UNIVATES

<programacao_orientada_objetos/>

#Prof. Edson Funke

<documentação/>

JDK 8
http://docs.oracle.com/javase/8/docs/index.html

Tutorial
https://docs.oracle.com/javase/tutorial/essential/

<conteúdo/>

Modificadores de acesso

Os modificadores de acesso auxiliam na organização dos componentes da sua aplicação ao tornar os membros das classes mais ou menos acessíveis por outras partes do seu programa.

Em ordem do mais restritivo para o menos restritivo:

- private
- padrão (default)ou acessibilidade de pacote
- protected
- public

<Modificadores/>

São 4 os modificadores de acesso básicos da linguagem Java:

- **private**
- **default**
- **protected**
- **public**

<public/>

<public/>

O modificador de acesso **public** é o menos restritivo de todos.

Ele permite que qualquer outra parte da sua aplicação tenha acesso ao componente marcado como **public**.

<public/>

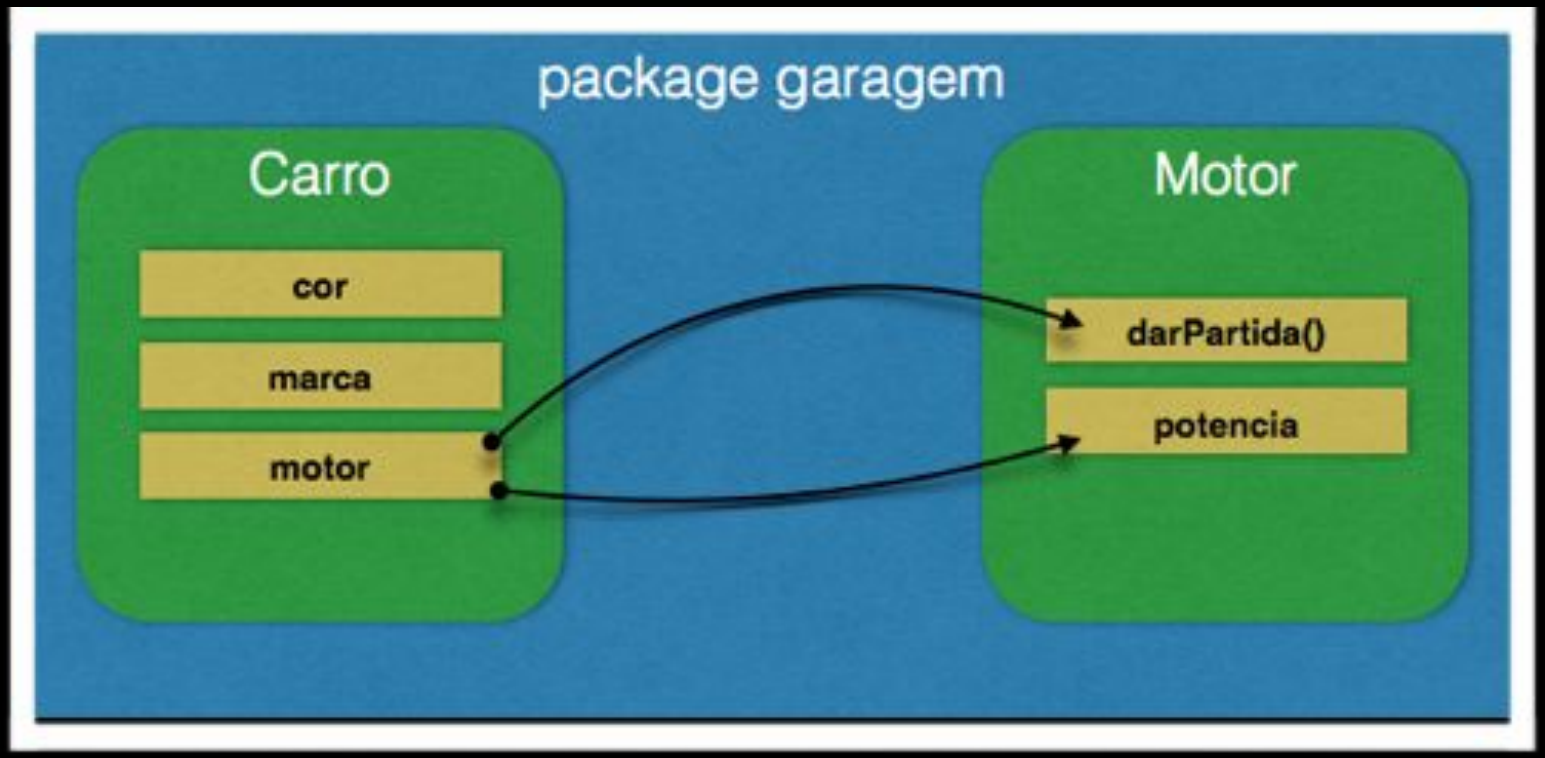
```
package garagem;
public class Carro {
    public String marca;
    public String cor;
    public Motor motor;

    public void ligar()
    {
        this.motor.darPartida();
    }

    public String toString()
    {
        return marca + " " + cor + "
" + motor.potencia;
    }
}
```

```
package garagem;
public class Motor {
    public int potencia;
    public void
darPartida(){}
}
```

<public/>

A classe **Carro** faz uso da classe **Motor**. Na classe **Carro** temos acesso à propriedade potencia e ao método darPartida() da classe **Motor**.

A classe **Carro** tem acesso a todos os membros marcados como **public** da classe **Motor** mesmo que as duas classes estejam em pacotes distintos.



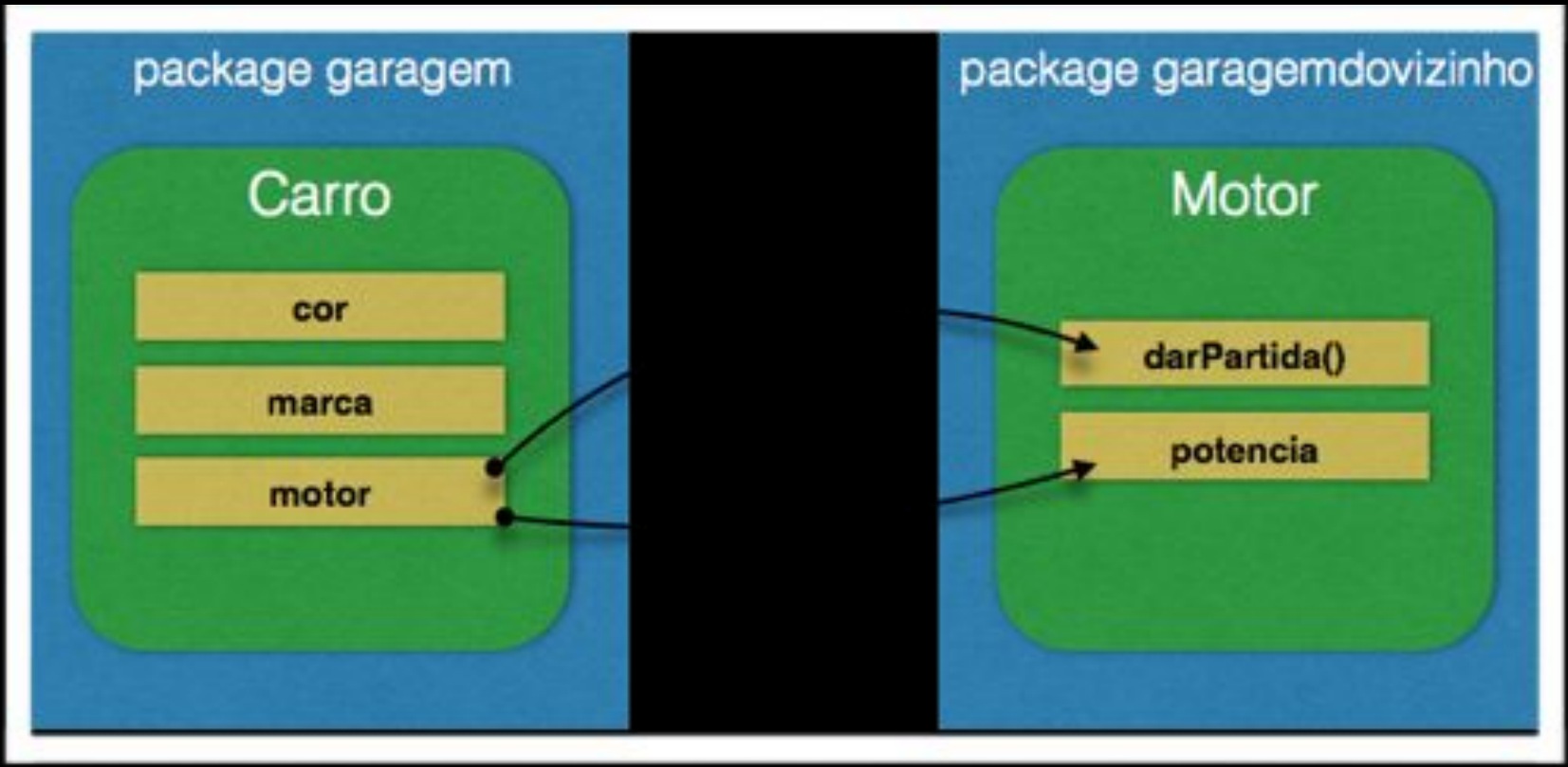
<public/>

```
package garagem;
import garagemdovizinho.Motor;
public class Carro {
    public String marca;
    public String cor;
    public Motor motor;

    public void ligar()
    {
        this.motor.darPartida();
    }

    public String toString()
    {
        return marca + " " + cor + "
" + motor.potencia;
    }
}
```

```
package
garagemdovizinho;
public class Motor {
    public int potencia;
    public void
darPartida(){}
}
```

<exemplo/>

Mesmo com a classe **Motor** dentro do um pacote diferente, a classe **Carro** continua conseguindo acessar o método **darPartida()** e a propriedade **potencia** da classe **Motor**.



<classe derivada/>

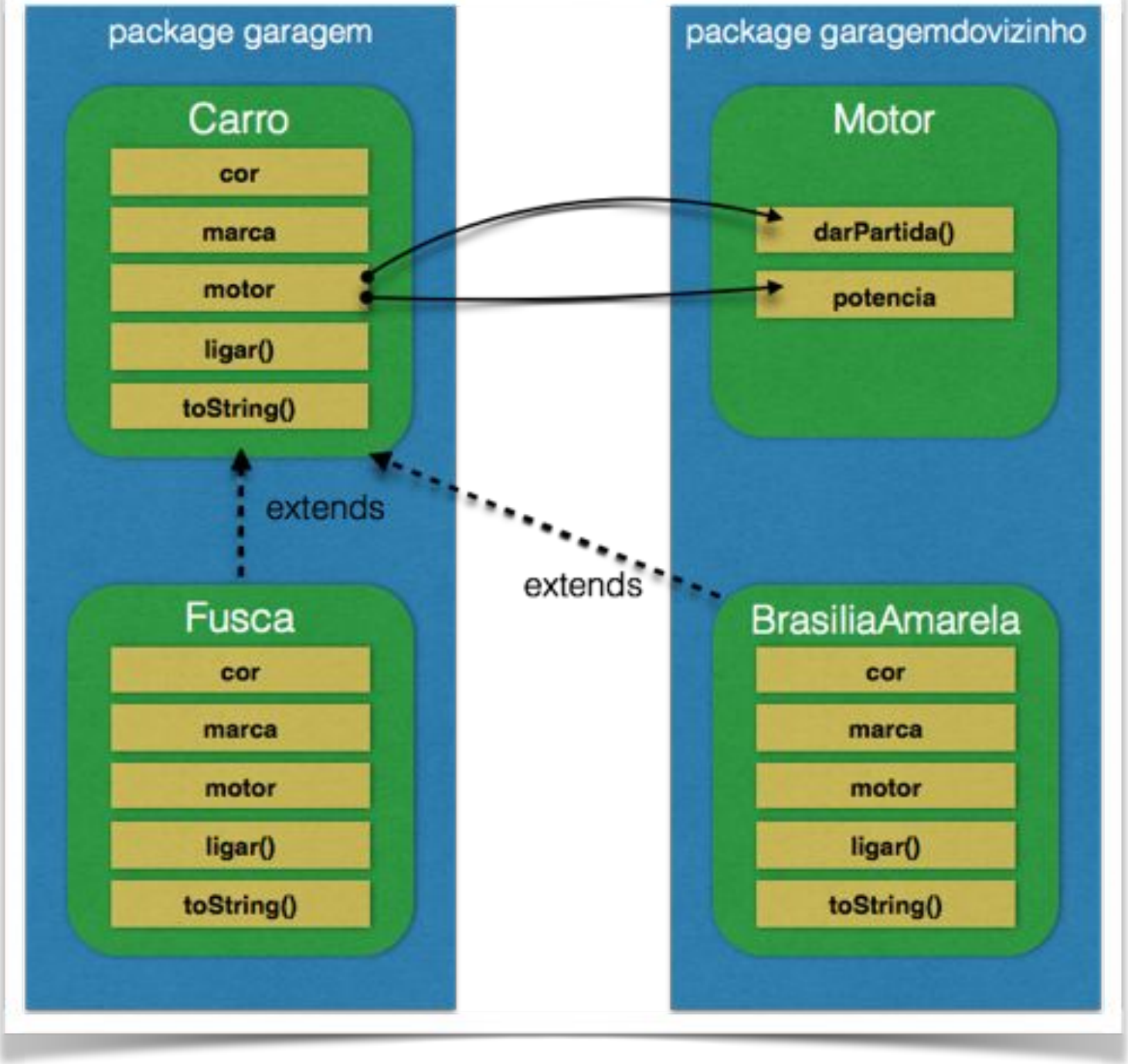
Membros marcados com o modificador de acesso public também são acessíveis por classes derivadas.

Declaração das classes:

Fusca no pacote **garagem**.

BrasiliaAmarela no pacote **garagemdovizinho**.

Ambas as classes são derivadas da classe **Carro**.

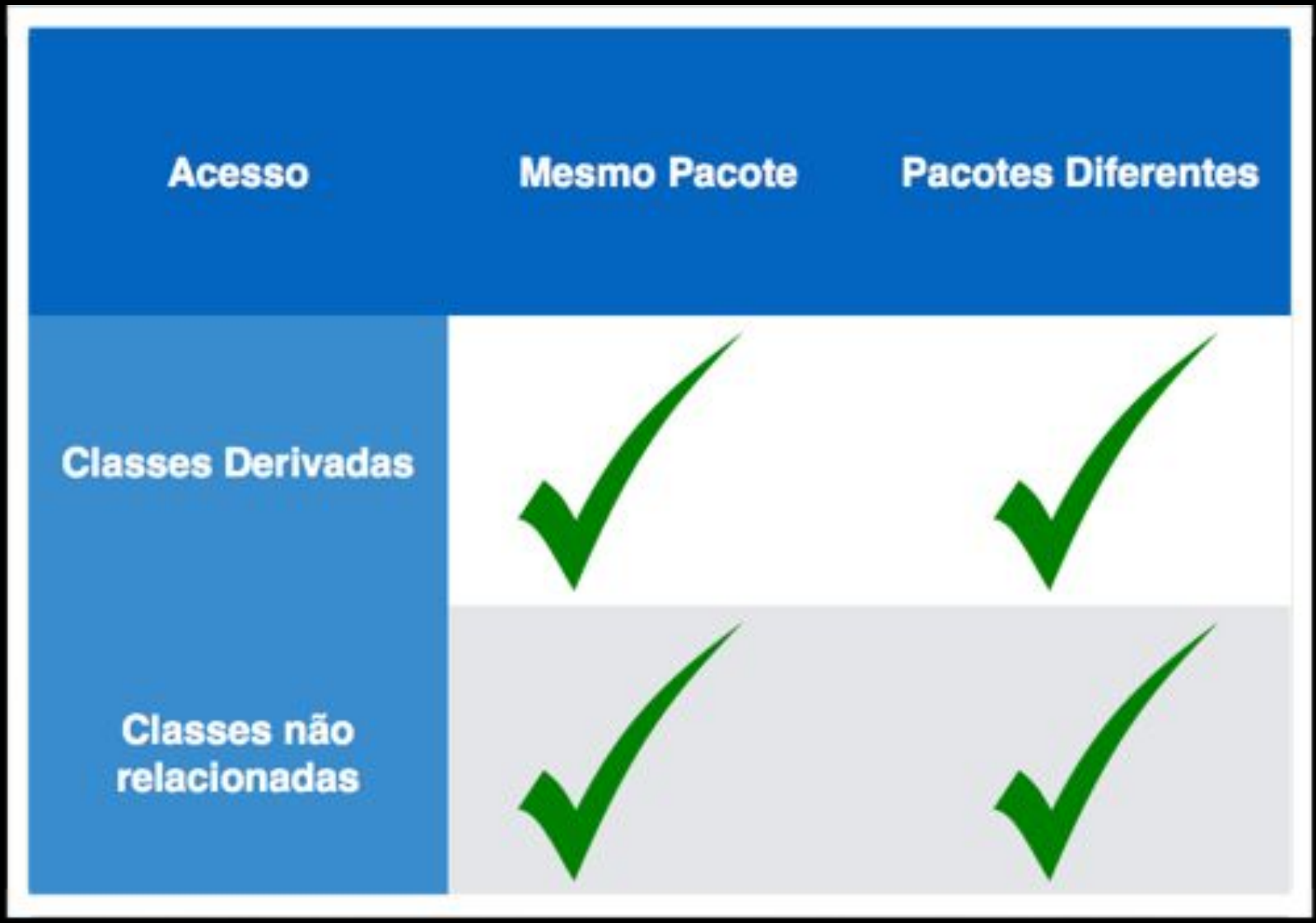

package garagem
Carro
cor
marca
motor
ligar()
toString()
extends
Fusca
cor
marca
motor
ligar()
toString()
package garagemdovizinho
Motor
darPartida()
potencia
extends
BrasiliaAmarela
cor
marca
motor
ligar()
toString()

<public - classes derivadas/>

```
package garagem;
public class Fusca extends Carro {

    public Fusca()
    {
        this.cor = "Branco";
        this.marca = "VW";
        this.ligar();
        this.toString();
    }
}
```

```
package
garagemdovizinho;
import garagem.Carro;

public class
BrasiliaAmarela extends
Carro {
    public
BrasiliaAmarela() {
        this.cor =
"Amarelo";
        this.marca =
"VW";
        this.ligar();
        this.toString();
    }
}
```

<acessos - public/>



Acesso	Mesmo Pacote	Pacotes Diferentes
Classes Derivadas	✓	✓
Classes não relacionadas	✓	✓

<protected/>

<protected/>

Os membros das classes marcados com o modificador de acesso **protected** serão acessíveis por classes e interfaces dentro do mesmo pacote e por classes derivadas mesmo que estejam em pacotes diferentes.

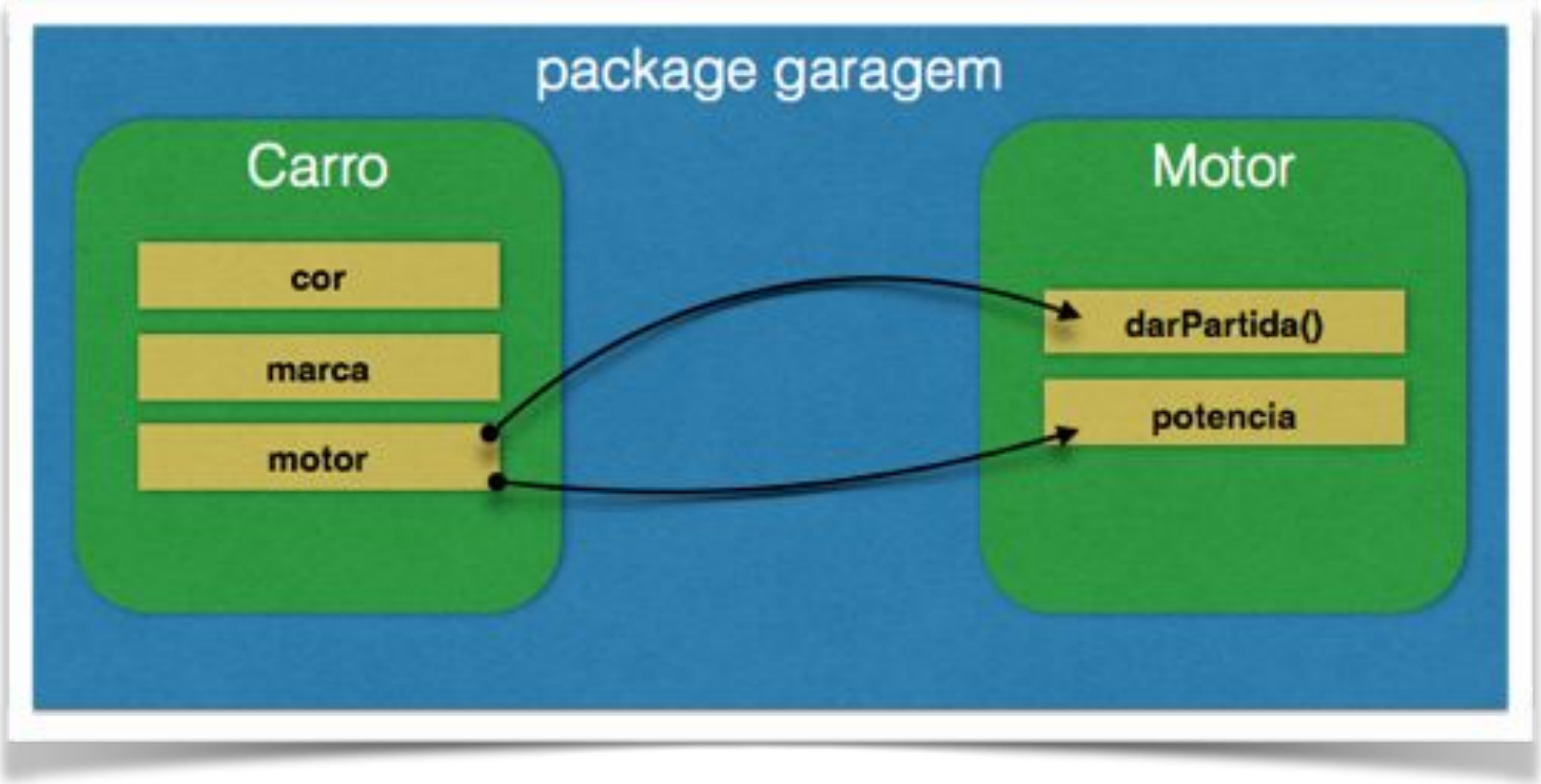
<protected/>

```
package garagem;
public class Motor {
    public int potencia;
    protected void
darPartida(){}
}
```

```
package garagem;
public class Carro {
    protected String marca;
    protected String cor;
    public Motor motor;
    protected void ligar()
    {
      this.motor.darPartida();
    }
    public String toString()
    {
      return marca + " " + cor
+ " " + motor.potencia;
    }
}
```

<protected/>

O método `darPartida()` da classe `Motor` é acessível pela classe `Carro`.



<protected/>

Vamos mover a classe **Motor** para o pacote **garagemdovizinho** e observar as mudanças.

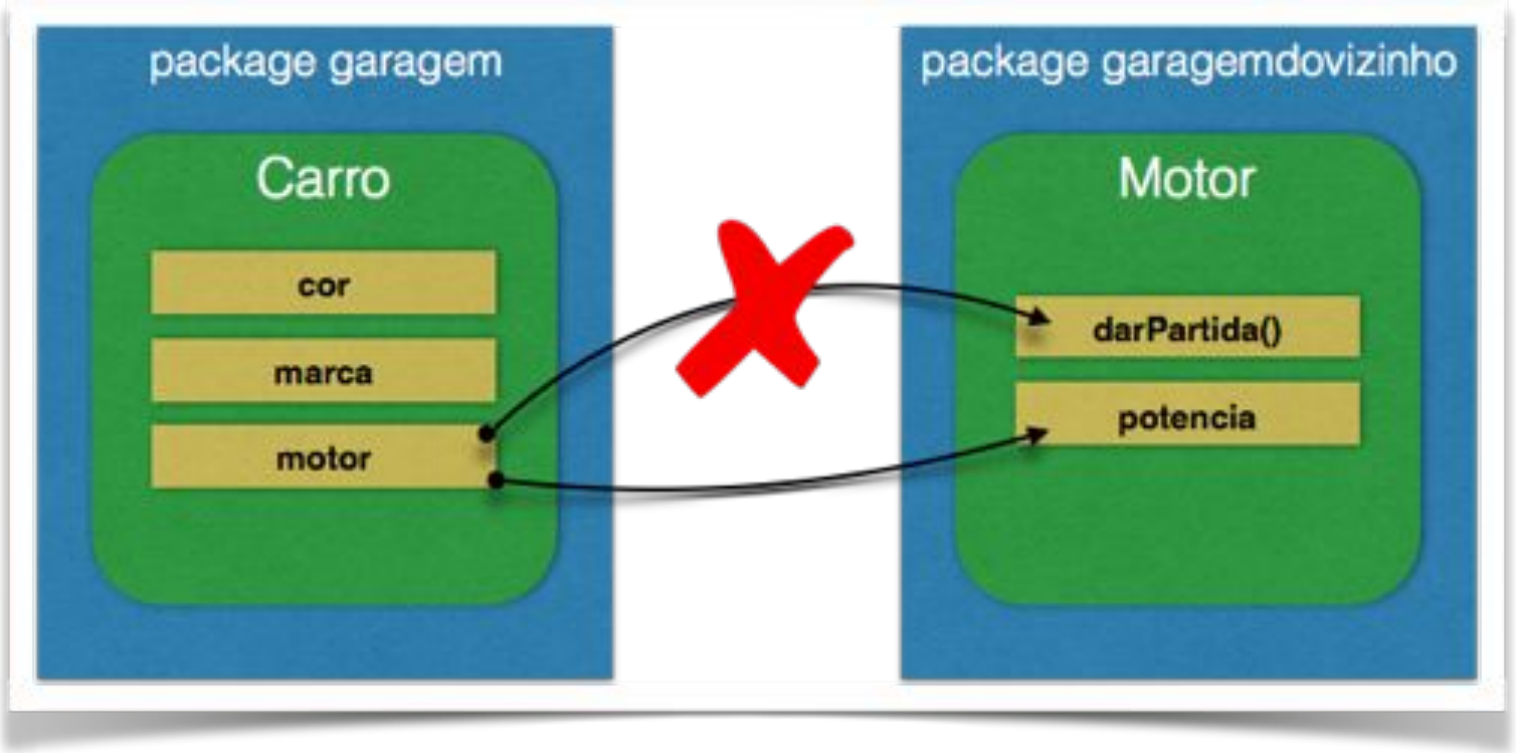
<protected/>

```
package garagemdovizinho;
public class Motor {
    public int potencia;
    protected void darPartida(){}
}
```

```
package garagem;
import garagemdovizinho.Motor;
public class Carro {
    protected String marca;
    protected String cor;
    public Motor motor;
    protected void ligar()
    {
this.motor.darPartida(); // O
método darPartida() do tipo
Motor não é visível.
    }
    public String toString()
    {
        return marca + " " +
cor + " " + motor.potencia;
    }
}
```

<protected/>

A classe Carro não compila e recebemos a mensagem de que o método darPartida() do tipo Motor não é visível.
Um membro marcado com o modificador de acesso protected só é visível para outras classes e interfaces localizadas dentro do mesmo pacote ou por seus herdeiros.



<classes derivadas/>

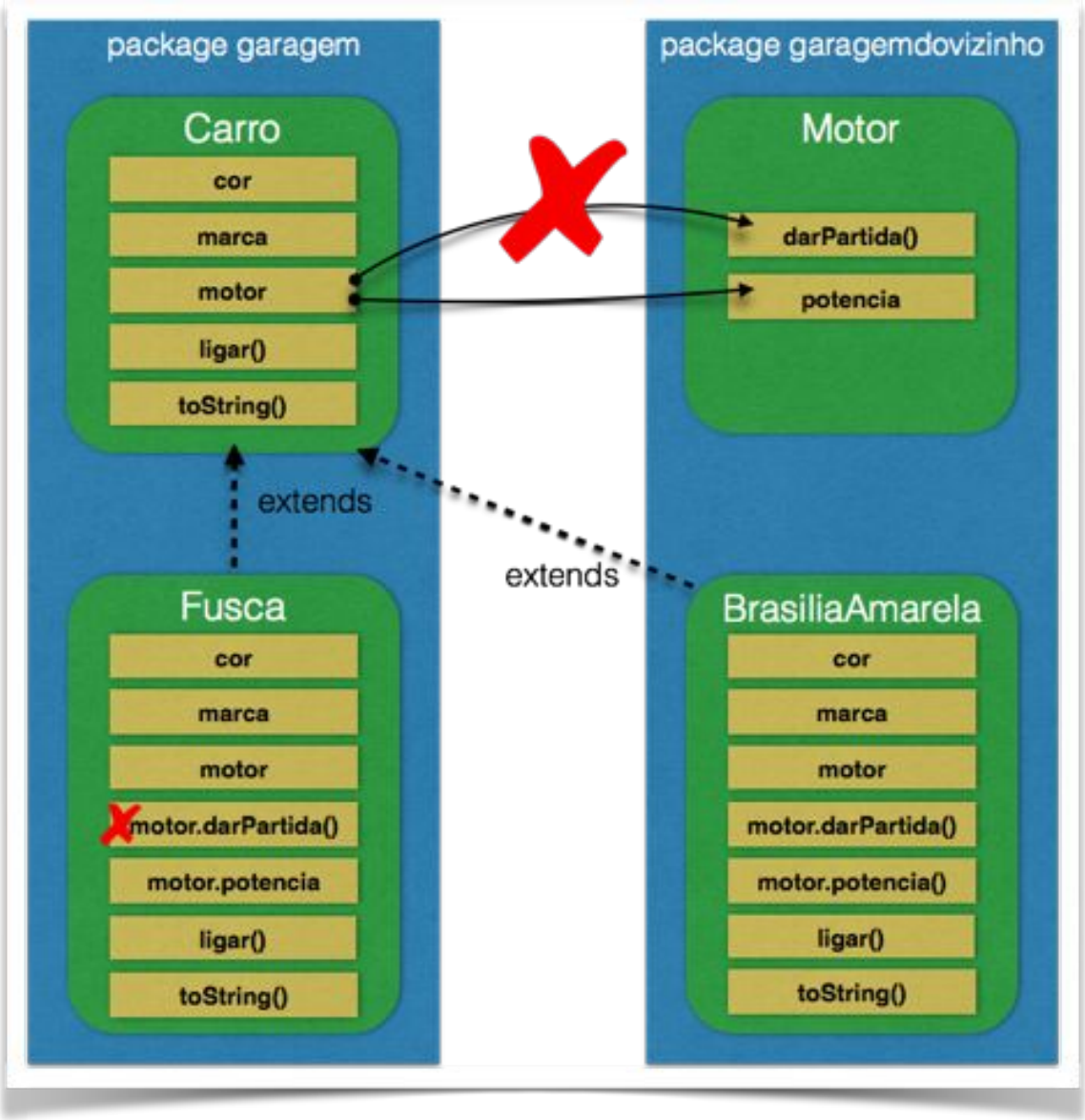
As classes derivadas têm acesso aos membros marcados como protected na sua classe mãe independentemente do pacote onde essas duas classes estejam localizadas.

<protected - classes derivadas/>

```
package garagem;
public class Fusca extends Carro {

    public Fusca()
    {
        this.cor = "Branco";
        this.marca = "VW";
        this.ligar();
        this.motor.darPartida(); // O
método darPartida() do tipo Motor não
é visível.
        this.motor.potencia = 100;
        this.toString();
    }
}
```

```
package garagemdovizinho;
import garagem.Carro;

public class
BrasiliaAmarela extends
Carro {

    public
BrasiliaAmarela() {
        this.cor =
"Branco";
        this.marca = "VW";
        this.ligar();

this.motor.darPartida();
        this.motor.potencia
= 100;
        this.toString();
    }
}
```

Apenas o método darPartida() da classe Motor não é acessível pela classe Fusca. Todos os membros das classes Carro e Motor são acessíveis pela classe BrasiliaAmarela.



<acessos - protected/>

O modificador de acesso **protected** é mais restritivo do que o modificador public, mas é menos restritivo do que os modificadores padrão e private.

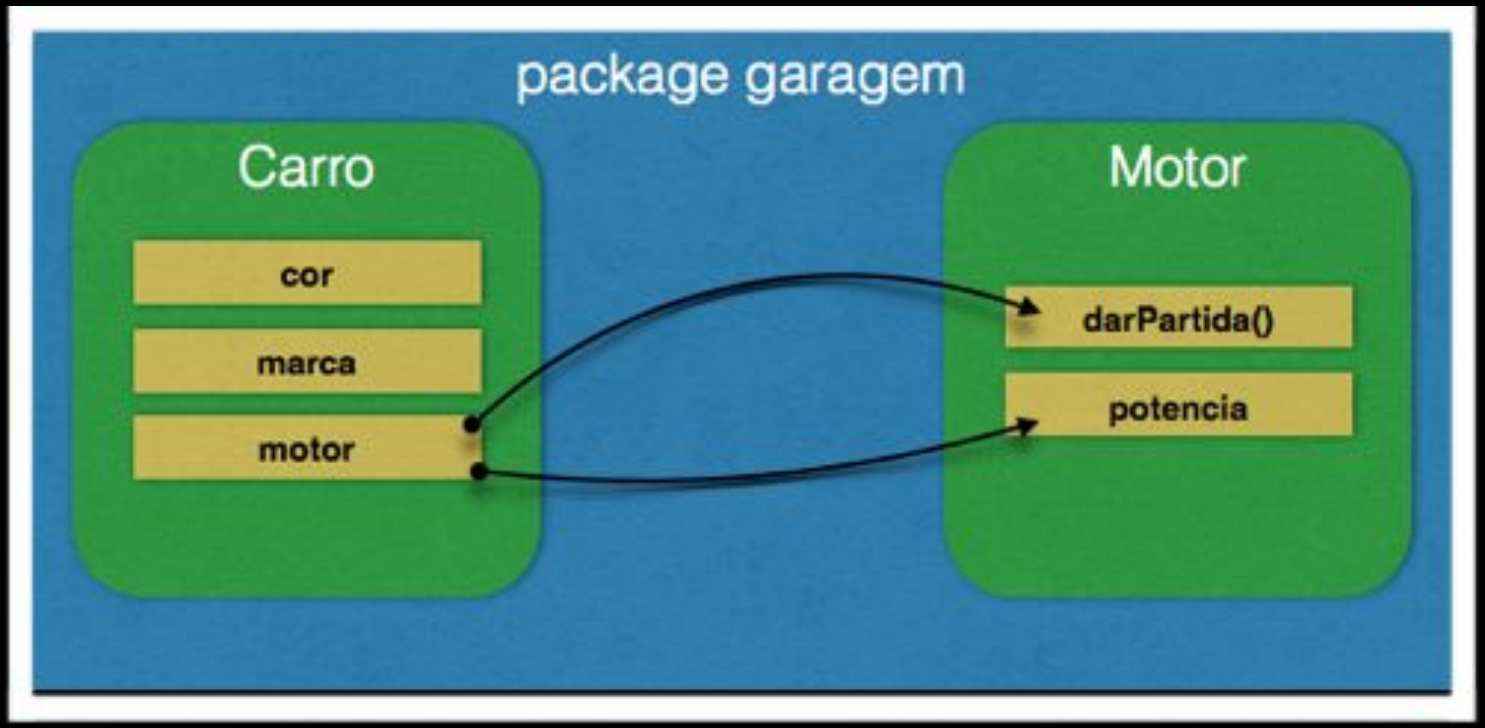
<padrão/>

default

<default/>

O modificador de acesso **padrão**, também conhecido como acessibilidade de pacote, é o modificador atribuído aos membros da classe que não foram marcados explicitamente com um outro modificador de acesso.

Membros com acessibilidade de pacote só podem ser acessados por outras classes ou interfaces definidas dentro do mesmo pacote.



<padrão/>

- O modificador de acesso padrão, também conhecido como acessibilidade de pacote, é o modificador atribuído aos membros da classe que não foram marcados explicitamente com um outro modificador de acesso.
- Membros com acessibilidade de pacote só podem ser acessados por outras classes ou interfaces definidas dentro do mesmo pacote (somente para as classes que se encontram sob o mesmo pacote da classe que possui o item)
- Em uma herança a restrição de acesso atinge também as classes filhas que não estejam no mesmo pacote, impedindo-as de acessarem o item sob o modificador default.

<padrão/>

- Para esse modificador não há uma palavra chave definida para o uso aqui.
- O modificador aqui é a omissão dos outros modificadores.
- Existe uma palavra-chave ‘default’ na Linguagem Java, mas essa palavra é utilizada em switches e interfaces.
- Esse modificador pode ser aplicado sobre os itens
  - Classes
  - Classes Internas
  - Interfaces Internas
  - Annotations Internas
  - Enums Internos
  - Métodos
  - Atributos

<default/>

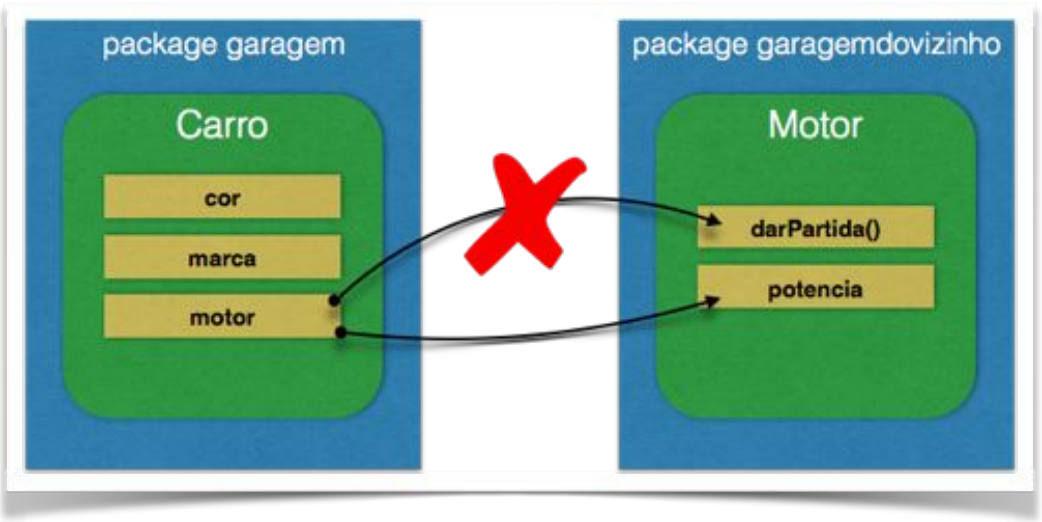
```
package garagem;
public class Motor {
    public int potencia;
    void darPartida(){}
}
```

```
package garagem;
public class Carro {
    String marca;
    String cor;
    public Motor motor;
    void ligar()
    {
       this.motor.darPartida();
    }
    public String toString()
    {
        return marca + " " +
cor + " " + motor.potencia;
    }
}
```

Ao colocar a classe `Motor` dentro do pacote `garagemdovizinho`, a classe `Carro` não consegue mais ter o acesso ao método `darPartida()` da classe `Motor`.

```
package garagemdovizinho;

public class Motor {
    public int potencia;
    void darPartida(){}
}
```

`Perdemos o acesso ao método darPartida() da classe Motor.`




```
package garagem;
import
garagemdovizinho.Motor;
public class Carro {
    String marca;
    String cor;
    public Motor motor;
    void ligar()
    {
this.motor.darPartida(); // O
método darPartida() do tipo
Motor não é visível.
    }

    public String toString()
    {
        return marca + " " +
cor + " " + motor.potencia;
    }
}
```

<default - classes derivadas/>

Com classes derivadas a regra continua a mesma, só as classes derivadas declaradas dentro do mesmo pacote têm acesso aos membros com acessibilidade de pacote da classe mãe.

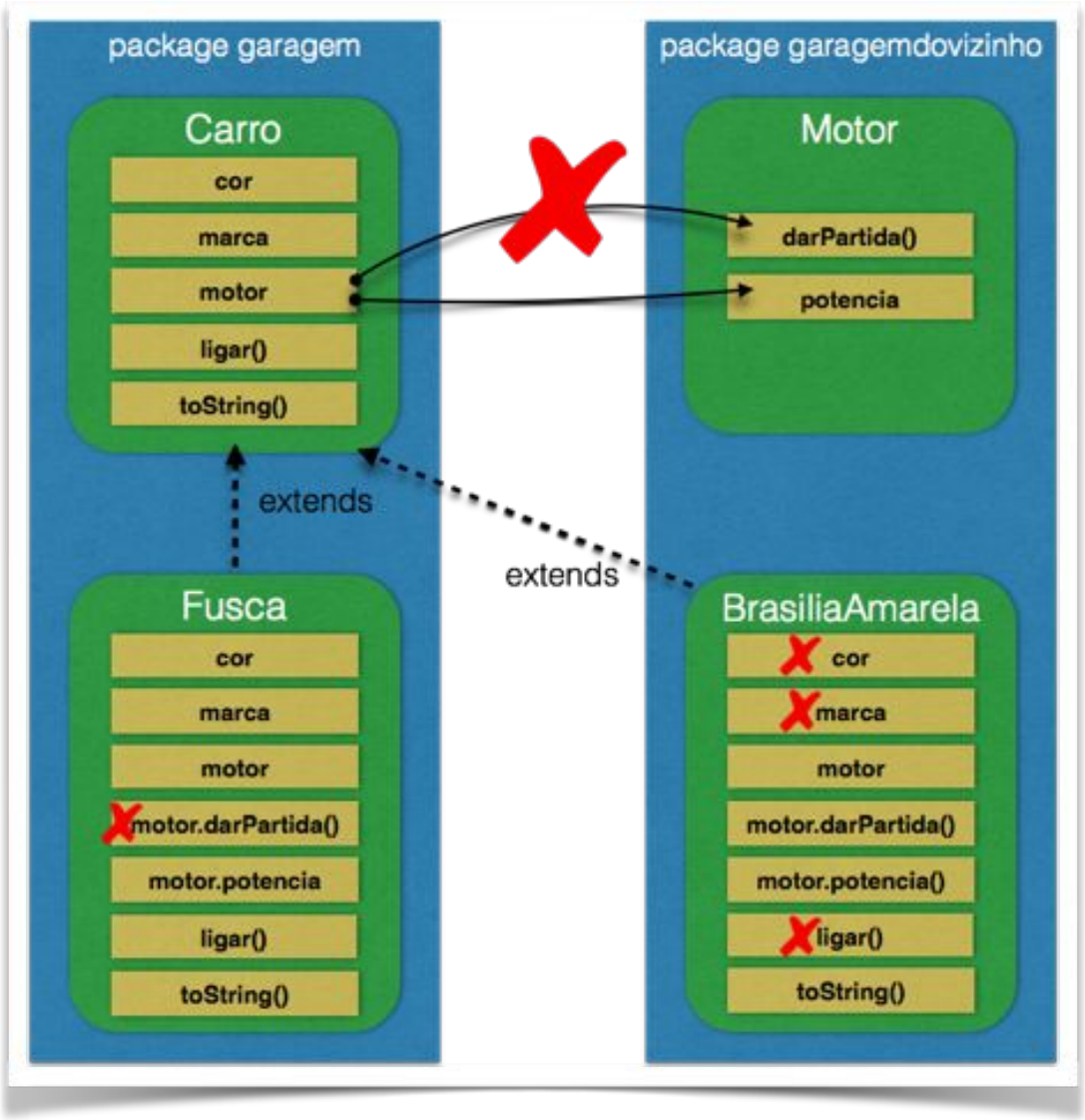
<default - classes derivadas/>

```
package garagem;
public class Fusca extends Carro {

    public Fusca()
    {
        this.cor = "Branco";
        this.marca = "VW";
        this.ligar();
        this.motor.darPartida(); // O
método darPartida() do tipo Motor não
é visível.
        this.motor.potencia = 100;
        this.toString();
    }
}
```

```
package garagemdovizinho;
import garagem.Carro;
public class BrasiliaAmarela
extends Carro {
     public BrasiliaAmarela() {
        this.cor = "Branco"; //
O campo Carro.cor não é visível.
        this.marca = "VW"; // O
campo Carro.marca não é visível.
        this.ligar(); // O
método ligar() do tipo Carro não
é visível.
        this.motor.darPartida();
        this.motor.potencia =
100;
        this.toString();
    }
}
```

Apesar da variável de classe motor do tipo Motor estar visível, o método darPartida() não está.

Na classe Carro, a qual a classe BrasiliaAmarela tem um relacionamento de herança, o método darPartida() da classe Motor não está visível. Mas na classe BrasiliaAmarela, o mesmo método é visível.



<default/>

Observe que apesar das classes Fusca e BrasiliaAmarela terem acesso à classe Motor por herdarem da classe Carro, a acessibilidade aos membros da classe Carro por essas duas classes filhas é diferente.

A classe Fusca só tem acesso à variável de classe potencia, enquanto a classe BrasiliaAmarela tem acesso à variável de classe potencia e ao método darPartida().

O modificador de acesso padrão é mais restritivo do que o modificador public e protected, mas ele é menos restritivo do que o modificador private.

<acessos - default/>

Acessibilidade do modificador **padrão**

<private/>

<private/>

- O modificador de acesso private é o mais restritivo modificador de acesso.
- O membro é visível somente para a classe que o definiu.
- Todo membro de uma classe definido com o modificador private só é acessível para a própria classe.
- Não importa a localização dentro de pacotes ou se a classe foi herdada ou não, um membro private só é acessível dentro da mesma classe em que ele foi declarado.
- Em uma herança a restrição de acesso atinge também as classes filhas, impedindo-as de acessarem o item sob o modificador private.
- Pode ser aplicado sobre os seguintes itens:
  - Classes Internas
  - Interfaces Internas
  - Annotations Internas
  - Enums Internos
  - Métodos
  - Atributos

<private/>

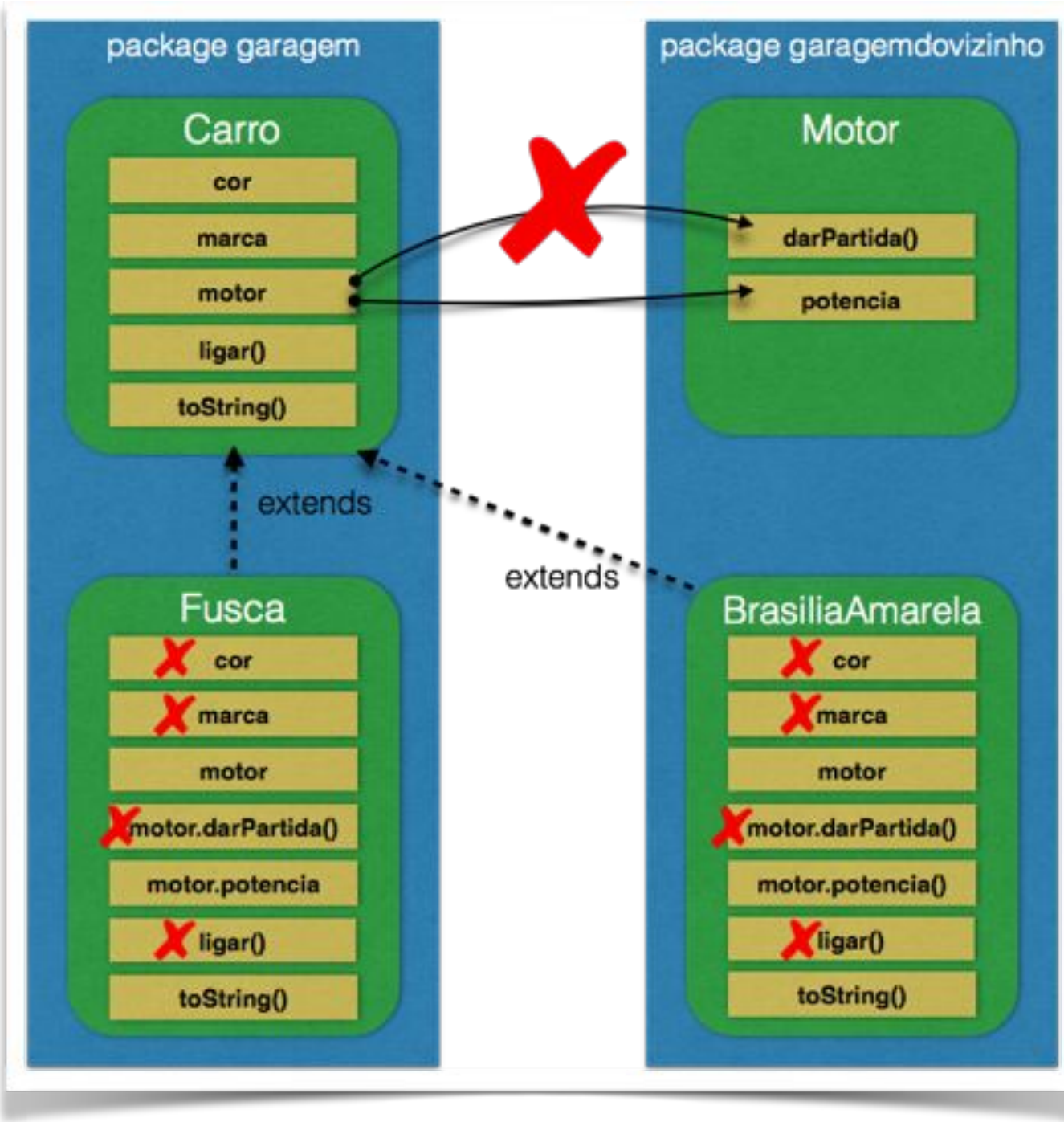
```
package garagemdovizinho;

public class Motor {
    public int potencia;
    private void
darPartida(){}
}
```

```
package garagem;
import garagemdovizinho.Motor;

public class Carro {
    private String marca;
    private String cor;
    public Motor motor;
    private void ligar()
    {
        this.motor.darPartida();
// O método darPartida() do tipo Motor não é
visível.
    }
    public String toString()
    {
        return marca + " " + cor + " " +
motor.potencia;
    }
}
```

<private/>

```
package garagem;

public class Fusca extends Carro {

    public Fusca()
    {
        this.cor = "Branco";
// O campo Carro.cor não é visível.
        this.marca = "VW";
// O campo Carro.marca não é visível.
        this.ligar();
// O método ligar() do tipo Carro não é visível.
        this.motor.darPartida();
// O método darPartida() do tipo Motor não é visível.
        this.motor.potencia = 100;
        this.toString();
    }
}
```

```
package garagemdovizinho;
import garagem.Carro;

public class BrasiliaAmarela
extends Carro {
  public BrasiliaAmarela() {
    this.cor = "Branco";
// O campo Carro.cor não é visível.
    this.marca = "VW";
// O campo Carro.marca não é visível.
    this.ligar();
// O método ligar() do tipo Carro não é visível.
   this.motor.darPartida();
// O método darPartida() do tipo Motor não é visível.
   this.motor.potencia = 100;
   this.toString();
    }
}
```


package garagem
Carro
cor
marca
motor
ligar()
toString()
extends
Fusca
cor
marca
motor
motor.darPartida()
motor.potencia
ligar()
toString()
package garagemdovizinho
Motor
darPartida()
potencia
extends
BrasiliaAmarela
cor
marca
motor
motor.darPartida()
motor.potencia()
ligar()
toString()

<acessos - private/>

Acessibilidade do modificador **private**

<visibilidade de acessos/>

Para testar a visibilidade de um item com o seu modificador de acesso há 5 cenários diferentes:

A partir da mesma classe

- Qualquer classe no mesmo package
- Qualquer classe filha no mesmo package
- Qualquer classe filha em package diferente
- Qualquer classe em package diferente

<visibilidade de acessos/>

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

//Se o código estiver na mesma classe ModificadorPrivate não ocorrerá erro de compilação
	System.out.println(new ModificadorPrivate().nome);

//Se a classe que contém o código estiver no mesmo package da classe ModificadorDefault não ocorrerá erro de compilação
	System.out.println(new ModificadorDefault().nome);

//Se a classe que contém o código estiver no mesmo package ou for filha da classe ModificadorProtected não ocorrerá erro de compilação
	System.out.println(new ModificadorProtected().nome);

//Não ocorrerá erro de compilação
	System.out.println(new ModificadorPublic().nome);
}
```

<acessos/>

As figuras verdes indicam que não houve erro de compilação e as figuras vermelhas indicam que houve erro de compilação.

Visibilidade	public	protected	default	private
A partir da mesma classe	✓	✓	✓	✓
Qualquer classe no mesmo pacote	✓	✓	✓	⊘
Qualquer classe filha no mesmo pacote	✓	✓	✓	⊘
Qualquer classe filha em pacote diferente	✓	✓	⊘	⊘
Qualquer classe em pacote diferente	✓	⊘	⊘	⊘