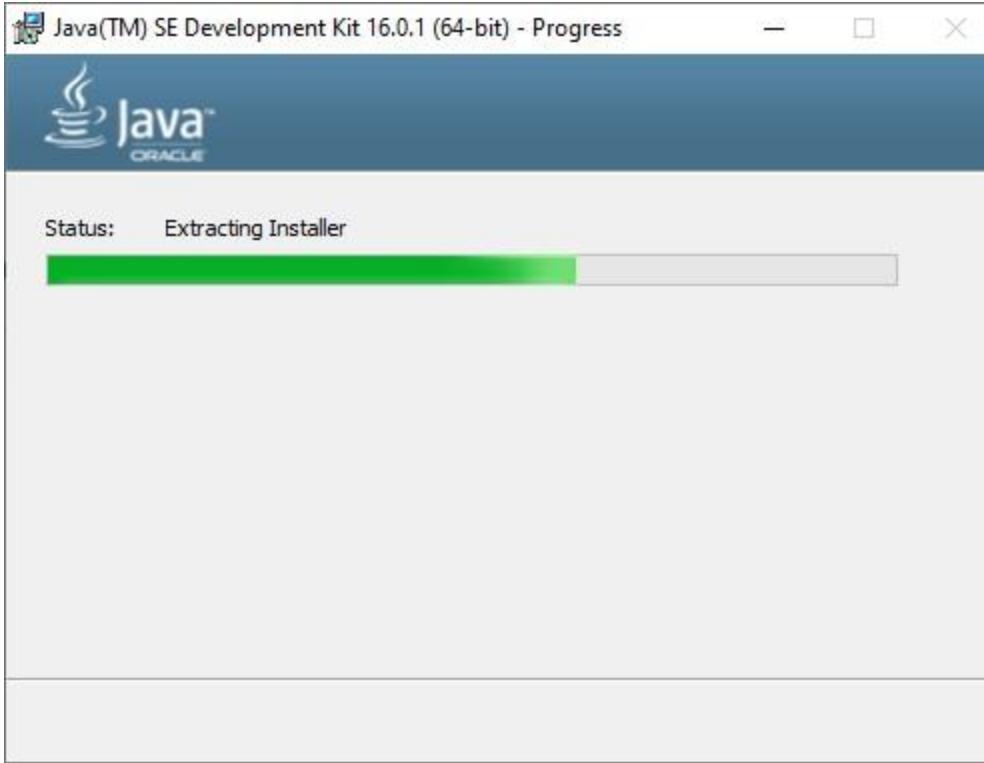
I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

[Download jdk-16.01_windows-x64_bin.exe](#)













Object-Oriented Programming



Programação Orientada a Objetos

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Vamos analisar o código ao lado...

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Precisamos criar variáveis para cada um dos atributos de cada carro que precisamos.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

A descrição do carro também precisa ser repetida para cada um deles...

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
          
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

E por último, as **velocidades** do carro são definidas de forma diferente:

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

O primeiro carro teve um incremento na sua velocidade inicial.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Já o **segundo** carro teve sua velocidade definida de forma instantânea.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

E por último, estamos comparando a velocidade dos carros e escrevendo um texto na tela!

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        String marcaCarro1 = "Honda";  
        String marcaCarro2 = "Volkswagen";  
  
        String modeloCarro1 = "City";  
        String modeloCarro2 = "Golf";  
        int velocidadeCarro1 = 0;  
        int velocidadeCarro2 = 0;  
  
        velocidadeCarro1 = velocidadeCarro1 + 10;  
        velocidadeCarro2 = 50;  
        String marcaModeloCarro1 = marcaCarro1 + " " + modeloCarro1;  
        String marcaModeloCarro2 = marcaCarro2 + " " + modeloCarro2;  
  
        if(velocidadeCarro1 > velocidadeCarro2){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Perceberam que nenhuma variável tem uma relação real com as outras?

Seus nomes indicam que pertencem ao mesmo carro, mas isso depende exclusivamente da atenção do programador.

Classes e Objetos

```
public class Carro {  
  
    String marca;  
  
    String modelo;  
  
    Integer velocidade;  
}
```

Podemos criar uma classe para armazenar as variáveis que têm alguma relação.

```
public class Carro {  
  
    String marca;  
  
    String modelo;  
  
    Integer velocidade;  
}
```

Classes são como um descritor (ou contrato) de um objeto.

São criadas pela palavra-chave class.

```
public class Carro {  
    String marca;  
    String modelo;  
    Integer velocidade;  
}
```

As variáveis tornam-se propriedades da classe e indicam características que aquele objeto pode ter.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.velocidade = carro1.velocidade + 10;  
        carro2.velocidade = 50;  
  
        String marcaModeloCarro1 = carro1.marca + " " + carro1.modelo;  
        String marcaModeloCarro2 = carro1.marca + " " + carro2.modelo;  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Vamos reescrever
nossa código para
utilizar a nova classe!

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.velocidade = carro1.velocidade + 10;  
        carro2.velocidade = 50;  
  
        String marcaModeloCarro1 = carro1.marca + " " + carro1.modelo;  
        String marcaModeloCarro2 = carro1.marca + " " + carro2.modelo;  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.velocidade = carro1.velocidade + 10;  
        carro2.velocidade = 50;  
  
        String marcaModeloCarro1 = carro1.marca + " " + carro1.modelo;  
        String marcaModeloCarro2 = carro1.marca + " " + carro2.modelo;  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Para utilizar uma classe precisamos **criar um objeto** a partir dela.

Isso é feito com o operador **new!**

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.velocidade = carro1.velocidade + 10;  
        carro2.velocidade = 50;  
  
        String marcaModeloCarro1 = carro1.marca + " " + carro1.modelo;  
        String marcaModeloCarro2 = carro1.marca + " " + carro2.modelo;  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Já podemos perceber avanços, nossas variáveis agora são relativas a um carro em específico.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.velocidade = carro1.velocidade + 10;  
        carro2.velocidade = 50;  
  
        String marcaModeloCarro1 = carro1.marca + " " + carro1.modelo;  
        String marcaModeloCarro2 = carro1.marca + " " + carro2.modelo;  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + marcaModeloCarro1 + " está mais rápido que o " + marcaModeloCarro2 + ".");  
        } else {  
            System.out.print("O " + marcaModeloCarro2 + " está mais rápido que o " + marcaModeloCarro1 + ".");  
        }  
    }  
}
```

Porém, a velocidade continua diferente para cada carro e a descrição continua duplicada para os dois.

```
public class Carro {  
  
    String marca;  
  
    String modelo;  
  
    Integer velocidade;  
  
    public String getDescricao() {  
        return marca + " " + modelo;  
    }  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {  
        velocidade = velocidade + porQuantosQuilometrosPorHora;  
    }  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {  
        velocidade = velocidade - porQuantosQuilometrosPorHora;  
    }  
}
```

Alteramos nossa classe Carro para conter 3 novas funções:

Recuperar descrição, acelerar e frear.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.marca = "Honda";  
        carro2.marca = "Volkswagen";  
  
        carro1.modelo = "City";  
        carro2.modelo = "Golf";  
  
        carro1.velocidade = 0;  
        carro2.velocidade = 0;  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.velocidade > carro2.velocidade){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

Quase lá, a situação atual ainda não impede que o programador altere a velocidade diretamente.

Encapsulamento

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade;  
  
    public String getDescricao() { ... }  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ... }  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ... }  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String novoModelo) {  
        modelo = novoModelo;  
    }  
  
    public Integer getVelocidade() {  
        return velocidade;  
    }  
}
```

Vamos fazer algumas alterações em nossa classe...

```
public class Carro {  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade;  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String novoModelo) {  
        modelo = novoModelo;  
    }  
  
    public Integer getVelocidade() {  
        return velocidade;  
    }  
}
```

Primeiro, deixaremos nossas **propriedades privadas**, através da palavra-chave **private**.

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade;  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String novoModelo) {  
        modelo = novoModelo;  
    }  
  
    public Integer getVelocidade() {  
        return velocidade;  
    }  
}
```

E em seguida, faremos
com que seus **valores**
sejam acessíveis
apenas através de
funções!

Isso chama-se
encapsulamento.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.setMarca("Honda");  
        carro2.setMarca("Volkswagen");  
  
        carro1.setModelo("City");  
        carro2.setModelo("Golf");  
  
        carro1.setVelocidade(0);  
        carro2.setVelocidade(0);  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

Vamos agora alterar nosso código para lidar com as propriedades encapsuladas.

```
public class App {
```

```
    Run | Debug
```

```
    public static void main(String[] args) throws Exception {
```

```
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();
```

```
        carro1.setMarca("Honda");
```

```
        carro2.setMarca("Volkswagen");
```

```
        carro1.setModelo("City");
```

```
        carro2.setModelo("Golf");
```

```
        carro1.setVelocidade(0);
```

```
        carro2.setVelocidade(0);
```

```
        carro1.acelera(10);
```

```
        carro2.acelera(50);
```

```
        if(carro1.getVelocidade() > carro2.getVelocidade()) {
```

```
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");
```

```
        } else {
```

```
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");
```

```
    }
```

```
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        carro1.setMarca("Honda");  
        carro2.setMarca("Volkswagen");  
  
        carro1.setModelo("City");  
        carro2.setModelo("Golf");  
  
        carro1.setVelocidade(0);  
        carro2.setVelocidade(0);  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido.");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido.");  
        }  
    }  
}
```

Opa, nosso código tem um erro!

Temos funções para acelerar e frear o carro mas não podemos definir sua velocidade inicial.

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    public String getDescricao() { ...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String novoModelo) {  
        modelo = novoModelo;  
    }  
  
    public Integer getVelocidade() {  
        return velocidade;  
    }  
}
```

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
private Integer velocidade = 0;  
  
    public String getDescricao() {...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String novoModelo) {  
        modelo = novoModelo;  
    }  
  
    public Integer getVelocidade() {  
        return velocidade;  
    }  
}
```

Nós podemos definir um **valor inicial** para nossa velocidade.

Afinal, todos os carros começam parados! 😊

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        // carro1.setMarca("Honda");  
        carro2.setMarca("Volkswagen");  
  
        carro1.setModelo("City");  
        carro2.setModelo("Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro();  
        Carro carro2 = new Carro();  
  
        // carro1.setMarca("Honda");  
        carro2.setMarca("Volkswagen");  
  
        carro1.setModelo("City");  
        carro2.setModelo("Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido.");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido.");  
        }  
    }  
}
```

Ok, todos os carros começam com velocidade zero.

Mas e se o programador esquecer de definir a marca?

Construtores

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {  
        marca = marcaDoCarro;  
        modelo = modeloDoCarro;  
    }  
  
    public String getDescricao() { ...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public String getMarca() { ...  
  
    public void setMarca(String novaMarca) { ...  
  
    public String getModelo() { ...  
  
    public void setModelo(String novoModelo) { ...  
  
    public Integer getVelocidade() { ...
```

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {  
        marca = marcaDoCarro;  
        modelo = modeloDoCarro;  
    }  
  
    public String getDescricao() { ...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ...  
  
    public String getMarca() { ...  
  
    public void setMarca(String novaMarca) { ...  
  
    public String getModelo() { ...  
  
    public void setModelo(String novoModelo) { ...  
  
    public Integer getVelocidade() { ...
```

Definindo um construtor, podemos forçar o desenvolvedor a informar propriedades obrigatórias.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro("Honda", "City");  
        Carro carro2 = new Carro("Volkswagen", "Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

Nosso código está bem mais limpo e organizado!

Verificando as peças do nosso carro

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro("Honda", "City");  
        Carro carro2 = new Carro("Volkswagen", "Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
}
```

Que tal indicarmos se certas peças do Carro precisam de reparo?

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {...  
  
        public Boolean isPneusBons(){  
            return pneusBons;  
        }  
  
        public void setPneusBons(Boolean pneusEstaoBons){  
            pneusBons = pneusEstaoBons;  
        }  
  
        public Boolean isPortasTrancando() {  
            return portasTrancando;  
        }  
  
        public void setPortasTrancando(Boolean portasEstaoTrancando){  
            portasTrancando = portasEstaoTrancando;  
        }  
    }
```

Já sabemos fazer isso, basta adicionar mais duas **propriedades** e deixá-las encapsuladas.

Assumimos que todo Carro começa com boas peças, então as iniciamos como "boas".

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro("Honda", "City");  
        Carro carro2 = new Carro("Volkswagen", "Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
  
        if(carro1.isPneusBons()){  
            System.out.println("Os pneus do carro " + carro1.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus do carro " + carro1.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro2.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro2.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro2.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Run | Debug

```
public static void main(String[] args) throws Exception {
```

```
    Carro carro1 = new Carro("Ford", "City");
    Carro carro2 = new Carro("Volkswagen", "Gol G5");
```

Com essas novas propriedades, podemos validar se as peças estão em boas condições!

```
carro1.acelera(10);
```

```
carro2.acelera(50);
```

```
if(carro1.getVelocidade() > carro2.getVelocidade()){
```

```
    System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");
```

```
} else {
```

```
    System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");
```

```
}
```

```
if(carro1.isPneusBons()) {
```

```
    System.out.println("Os pneus do carro " + carro1.getDescricao() + " estão em bom estado!");
```

```
} else {
```

```
    System.out.println("Cuidado, os pneus do carro " + carro1.getDescricao() + " precisam ser trocados!");
```

```
}
```

```
if(carro2.isPortasTrancando()) {
```

```
    System.out.println("As portas do carro " + carro2.getDescricao() + " estão trancando corretamente!");
```

```
} else {
```

```
    System.out.println("Cuidado, as portas do carro " + carro2.getDescricao() + " não estão trancando!");
```

```
}
```

```
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro1 = new Carro("Honda", "City");  
        Carro carro2 = new Carro("Volkswagen", "Golf");  
  
        carro1.acelera(10);  
        carro2.acelera(50);  
  
        if(carro1.getVelocidade() > carro2.getVelocidade()){  
            System.out.print("O " + carro1.getDescricao() + " está mais rápido que o " + carro2.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro2.getDescricao() + " está mais rápido que o " + carro1.getDescricao() + ".");  
        }  
    }  
  
    if(carro1.temPneusBomEstado()) {  
        System.out.println("Os pneus do carro " + carro1.getDescricao() + " estão em bom estado!");  
    } else {  
        System.out.println("Cuidado, os pneus do carro " + carro1.getDescricao() + " precisam ser trocados!");  
    }  
  
    if(carro2.isPortasTrancando()) {  
        System.out.println("As portas do carro " + carro2.getDescricao() + " estão trancando corretamente!");  
    } else {  
        System.out.println("Cuidado, as portas do carro " + carro2.getDescricao() + " não estão trancando!");  
    }  
}
```

Mas e se agora quisermos comparar um carro
e uma moto?

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Carro moto = new Carro("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(carro.isPneusBons()){  
            System.out.println("Os pneus do carro " + carro.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus do carro " + carro.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(moto.isPortasTrancando()){  
            System.out.println("As portas do carro " + moto.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + moto.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Carro moto = new Carro("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(carro.isPneusBons()){  
            System.out.println("Os pneus do carro " + carro.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus do carro " + carro.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(moto.isPortasTrancando()){  
            System.out.println("As portas do carro " + moto.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + moto.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Ok, renomeamos a variável "carro2" para "moto" e mudamos a marca e o modelo. Mas temos mesmo uma moto?

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Carro moto = new Carro("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao());  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao());  
        }  
  
        if(carro.isPneusBons()){  
            System.out.println("Os pneus do carro " + carro.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus do carro " + carro.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(moto.isPortasTrancando()){  
            System.out.println("As portas do carro " + moto.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + moto.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Não só estamos criando um objeto do tipo "Carro", como também estamos validando se as portas da moto estão trancando!? 😅

```
public class Moto {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Moto(String marcaDaMoto, String modeloDaMoto) {...  
  
    public Boolean isPneusBons(){...  
  
    public void setPneusBons(Boolean pneusEstaoBons){...  
  
    public String getDescricao() {...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public String getMarca() {...  
  
    public void setMarca(String novaMarca) {...  
  
    public String getModelo() {...  
  
    public void setModelo(String novoModelo) {...  
  
    public Integer getVelocidade() {...  
}
```

Podemos repetir o processo e criar uma classe para representar nossa moto...

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Agora estamos
utilizando a classe
correta e criando um
objeto do tipo Moto!

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {...  
  
    public Boolean isPneusBons(){...  
  
    public void setPneusBons(Boolean pneusEstaoBons){...  
  
    public Boolean isPortasTrancando(){...  
  
    public void setPortasTrancando(Boolean portasEstaoTrancando){...  
  
    public String getDescricao(){...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public String getMarca() {...  
  
    public void setMarca(String novaMarca) {...  
  
    public String getModelo() {...  
  
    public void setModelo(String novoModelo) {...  
  
    public Integer getVelocidade() {...  
}
```

```
1  public class Moto {  
2  
3      private String marca;  
4  
5      private String modelo;  
6  
7      private Integer velocidade = 0;  
8  
9      private Boolean pneusBons = true;  
10 >     public Moto(String marcaDaMoto, String modeloDaMoto) {...  
15 >     public Boolean isPneusBons(){...  
19 >     public void setPneusBons(Boolean pneusEstaoBons){...  
23 >     public String getDescricao() {...  
27 >     public void acelera(Integer porQuantosQuilometrosPorHora) {...  
32 >     public void desacelera(Integer porQuantosQuilometrosPorHora) {...  
35 >     public String getMarca() {...  
39 >     public void setMarca(String novaMarca) {...  
43 >     public String getModelo() {...  
47 >     public void setModelo(String novoModelo) {...  
51 >     public Integer getVelocidade() {...  
55 >
```

Observem agora nossas classes Carro e Moto. Perceberam a quantidade de código duplicado?

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {...}  
  
    public Boolean isPneusBons(){...}  
  
    public void setPneusBons(Boolean pneusEstaoBons){...}  
  
    public Boolean isPortasTrancando() {...}  
  
    public void setPortasTrancando(Boolean portasEstaoTrancando){...}  
  
    public String getDescricao() {...}  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...}  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...}  
  
    public String getMarca() {...}  
  
    public void setMarca(String novaMarca) {...}  
  
    public String getModelo() {...}  
  
    public void setModelo(String novoModelo) {...}  
  
    public Integer getVelocidade() {...}  
}
```

```
1  public class Moto {  
2  
3      private String marca;  
4  
5      private String modelo;  
6  
7      private Integer velocidade = 0;  
8  
9      private Boolean pneusBons = true;  
10 >     public Moto(String marcaDaMoto, String modeloDaMoto) {...}  
15  
16 >     public Boolean isPneusBons(){...}  
19  
20 >     public void setPneusBons(Boolean pneusEstaoBons){...}  
23  
24 >     public String getDescricao() {...}  
27  
28 >     public void acelera(Integer porQuantosQuilometrosPorHora) {...}  
31  
32 >     public void desacelera(Integer porQuantosQuilometrosPorHora) {...}  
35  
36 >     public String getMarca() {...}  
39  
40 >     public void setMarca(String novaMarca) {...}  
43  
44 >     public String getModelo() {...}  
47  
48 >     public void setModelo(String novoModelo) {...}  
51  
52 >     public Integer getVelocidade() {...}  
55 >
```

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) { ... }  
  
    public Boolean isPneusBons() { ... }  
  
    public void setPneusBons(Boolean pneusEstaoBons) { ... }  
  
    public Boolean isPortasTrancando() { ... }  
    public void setPortasTrancando(Boolean portasEstaoTrancando) { ... }  
  
    public String getDescricao() { ... }  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) { ... }  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) { ... }  
  
    public String getMarca() { ... }  
  
    public void setMarca(String novaMarca) { ... }  
  
    public String getModelo() { ... }  
  
    public void setModelo(String novoModelo) { ... }  
  
    public Integer getVelocidade() { ... }  
}
```

```
1  public class Moto {  
2  
3      private String marca;  
4  
5      private String modelo;  
6  
7      private Integer velocidade = 0;  
8  
9      private Boolean pneusBons = true;  
10     > public Moto(String marcaDaMoto, String modeloDaMoto) { ... }  
11     > public Boolean isPneusBons() { ... }  
12     > public void setPneusBons(Boolean pneusEstaoBons) { ... }  
13     > public String getDescricao() { ... }  
14     > public void acelera(Integer porQuantosQuilometrosPorHora) { ... }  
15     > public void desacelera(Integer porQuantosQuilometrosPorHora) { ... }  
16     > public String getMarca() { ... }  
17     > public void setMarca(String novaMarca) { ... }  
18     > public String getModelo() { ... }  
19     > public void setModelo(String novoModelo) { ... }  
20     > public Integer getVelocidade() { ... }  
21  
22     > public void freia(Integer porQuantosQuilometrosPorHora) { ... }  
23     > public void giraEsquerda() { ... }  
24     > public void giraDireita() { ... }  
25     > public void apertaFreio() { ... }  
26     > public void soltaFreio() { ... }  
27     > public void ligaSinal() { ... }  
28     > public void desligaSinal() { ... }  
29     > public void abrePorta() { ... }  
30     > public void fechaPorta() { ... }  
31     > public void abrePortaDianteira() { ... }  
32     > public void fechaPortaDianteira() { ... }  
33     > public void abrePortaTraseira() { ... }  
34     > public void fechaPortaTraseira() { ... }  
35     > public void abrePortaPassageiro() { ... }  
36     > public void fechaPortaPassageiro() { ... }  
37     > public void abrePortaMotorista() { ... }  
38     > public void fechaPortaMotorista() { ... }  
39     > public void abrePortaCarga() { ... }  
40     > public void fechaPortaCarga() { ... }  
41     > public void abrePortaReservorio() { ... }  
42     > public void fechaPortaReservorio() { ... }  
43     > public void abrePortaArCondicionado() { ... }  
44     > public void fechaPortaArCondicionado() { ... }  
45     > public void abrePortaAssentoMotorista() { ... }  
46     > public void fechaPortaAssentoMotorista() { ... }  
47     > public void abrePortaAssentoPassageiro() { ... }  
48     > public void fechaPortaAssentoPassageiro() { ... }  
49     > public void abrePortaAssentoCrianca() { ... }  
50     > public void fechaPortaAssentoCrianca() { ... }  
51     > public void abrePortaAssentoCachorro() { ... }  
52     > public void fechaPortaAssentoCachorro() { ... }  
53     > public void abrePortaAssentoPasseio() { ... }  
54     > public void fechaPortaAssentoPasseio() { ... }  
55     > public void abrePortaAssentoCadeira() { ... }
```

Isso pode ser otimizado através da Herança!

Herança

```
public class Carro {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {...}  
  
    public Boolean isPneusBons(){...}  
  
    public void setPneusBons(Boolean pneusEstaoBons){...}  
  
    public Boolean isPortasTrancando() {...}  
  
    public void setPortasTrancando(Boolean portasTrancando){...}  
  
    public String getDescricao() {...}  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...}  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...}  
  
    public String getMarca() {...}  
  
    public void setMarca(String novaMarca) {...}  
  
    public String getModelo() {...}  
  
    public void setModelo(String novoModelo) {...}  
  
    public Integer getVelocidade() {...}  
}
```

```
1  public class Moto {  
2  
3      private String marca;  
4  
5      private String modelo;  
6  
7      private Integer velocidade = 0;  
8  
9      private Boolean pneusBons = true;  
10 >     public Moto(String marcaDaMoto, String modeloDaMoto) {...}  
15 >     public Boolean isPneusBons(){...}  
19 >     public void setPneusBons(Boolean pneusEstaoBons){...}  
23 >     public String getDescricao() {...}  
27 >     public void acelera(Integer porQuantosQuilometrosPorHora) {...}  
31 >     public void desacelera(Integer porQuantosQuilometrosPorHora) {...}  
35 >     public String getMarca() {...}  
39 >     public void setMarca(String novaMarca) {...}  
43 >     public String getModelo() {...}  
47 >     public void setModelo(String novoModelo) {...}  
51 >     public Integer getVelocidade() {...}  
55 }
```

Observando nossas classes, podemos perceber duas coisas:

1. Várias propriedades dizem respeito às duas classes;
2. Ambas são veículos motorizados;

```
public class Veiculo {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) {...  
  
    public Boolean isPneusBons(){...  
  
    public void setPneusBons(Boolean pneusEstaoBons){...  
  
    public String getDescricao() {...  
  
    public void acelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {...  
  
    public String getMarca() {...  
  
    public void setMarca(String novaMarca) {...  
  
    public String getModelo() {...  
  
    public void setModelo(String novoModelo) {...  
  
    public Integer getVelocidade() {...  
}
```

Vamos criar uma nova classe chamada Veiculo e colocar nela todas as propriedades e funções em comum entre as classes Carro e Moto.

```
public class Moto extends Veiculo {  
  
    public Moto(String marcaDaMoto, String modeloDaMoto) {  
        super(marcaDaMoto, modeloDaMoto);  
    }  
}
```

As classes Moto e Carro não precisam mais declarar as propriedades em comum, elas serão "herdadas" de Veiculo.

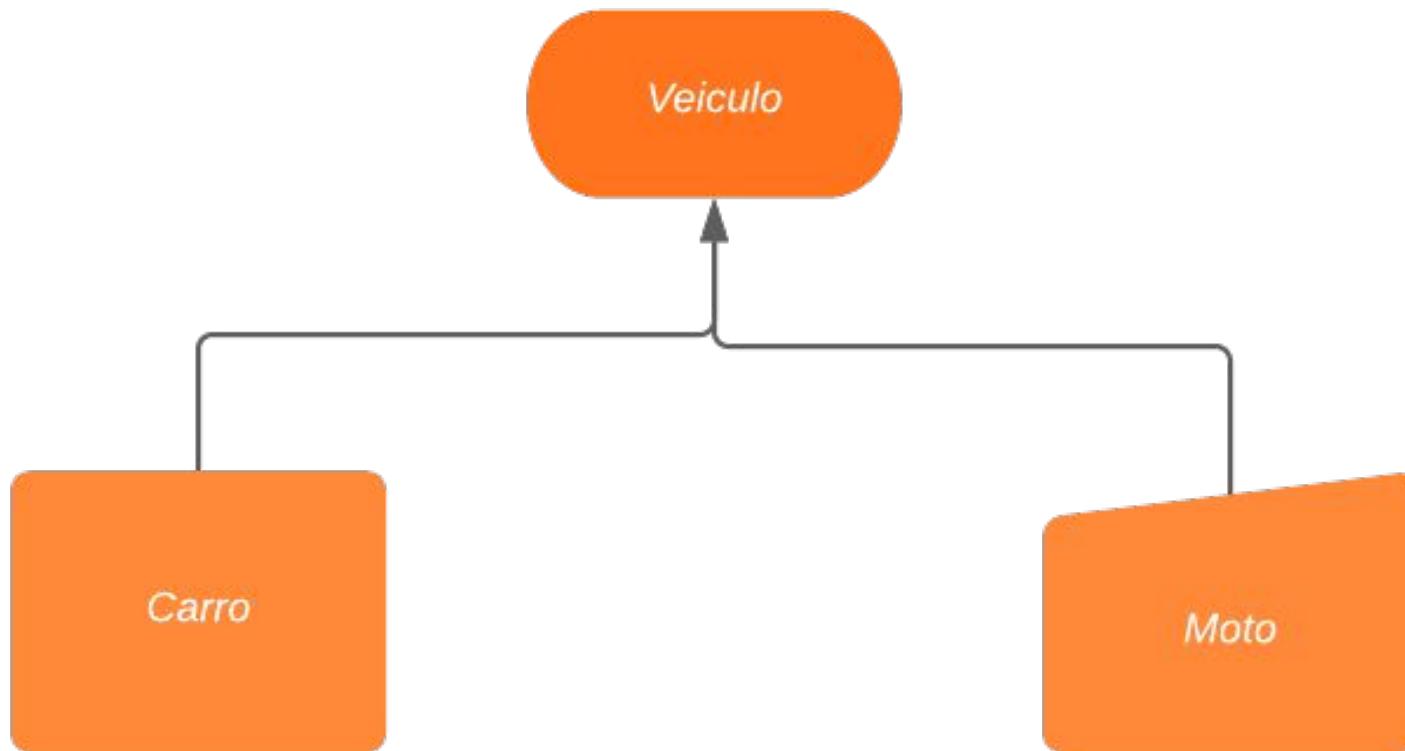
```
public class Moto extends Veiculo {  
    public Moto(String marcaDaMoto, String modeloDaMoto) {  
        super(marcaDaMoto, modeloDaMoto);  
    }  
}
```

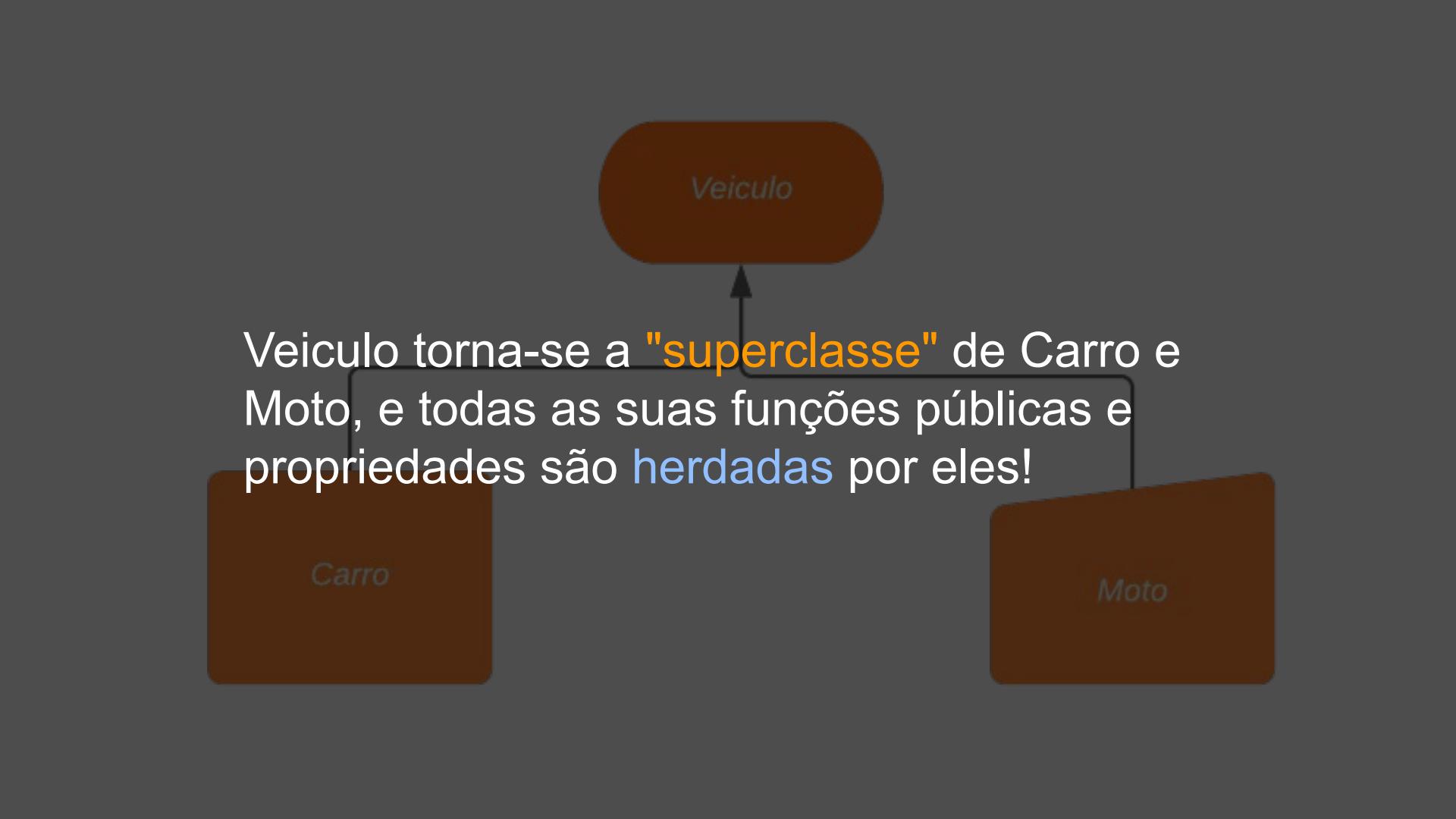
Isso acontece através da palavra chave **extends**, que indica que a **classe** Moto "extende" a **classe** Veiculo.

```
public class Carro extends Veiculo {  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {  
        super(marcaDoCarro, modeloDoCarro);  
    }  
  
    public Boolean isPortasTrancando() {  
        return portasTrancando;  
    }  
  
    public void setPortasTrancando(Boolean portasEstaoTrancando){  
        portasTrancando = portasEstaoTrancando;  
    }  
}
```

```
public class Carro extends Veiculo {  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {  
        super(marcaDoCarro, modeloDoCarro);  
    }  
  
    public Boolean isPortasTrancando() {  
        return portasTrancando;  
    }  
  
    public void setPortasTrancando(Boolean portasEstaoTrancando){  
        portasTrancando = portasEstaoTrancando;  
    }  
}
```

Fazemos a mesma coisa na classe Carro, apenas adicionando as propriedades e funções que dizem respeito apenas a esta classe.





Veiculo

Veiculo torna-se a "**superclasse**" de Carro e Moto, e todas as suas funções públicas e propriedades são **herdadas** por eles!

Carro

Moto

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBomEstado())  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando())  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

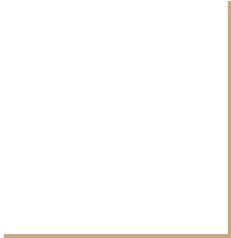
Percebam que nada precisa ser alterado em
nosso código original, ele nem percebeu que

as funções mudaram de lugar!

Dúvidas?



Descomplicando a programação com Java



Workshop - dia 2

Agenda

Dia 1

Java e Orientação a Objetos.

Dia 2

Neste dia criaremos uma aplicação simples com acompanhamento do instrutor.

Dia 3

Faremos a correção de alguns exercícios em tempo real e responderemos dúvidas ao vivo.

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Nosso código ontem estava desta forma:

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
  
        carro.acelera(10);  
        moto.acelera(50);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        Vamos adicionar um terceiro tipo de veículo,  
        uma bicicleta!  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
        Bicicleta bicicleta = new Bicicleta("Caloi", "MTB aro 29");  
  
        carro.acelera(10);  
        moto.acelera(50);  
        bicicleta.acelera(10);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em ótimo estado.");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " estão com problemas de desgaste.");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Sem problemas até aqui.

Porém uma dúvida:
qual a diferença entre
acelerar um carro e
uma bicicleta?

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
        Bicicleta bicicleta = new Bicicleta("Caloi", "MTB aro 29");  
  
        carro.acelera(10);  
        moto.acelera(50);  
        bicicleta.acelera(10);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Queremos que todos os veículos possam acelerar mas não sabemos como cada um deles deve ser acelerado!

Polimorfismo

Polimorfismo significa literalmente "várias formas". Em Java, expressamos essa característica através de métodos sobrecarregados.

```
public abstract class Veiculo {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) {...  
  
    public Boolean isPneusBons(){...  
  
    public void setPneusBons(Boolean pneusEstaoBons){...  
  
    public String getDescricao() {...  
  
    public abstract void acelera(Integer porQuantosQuilometrosPorHora);  
  
    public abstract void desacelera(Integer porQuantosQuilometrosPorHora);  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

Para começar,
precisamos alterar
nossa classe Veículo,
para torná-la abstrata.

```
public abstract class Veiculo {  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) {...}  
  
    public Boolean isPneusBons() {...}  
  
    public void setPneusBons(Boolean pneusEstaoBons){ ...}  
  
    public String getDescricao() {...}  
  
    public abstract void acelera(Integer porQuantosQuilometrosPorHora);  
  
    public abstract void desacelera(Integer porQuantosQuilometrosPorHora);  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

Isso é feito através da palavra-chave **abstract**.

Classes abstratas definem características e não podem ser instanciadas.

```
public abstract class Veiculo {  
  
    private String marca;  
  
    private String modelo;  
  
    private Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) {...  
  
    public Boolean isPneusBons() {...  
  
    public void setPneusBons(Boolean pneusEstaoBons){ ...  
  
    public String getDescricao() {...  
  
    public abstract void acelera(Integer porQuantosQuilometrosPorHora);  
  
    public abstract void desacelera(Integer porQuantosQuilometrosPorHora);  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

Com nossa classe definida como abstrata, podemos escolher métodos para serem abstratos também.

Isso forçará que as classes que herdem desta criem uma implementação para estes métodos.

```
public class Bicicleta extends Veiculo {  
  
    public Bicicleta(String marca, String modelo) {  
        super(marca, modelo);  
    }  
  
    @Override  
    public void acelera(Integer porQuantosQuilometrosPorHora) {  
        // TODO Auto-generated method stub  
    }  
  
    @Override  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {  
        // TODO Auto-generated method stub  
    }  
}
```

Agora precisamos implementar os métodos nas classes Bicicleta e Veiculo.

Notem a anotação **@Override**, que indica que sobrescrevemos um método da superclasse.

```
public class Bicicleta extends Veiculo {  
  
    public Bicicleta(String marca, String modelo) {  
        super(marca, modelo);  
    }  
  
    @Override  
    public void acelera(Integer porQuantosQuilometrosPorHora) {  
        this.velocidade = velocidade + porQuantosQuilometrosPorHora;  
    }  
  
    @Override  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {  
        velocidade = velocidade - porQuantosQuilometrosPorHora;  
    }  
}
```

Porém, as propriedades de Veículo são inacessíveis diretamente pelas subclasses.

Isso acontece por termos definido seu modificador de acesso como "private" (privado).

```
public abstract class Veiculo {  
  
    private String marca;  
  
    private String modelo;  
  
    protected Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) {...  
  
    public Boolean isPneusBons() {...  
  
    public void setPneusBons(Boolean pneusEstaoBons){ ...  
  
    public String getDescricao() {...  
  
    public abstract void acelera(Integer porQuantosQuilometrosPorHora);  
  
    public abstract void desacelera(Integer porQuantosQuilometrosPorHora);  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

Uma das alternativas que temos é diminuir as restrições de acesso, usando outro modificador.

O escolhido foi o "**protected**", que deixa acessível por classes no mesmo pacote ou por subclasses.

```
public abstract class Veiculo {  
  
    private String marca;  
  
    private String modelo;  
  
    protected Integer velocidade = 0;  
  
    private Boolean pneusBons = true;  
  
    public Veiculo(String marcaDoVeiculo, String modeloDoVeiculo) { ... }  
    public Boolean isPneusBons() { ... }  
    public void setPneusBons(Boolean pneusEstaoBons) { ... }  
    public String getDescricao() { ... }  
    public abstract void acelerar(Integer porQuantosQuilometrosPorHora);  
    public abstract void freiar(Integer porQuantosQuilometrosPorHora);  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String novaMarca) {  
        marca = novaMarca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }
```

Atenção: O modificador de acesso pode ser adicionado às funções "getter e setter" ao invés da propriedade, mantendo o encapsulamento e melhorando ainda mais nosso código.

```
public class Carro extends Veiculo {  
  
    private Boolean portasTrancando = true;  
  
    public Carro(String marcaDoCarro, String modeloDoCarro) {  
        super(marcaDoCarro, modeloDoCarro);  
    }  
  
    public Boolean isPortasTrancando() {  
        return portasTrancando;  
    }  
  
    public void setPortasTrancando(Boolean portasEstaoTrancando){  
        portasTrancando = portasEstaoTrancando;  
    }  
  
    @Override  
    public void acelera(Integer porQuantosQuilometrosPorHora) {  
        System.out.println("Queimando combustível e aumentando a velocidade...");  
  
        this.velocidade = velocidade + porQuantosQuilometrosPorHora;  
    }  
  
    @Override  
    public void desacelera(Integer porQuantosQuilometrosPorHora) {  
        System.out.println("Reduzindo a queima de combustível e diminuindo a velocidade...");  
  
        velocidade = velocidade - porQuantosQuilometrosPorHora;  
    }  
}
```

Basta criarmos o mesmo método na classe Carro e estamos prontos!

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
        Bicicleta bicicleta = new Bicicleta("Caloi", "MTB aro 29");  
  
        carro.acelera(10);  
        moto.acelera(50);  
        bicicleta.acelera(10);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + moto.getDescricao() + " está mais rápido que o " + carro.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

```
public class App {  
    Run | Debug  
    public static void main(String[] args) throws Exception {  
        Carro carro = new Carro("Honda", "City");  
        Moto moto = new Moto("Harley-Davidson", "XLCR 1000");  
        Bicicleta bicicleta = new Bicicleta("Caloi", "MTB aro 29");  
  
        carro.acelera(10);  
        moto.acelera(50);  
        bicicleta.acelera(10);  
  
        if(carro.getVelocidade() > moto.getVelocidade()){  
            System.out.print("O " + carro.getDescricao() + " está mais rápido que o " + moto.getDescricao() + ".");  
        } else {  
            System.out.print("O " + carro.getDescricao() + " é mais lento que o " + moto.getDescricao() + ".");  
        }  
  
        if(moto.isPneusBons()){  
            System.out.println("Os pneus da moto " + moto.getDescricao() + " estão em bom estado!");  
        } else {  
            System.out.println("Cuidado, os pneus da moto " + moto.getDescricao() + " precisam ser trocados!");  
        }  
  
        if(carro.isPortasTrancando()){  
            System.out.println("As portas do carro " + carro.getDescricao() + " estão trancando corretamente!");  
        } else {  
            System.out.println("Cuidado, as portas do carro " + carro.getDescricao() + " não estão trancando!");  
        }  
    }  
}
```

Nosso código original está pronto e completamente funcional!

Sobre modificadores de acesso...

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Exercício

Exercício

Vamos incrementar nossa solução de Oficina, utilizando Orientação a Objeto.

Implemente o seguinte cenário segundo o que aprendemos no workshop:

- Temos dois tipos de Oficinas: A oficina especializada em bicicletas e a oficina especializada em carros e motos. A oficina deve ter um método chamado que faça o Orçamento dos veículos que ela pode trabalhar (dica: Classes também podem ser argumentos de funções);
- Crie um objeto chamado "Peça". Adicione as peças a seus veículos e as use para calcular o orçamento nas oficinas. Dica: as propriedades da peça podem ser:
 - Nome da peça;
 - Indicação se a peça precisa de reparo;
 - O valor do reparo da peça;