Herança

P. O. O.

Prof. Grace

O que é Herança?

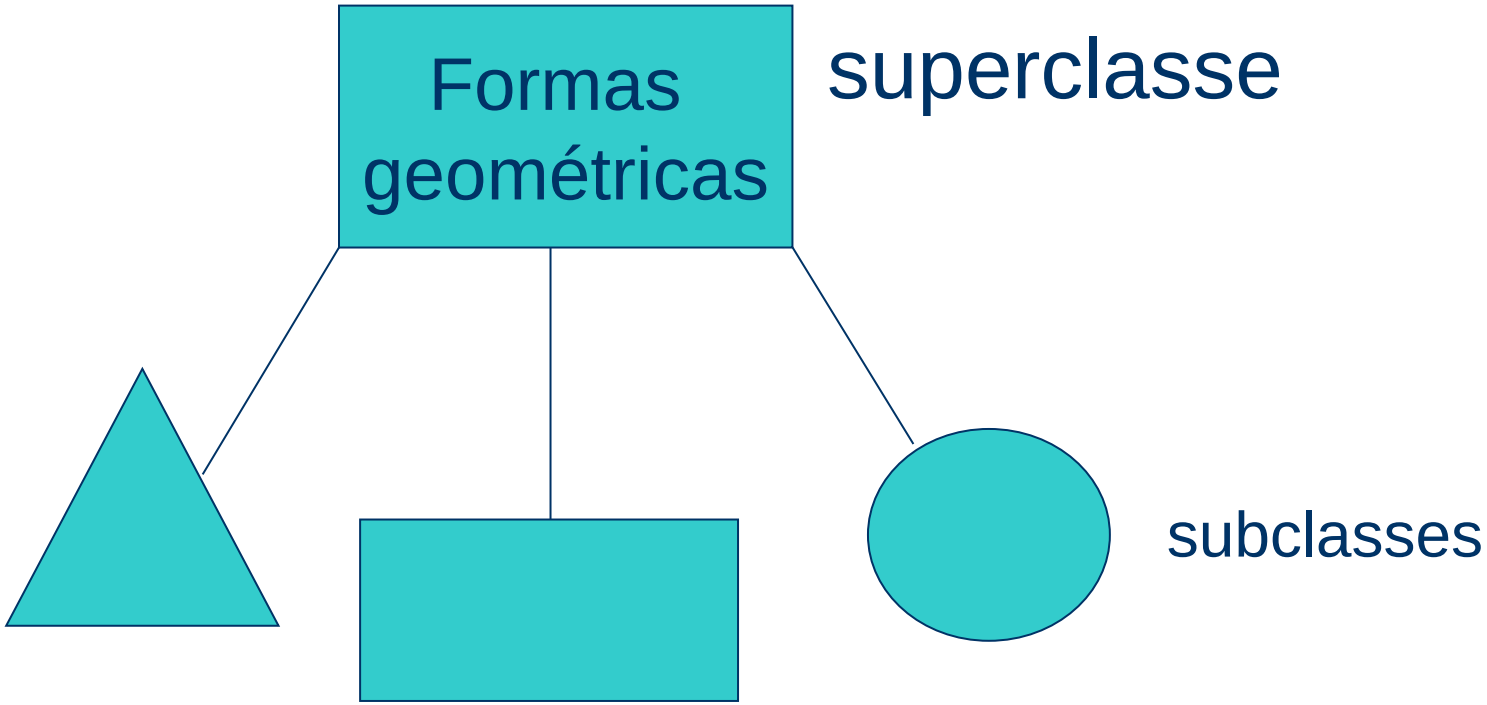
Herança

- Cria uma nova classe a partir de uma classe existente:
  - absorvendo os dados e comportamentos da classe existente; e
  - aprimorando-a com novas capacidades.
- Adota um relacionamento hierárquico entre classes
- Permite melhor organização e reuso de código

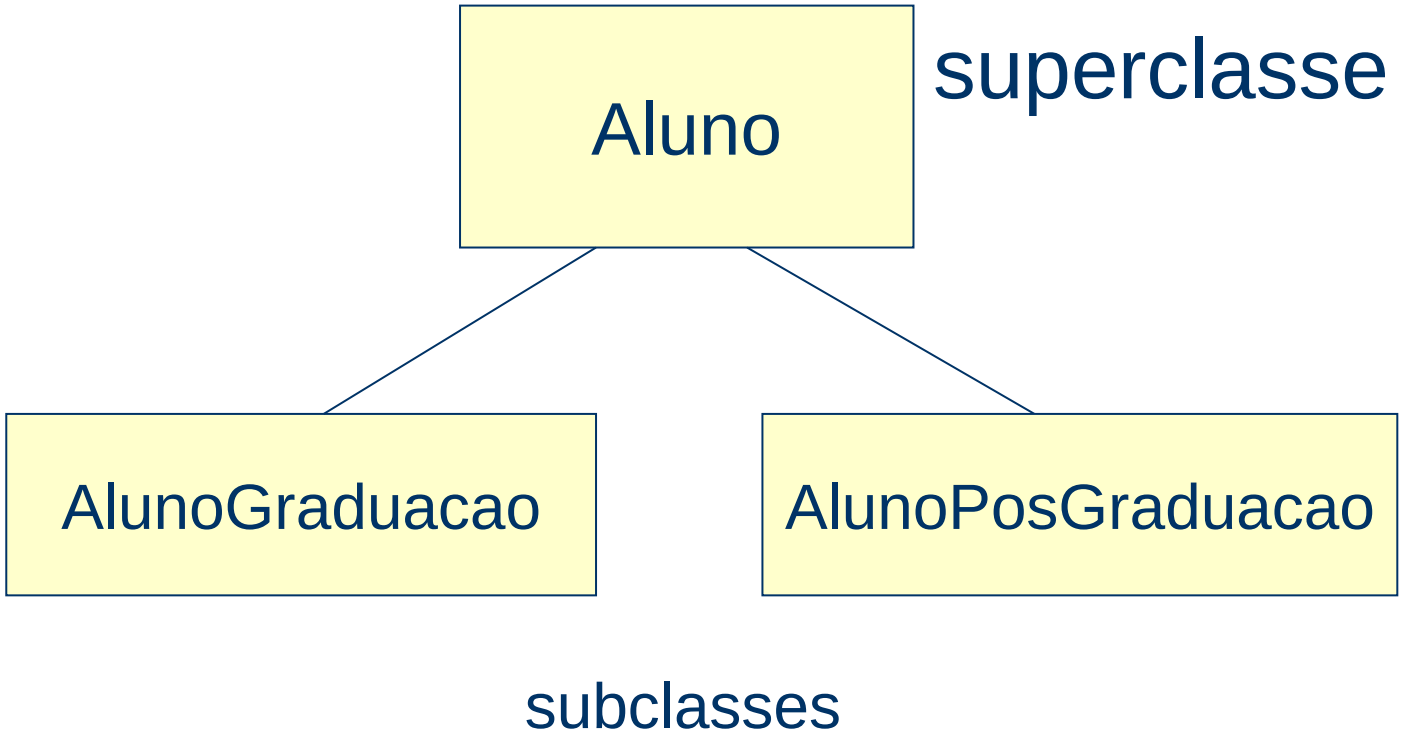
Tipos de classes quanto a Herança

- Subclasse ou classe derivada:
  - criada a partir de outra classe (classe mãe)
  - herda características da classe mãe
  - também possui características próprias
- Superclasse ou classe base:
  - concede características a classe derivada
- Relação: Subclasse estende a superclasse

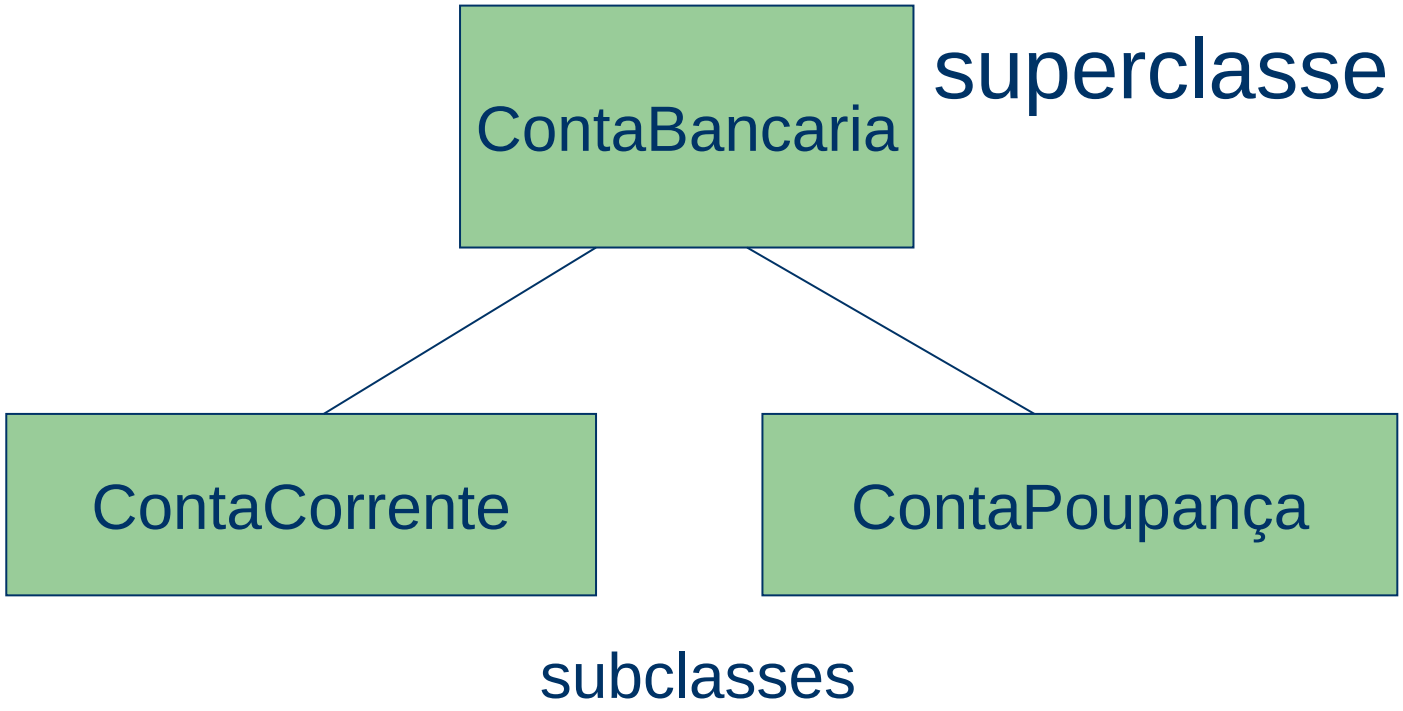
Exemplo: Formas geométricas



Herança - Alunos



Herança – Conta Bancária



Herança – Hierarquia de classes

- A superclasse representa um conjunto maior de objetos do que as subclasses.
  - Superclasse **Veículo:** representa carros, caminhões, barcos, bicicletas…
  - Subclasse **Carro:** representa um subconjunto específico de veículos
- Relação de hierarquia: “é um”
  - Carro “é um” Veículo

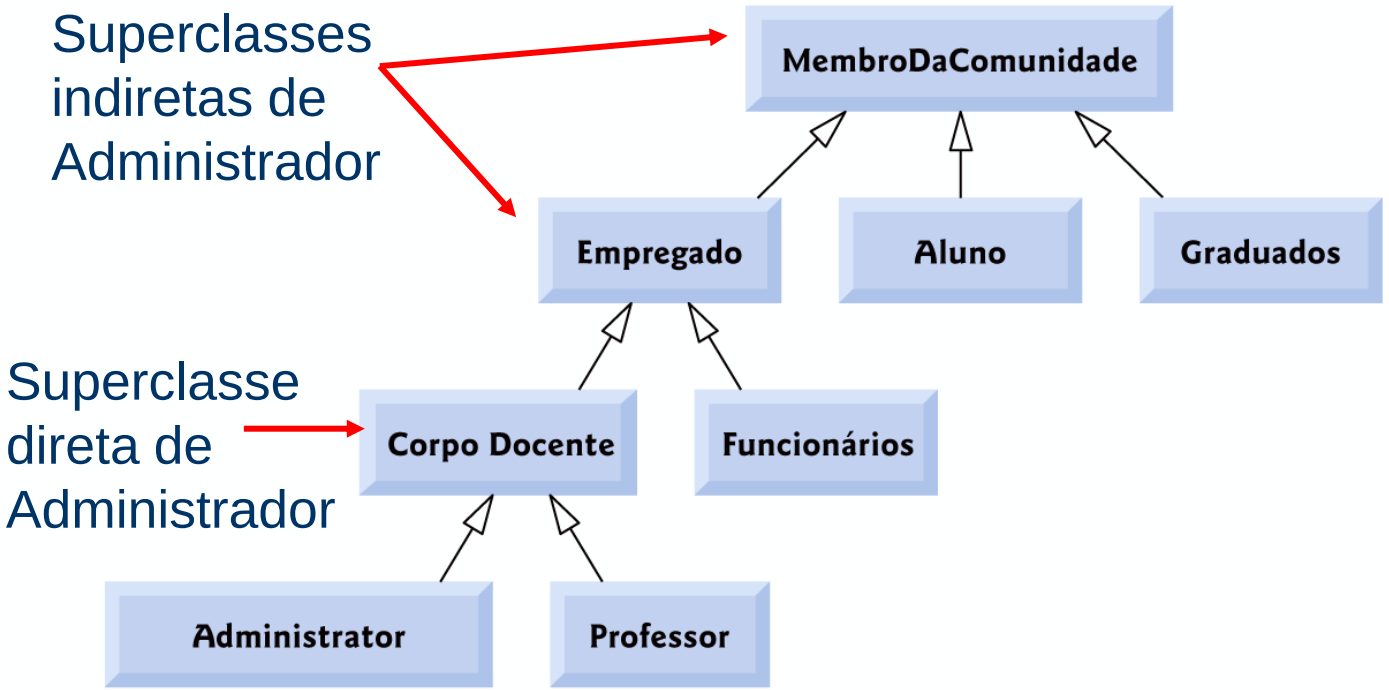
Herança – Hierarquia de classes

- A superclasse é mais geral do que suas subclasses.
- Uma subclasse é uma especialização de uma superclasse;
- A superclasse é uma generalização de subclasses;

Herança – Hierarquia de classes

- Superclasse direta:
  - Herdada explicitamente (um nível acima na hierarquia).
- Superclasse indireta:
  - Herdada de dois ou mais níveis acima na hierarquia.

Hierarquia de classes



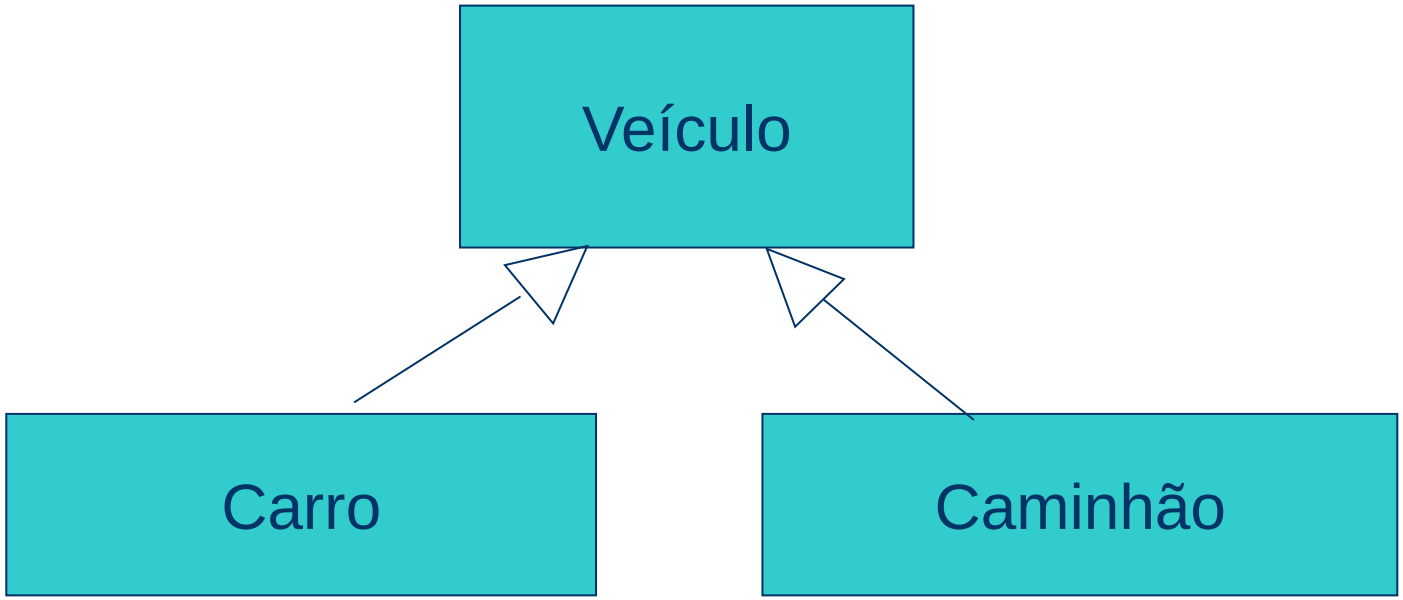
Superclasse Object

- O compilador Java configura a superclasse de uma classe como `Object` quando a declaração de classe não estender uma superclasse explicitamente.

Herança – Hierarquia de classes

- Herança única:
  - Herda de uma superclasse.
- Herança múltipla:
  - Herda de múltiplas superclasses.
  - O Java não suporta herança múltipla.

Exemplo – Veículo



Exemplo – Classe Veículo

- Atributos básicos
  - Modelo
  - Placa
  - Ano Fabricação
  - Valor
- Métodos básicos
  - Sets e gets
  - Depreciar valor do veículo
  - Impressão dos dados

Classe Veículo

Veiculo
- modelo: String - placa: String - ano Fabricação: int - valor: double
+ <<constructor>> Veiculo (mod: String; pl: String; ano: int; vlr: double) + setModelo (mod: String) + setPlaca (pl: String) + setAno (ano: int) + setValor (vlr: double) + getModelo: String + getPlaca: String + getAno: int + getValor: double + deprecia (tx: float) + imprime( )

Exemplo – Classe Veículo

```
public class Veiculo {
    private String modelo, placa;
    private int anoFabr;
    private double valor;
    public Veiculo(String pModelo, String pPlaca, int pAnoFabr, double pValor){
        setModelo(pModelo);
        setPlaca(pPlaca);
        setAnoFabr(pAnoFabr);
        setValor(pValor);
      }

    public void setModelo(String modelo){
        this.modelo = modelo;
    }
```

Exemplo – Classe Veículo (cont.)

```
public String getModelo( )  {
     return this.modelo;                  // uso opcional nesse caso
}

public void setPlaca(String placa) {
          this.placa = placa;
}

public String getPlaca() {
          return placa;
}

public void setAnoFabr(int anoFabr) {
          this.anoFabr = anoFabr;
}

public int getAnoFabr() {
          return anoFabr;
}
```

Exemplo – Classe Veículo (cont.)

```
    public void setValor(double valor) {
            if (valor >= 0)         this.valor = valor;
            else                    this.valor = 0;
    }

    public double getValor() {
            return valor;
    }

    public void deprecia(float taxa) {
            setValor(valor – valor * taxa/100);
    }

    public void imprime(){
            System.out.printf(
                    " \nVeiculo: %s\nPlaca: %7s\nAno: %4d\nValor: R$%.2f\n",
                    modelo, placa, anoFabr, valor);
    }
}
```

Teste da classe - veículo

```
 1 public class TesteVeiculo
 2 {
 3     public static void main(String args[])
 4     {
 5         Veiculo v;
 6         v = new Veiculo("Fiesta", "ABC1678", 2007,34000);
 7         v.imprime();
 8
 9         v.deprecia(10);
10         System.out.println("Veículo depreciado");
11         v.imprime();
12     }
13 }
```

Implemente
Compile e execute

Resultado

```
 ----jGRASP exec: java TesteVeiculo


Veiculo: Fiesta
Placa: ABC1678
Ano: 2007
Valor: R$ 34000,00
Veículo depreciado

Veiculo: Fiesta
Placa: ABC1678
Ano: 2007
Valor: R$ 30600,00

 ----jGRASP: operation complete.
```

Subclasse Carro

- Atributos específicos
  - Número de portas
  - Ano do modelo
- Métodos
  - Construtor
  - Sets e gets específicos
  - Impressão dos dados do carro

Subclasse Carro

Carro
- numPortas: int - anoModelo: int
+ <<constructor>> carro (mod: String; pl: String; nPortas: int; anoFabr: int; anoMod: int; vlr: double) + setPortas (nPortas: int) + setAnoModelo (ano: int) + getPortas: int + getAnoModelo: int

Demais métodos?

Classe Carro – Atributos e construtor

```
public class Carro extends Veiculo {
    private int numPortas;
    private int anoModelo;

    public Carro(String modelo, String placa, int anoFabr, int
                                anoModelo, int numPortas, double valor) {
        setModelo(modelo);
        setPlaca(placa);
        setAnoFabr(anoFabr);
        setValor(valor);
        setPortas(numPortas);
        setAnoModelo(anoModelo);
    }
```

Implemente

Classe Carro – sets e gets

```
    public void setPortas(int numPortas) {
            this.numPortas = 2;
            if (numPortas > 2)          this.numPortas = numPortas;
    }
    public int getPortas() {
            return numPortas;
    }
    public void setAnoModelo(int anoModelo) {
            this.anoModelo = anoModelo;
    }
    public int getAnoModelo() {
            return anoModelo;
    }
}
```

Compilou?

Implemente e compile

Compilação da classe Carro: erro no construtor!!!

- Caso a superclasse possua construtor definido, a subclasse deve utilizá-lo para garantir integridade dos atributos básicos.

```
public Carro(String modelo, String placa, int anoFabr,
                        int anoModelo, int numPortas, double valor) {
    super (modelo, placa, anoFabr, valor);
    setPortas(numPortas);
    setAnoModelo(anoModelo);
}
```

Teste - Classe Carro

```
1 public class TesteCarro
2 {
3    public static void main(String args[])
4    {  // declaração de objeto da subclasse
5       Carro c;
6
7       // uso do construtor da subclasse
8       c = new Carro("Fiesta", "ABC1678", 2006, 2007, 3, 31000);
9
10      //uso de métodos da superclasse
11      c.imprime();
12      c.deprecia(10);
13
14      System.out.println("Carro depreciado");
15      c.imprime();
16   }
17 }
```

```
Veiculo: Fiesta
Placa: ABC1678
Ano: 2006
Valor: R$ 31000,00

Carro depreciado:
Veiculo: Fiesta
Placa: ABC1678
Ano: 2006
Valor: R$ 27900,00
```

Imprimido dados do carro

- O método “imprime( )” foi herdado da classe Veículo
- Não possui todos os dados do carro
- Como codificar método imprime( ) para objetos do tipo Carro?
- No programa teste, qual método será executado: da classe Veículo ou da classe Carro?

Alterando a Classe Carro

```
public void imprime( )
{
    System.out.printf("\nVeiculo: %s\nPlaca: %7s", modelo, placa);
    System.out.printf("\nFabr: %4d\nModelo: %4d", anoFabr,
            anoModelo);
    System.out.printf("\n%02d Portas\nR$ %.2f\n", numPortas, valor);
}
```

O teste do carro funcionou ?

Resultado

```
Carro.java:37: modelo has private access in Veiculo
                                                modelo, placa);
                                                ^
Carro.java:37: placa has private access in Veiculo
                                                modelo, placa);
                                                       ^
Carro.java:39: anoFabr has private access in Veiculo
                                                anoFabr, anoModelo);
                                                ^
Carro.java:41: valor has private access in Veiculo
                                                numPortas, valor);
                                                           ^
4 errors
```

Herança – Acesso aos membros

- Acesso public:
  - Subclasses acessam diretamente membros public de sua superclasse.
- Acesso private:
  - Subclasses não acessam diretamente atributos private da superclasse. Apenas por meio de métodos não-private.

Herança – Acesso aos membros

- Acesso protected:
  - Subclasses acessam diretamente membros protected de sua superclasse.
  - Pode ser usada a palavra-chave “super.”
  - Também são acessíveis a classes de um mesmo pacote (conjunto de classes)

Alterando a classe Veiculo

```
public class Veiculo
{
   protected String modelo, placa;
   protected int anoFabr;
   protected double valor;

...
}
```

O teste do carro funcionou ?

Resultado

```
 ----jGRASP exec: java TesteCar


Veiculo: Fiesta
Placa: ABC1678
Fabr: 2006
Modelo: 2007
03 Portas
R$ 31000,00

Carro depreciado:
Veiculo: Fiesta
Placa: ABC1678
Fabr: 2006
Modelo: 2007
03 Portas
R$ 27900,00

 ----jGRASP: operation complete
```

Atividade – Classe Caminhão

- Subclasse Caminhao estende Veiculo
  - Atributos específicos
    - Capacidade
    - Número de eixos
  - Métodos
    - Construtor
    - Sets e gets
    - Impressão dos dados do caminhão

Teste – Classe Caminhao

```
1 public class TesteCaminhao
2 {
3    public static void main(String args[])
4    {  // declaração de objeto da subclasse
5       Caminhao c;
6
7       // uso do construtor da subclasse
8       c = new Caminhao("Mercedes", "XYZ3456", 2000, 10, 3, 50000);
9
10      //uso de métodos da superclasse
11      c.imprime();
12      c.deprecia(10);
13
14      System.out.println("Caminhao depreciado");
15      c.imprime();
16   }
17 }
```

Atividades

- Classes:
  - Veiculo
  - Carro + Programa de testeCarro
  - Caminhão + Programa de testeCaminhao

- E-mail ***poo.profgrace@yahoo.com.br***
- Identifique quais atividades estão sendo enviadas no ***subject/ assunto*** da mensagem.
  - Ex.: Assunto: Entrega de Atividades – Herança