




Java WebDeveloper - Aula 02 - Noite Seg, Qua, Sex - Noite



Tema da aula:

Orientação a objetos, Calculo, Ternário, Validação de Dados, Regra Pattern, Mostra mensagens, Flag, Diferença entre & e | (ou), Herança, Construtor super, Lista, Vetor, Stream, Reduce

—  Prof Edson Belém - profedsonbelem@gmail.com (mailto:profedsonbelem@gmail.com) 🕒 Quarta, Mar 11, 2020

Herança

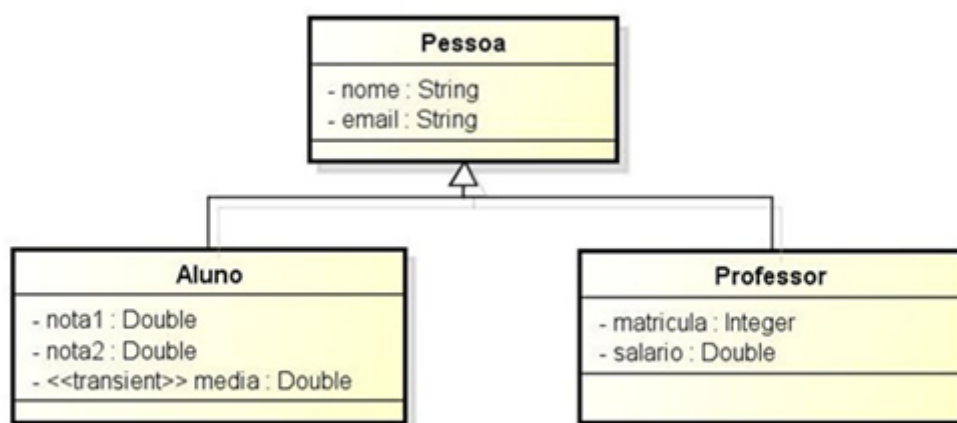


A herança é um relacionamento pelo qual uma classe, chamada de sub-classe, herda todos comportamentos e estados possíveis de outra classe, chamada de super-classe ou classe base. É permitido que a sub-classe estenda os comportamentos e estados possíveis da super-classe (por isso este relacionamento também é chamado de extensão). Essa extensão ocorre adicionando novos membros a sub-classe, como novos métodos e atributos.

É também possível que a sub-classe altere os comportamentos e estados possíveis da super-classe. Neste caso, a sub-classe sobrescreve membros da super-classe, tipicamente métodos.

Quando uma classe herda de mais de uma super-classe, ocorre uma herança múltipla. Esta técnica é possível em C++ e em Python, mas não é possível em Java e C#, no entanto estas linguagens permitem múltipla tipagem através do uso de interfaces.

O objetivo da herança no Java é agrupar atributos comuns de varias classes em uma única classe denominada Super-Classe, e as características específicas ficam na Sub-Classe. No Java o comando que representa a herança é chamado extends. E uma Sub-Classe não pode ter mais de uma Super-Classe.



super() x super

O comando Java `super()`, "super com parêntese", no construtor da sub-Classe tem a finalidade de chamar os atributos da Super-Classe. O comando obrigatoriamente fica localizado na primeira linha do construtor cheio.

```
1 public Aluno(String nome, String email, Double nota1, Double nota2) {
2     super(nome, email);
3     this.nota1 = nota1;
4     this.nota2 = nota2;
5 }
```

O comando Java super, "super sem parêntese" , no método da sub-classe tem a finalidade de chamar o método com o mesmo nome da Super-Classe.

```
1  @Override
2  public String toString() {
3      return super.toString() + "Aluno [nota1=" + nota1 + ", nota2="
4      + nota2 + "];"
5  }
```

Apresentação das Classes no Java.

Super-Classe Pessoa possui os atributos comuns as Sub-Classes Aluno e Professor.

```
1  package entity;
2
3  public class Pessoa {
4
5      private String nome;
6      private String email;
7
8      public Pessoa() {
9      }
10
11     public Pessoa(String nome, String email) {
12         super();
13         this.nome = nome;
14         this.email = email;
15     }
16
17     @Override
18     public String toString() {
19         return "Pessoa [nome=" + nome + ", email=" + email + "];"
20     }
21
22     public String getNome() {
23         return nome;
24     }
25     public void setNome(String nome) {
26         this.nome = nome;
27     }
28     public String getEmail() {
29         return email;
30     }
31     public void setEmail(String email) {
32         this.email = email;
33     }
34 }
```

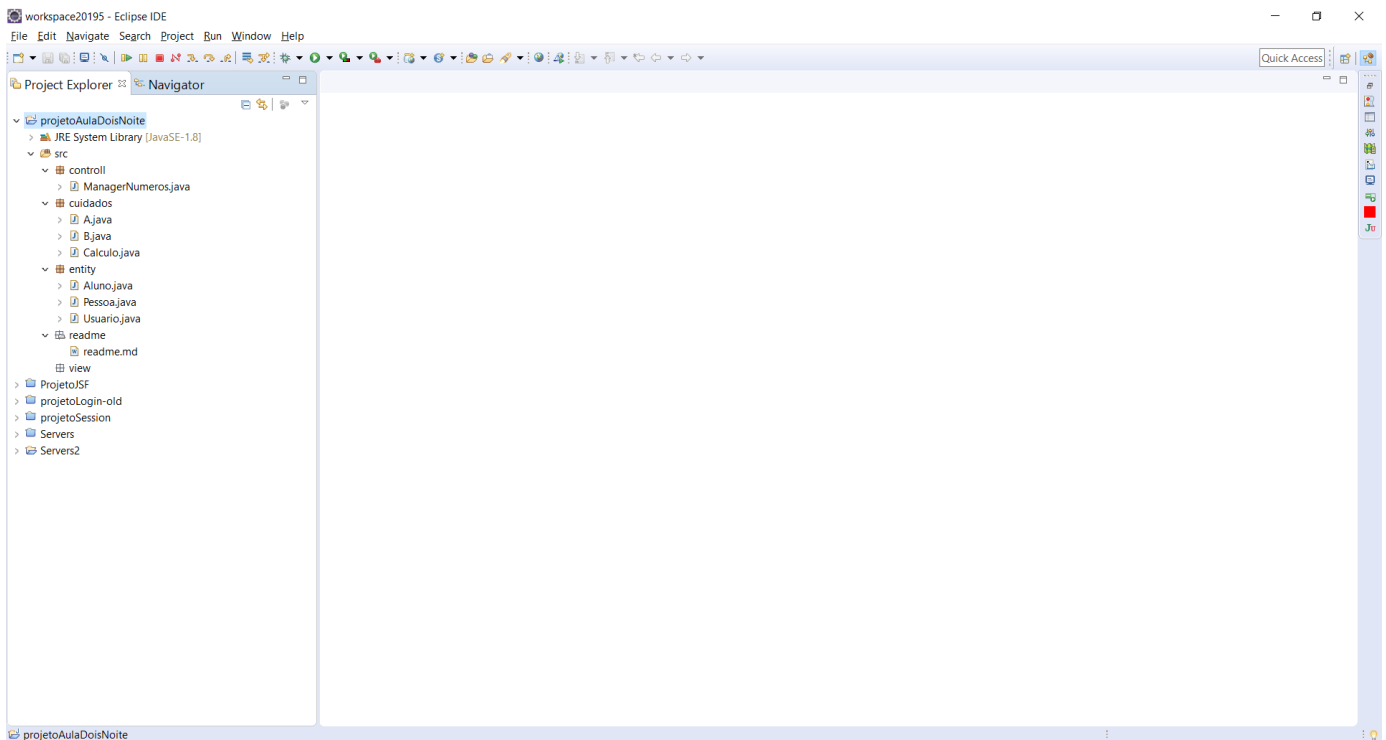
Sub-Classes Aluno e Professor com esses atributos específicos.

```
1  package entity;
2
3  public class Aluno extends Pessoa {
4
5      public Double nota1;
6      public Double nota2;
7      public transient Double media;
8
9      public Aluno() {
10     }
11
12     public Aluno(String nome, String email, Double nota1, Double nota2) {
13         super(nome, email);
14         this.nota1 = nota1;
15         this.nota2 = nota2;
16     }
17
18     @Override
19     public String toString() {
20         return super.toString() + "Aluno [nota1=" + nota1 +
21         ", nota2=" + nota2 + "]";
22     }
23
24     public Double getNota1() {
25         return nota1;
26     }
27     public void setNota1(Double nota1) {
28         this.nota1 = nota1;
29     }
30     public Double getNota2() {
31         return nota2;
32     }
33     public void setNota2(Double nota2) {
34         this.nota2 = nota2;
35     }
36     public Double getMedia() {
37         return media;
38     }
39     public void setMedia(Double media) {
40         this.media = media;
41     }
42 }
```

```
1  package entity;
2
3  public class Professor extends Pessoa {
4
5      private Integer matricula;
6      private Double salario;
7
8      public Professor() {
9      }
10
11     public Professor(String nome, String email, Integer matricula,
12         Double salario) {
13         super(nome, email);
14         this.matricula = matricula;
15         this.salario = salario;
16     }
17
18     @Override
19     public String toString() {
20         return super.toString() + "Professor [matricula=" +
21             matricula + ", salario=" + salario + "];"
22     }
23
24     public Integer getMatricula() {
25         return matricula;
26     }
27     public void setMatricula(Integer matricula) {
28         this.matricula = matricula;
29     }
30     public Double getSalario() {
31         return salario;
32     }
33     public void setSalario(Double salario) {
34         this.salario = salario;
35     }
36 }
```

```
1 package main;
2
3 import entity.Aluno;
4 import entity.Professor;
5
6 public class Main {
7
8     public static void main(String[] args) {
9
10        // Herança
11        // Instanciando os objetos pelas Sub-Classes
12        Aluno a = new Aluno("Jose", "jose@gmail.com", 10., 8.);
13        System.out.println(a);
14        //-----
15        Professor p = new Professor("Luiz", "luiz@gmail.com", 123, 3000.);
16        System.out.println(p);
17    }
18 }
```

👁 Estrutura do projeto depois de finalizado:



</> **Readme.md** (<http://Readme.md>)

```
1  ## Classe Readme ...
2  package entity;
3
4  public class Pessoa {
5
6      //set (Alimentar)
7      //get
8      private String nome;
9      private String sexo;
10     private Integer idade; //jamais armazene em BD
11     //alt + s gerar get e set
12     //alt + s gerar Construtor (vazio e cheio)
13
14     //ctrl + espaço (Sobrecarga de Construtor) _ tecnicamente
15     public Pessoa() {
16     } //escolha (Sobrecarga)
17
18     //vou utilizar o set ....
19
20     //ele irá elmiar os futuros set ....
21     public Pessoa(String nome, String sexo, Integer idade) {
22         this.nome = nome;
23         this.sexo = sexo;
24         this.idade = idade;
25     }
26
27     //toString() retorna tudo
28     //Construtor cheio ..
29     ///Retornar um valor (toString()) Retorna todos os Valores ...
30     @Override
31     public String toString() {
32         return "Pessoa [nome=" + nome + ", sexo=" + sexo +
33         ", idade=" + idade + "]";
34     }
35
36     public String getNome() {
37         return nome;
38     }
39     public void setNome(String nome) {
40         this.nome = nome;
41     }
42     public String getSexo() {
43         return sexo;
44     }
45     public void setSexo(String sexo) {
46         this.sexo = sexo;
47     }
48     public Integer getIdade() {
49         return idade;
50     }
51     public void setIdade(Integer idade) {
52         this.idade = idade;
53     }
54 }
```

```
55  ##
56
57
58  package controll;
59
60  import java.util.ArrayList;
61  import java.util.Arrays;
62
63  public class ManagerNumeros {
64      //stream
65      //static (aloca)
66      static ArrayList<Integer> numeros = new ArrayList<Integer>();
67
68      //anonimo static ( 1 primeiro método a ser executado
69      static {
70          //adiciona um a um
71          numeros.add(1);numeros.add(10);numeros.add(4); numeros.add(5);
72          //adiciona um grupo ...
73          numeros.addAll(Arrays.asList(new Integer[] {6,7,8}));
74          //addAll (adicionar grupo)
75          // Arrays.asList ((converte vetor em lista)
76          //new Integer[]{valores} ... tamanho indeterminado
77      }
78
79      public static void main(String[] args) {
80          //O que é objeto ? espaço de execucao de mem
81          //static está alocado !!! (nao precisa de objeto)
82          System.out.println(ManagerNumeros.numeros);
83      }
84
85  }
```

</> Aluno.java


```
1 package entity;
2
3 public class Aluno {
4
5     // 4 Entradas (id, nome, disciplina, nota1, nota2)
6     private Integer id;
7     private String nome;
8     private String disciplina;
9     private Double nota1;
10    private Double nota2;
11
12    private Double media; // saída
13    private String situacao; // saída
14    // No construtor nao tenho saída só tenho Entrada ...
15
16    public Aluno(Integer id, String nome, String disciplina,
17        Double nota1, Double nota2) {
18        super();
19        this.id = id;
20        this.nome = nome;
21        this.disciplina = disciplina;
22        this.nota1 = nota1;
23        this.nota2 = nota2;
24    }
25
26    public Aluno() {
27    }
28
29    // e retorno a Classe (Insiro this) ...
30    // a uma substituição para de void para Classe
31    // public void gerarMedia() {
32    // this.media =(this.nota1 + this.nota2)/2;
33    // }
34    // metodotodo imã do bem ...
35
36    public Aluno gerarMedia() {
37        this.media = (this.nota1 + this.nota2) / 2;
38        return this;
39    }
40
41    @Override
42    public String toString() {
43        return "Aluno [id=" + id + ", nome=" + nome + ",
44            disciplina=" + disciplina + ", nota1=" + nota1 + ", nota2="
45            + nota2 + ", media=" + media + ", situacao=" + situacao + "];"
46    }
47
48    public Aluno gerarSituacao() {
49        //Fazemos um ternario aqui
50        this.situacao = (this.media >= 7) ? "aprovado" :
51            (this.media >= 5) ? "recuperacao" : "reprovado";
52        return this;
53    }
54
```

```
55     public static void main(String[] args) {
56         Aluno a = new Aluno(100, "belem", "java", 6., 4.).
57         gerarMedia().gerarSituacao();
58         System.out.println(a);
59     }
60
61     public Integer getId() {
62         return id;
63     }
64
65     public void setId(Integer id) {
66         this.id = id;
67     }
68
69     public String getNome() {
70         return nome;
71     }
72
73     public void setNome(String nome) {
74         this.nome = nome;
75     }
76
77     public String getDisciplina() {
78         return disciplina;
79     }
80
81     public void setDisciplina(String disciplina) {
82         this.disciplina = disciplina;
83     }
84
85     public Double getNota1() {
86         return nota1;
87     }
88
89     public void setNota1(Double nota1) {
90         this.nota1 = nota1;
91     }
92
93     public Double getNota2() {
94         return nota2;
95     }
96
97     public void setNota2(Double nota2) {
98         this.nota2 = nota2;
99     }
100
101     public Double getMedia() {
102         return media;
103     }
104
105     public void setMedia(Double media) {
106         this.media = media;
107     }
108
109     public String getSituacao() {
```

```
110         return situacao;
111     }
112
113     public void setSituacao(String situacao) {
114         this.situacao = situacao;
115     }
116 }
```

</> Pessoa.java

```
1 package entity;
2
3 import java.util.regex.Matcher;
4 import java.util.regex.Pattern;
5
6 public class Pessoa {
7
8     private String nome;
9     private String sexo;
10    private Integer idade;
11
12    public Pessoa() {
13    }
14
15    public Pessoa(String nome, String sexo, Integer idade) {
16        this.nome = nome;
17        this.sexo = sexo;
18        this.idade = idade;
19    }
20
21    @Override
22    public String toString() {
23        return "Pessoa [nome=" + nome + ", sexo=" + sexo +
24        ", idade=" + idade + "]\n";
25    }
26
27    public String getNome() {
28        return nome;
29    }
30
31    public void setNome(String nome) {
32        this.nome = nome;
33    }
34
35    public String getSexo() {
36        return sexo;
37    }
38
39    public void setSexo(String sexo) {
40        this.sexo = sexo;
41    }
42
43    public Integer getIdade() {
44        return idade;
45    }
46
47    public void setIdade(Integer idade) {
48        this.idade = idade;
49    }
50
51    public Boolean isNome() {
52        // regra
53        Pattern p = Pattern.compile("[a-z A-Z]{2,50}");
54        Matcher m = p.matcher(getNome());
```

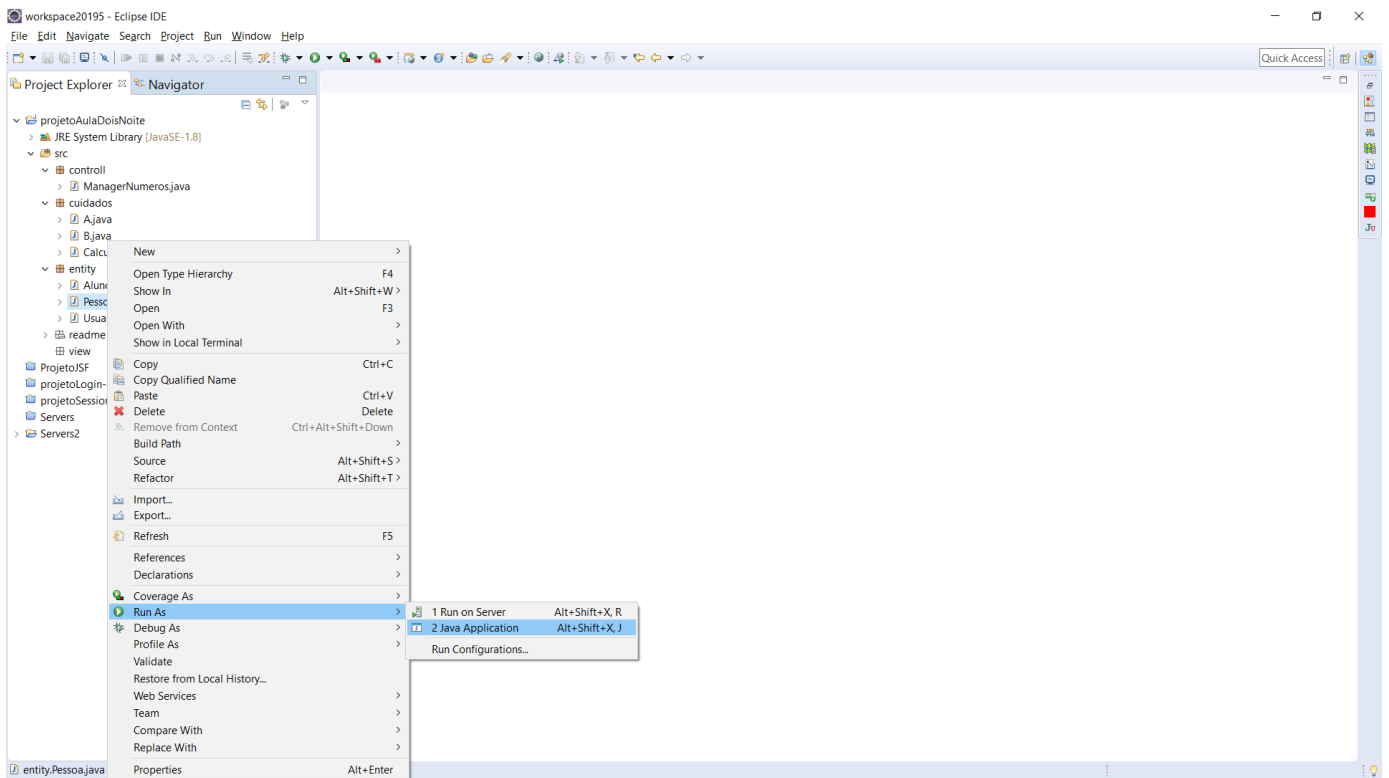
```
55         return m.matches(); // true ou false
56     }
57
58     // isNome()
59
60     public Boolean isIdade() {
61         if (idade < 0 || idade >= 130) {
62             return false;
63         }
64         return true;
65     }
66
67     public Boolean isSexo() {
68         if (this.getSexo().equalsIgnoreCase("M") |
69         this.getSexo().equalsIgnoreCase("F")) {
70             return true;
71         }
72         return false;
73     }
74
75     // main + ctrl + espaco e enter
76     // main + ctrl + espaco de enter
77
78     public static void main(String[] args) {
79         // Construtor cheio
80         Pessoa p = new Pessoa("jose", "m", 30);
81         // alimentando os atributos
82         // xoxean
83         System.out.println(p);
84
85         Pessoa p2 = new Pessoa(); // implicito...
86         p2.setNome("lu");
87         p2.setIdade(22);
88         p2.setSexo("f");
89         System.out.println(p2);
90
91         // -----
92         Pessoa x = new Pessoa("belem1", "", 140);
93         String msg = "";
94         // (Pattern Nao tem else if)
95         int flag = 0;
96         if (!x.isNome()) {
97             msg += "Nome invalido \n";
98             flag = 1;
99         }
100         if (!x.isIdade()) {
101             msg += "idade invalido \n";
102             flag = 1;
103         }
104         if (!x.isSexo()) {
105             msg += "sexo invalido \n";
106             flag = 1;
107         }
108
109         if (flag == 0) {
```

```
110         System.out.println("Dados Ok :" + x);
111     } else {
112         System.out.println("Dados invalido(s):" + x);
113         System.out.println(msg);
114     }
115
116     // glossario
117     // variavel += "adicionar" .... String
118     // numero += acumular
119
120     // Pattern (regra)
121     // Matcher A aplicacao da regra
122     // matches booleano (true ou false)
123
124     // Integer
125     int a = 10; // ==
126     int b = 20;
127     // == primitivo
128     // cacadores da verdade (quando encontra a verdade)
129     // ele para nao continua e entra na condicao
130     // | (ou) ( | curioso)
131
132     // || (caça a verdade e para)
133     // ( | imprensa ... curioso) ...
134     if (a++ == 10 | b++ == 30) {
135         System.out.println("Ok:" + a + "," + b);
136     }
137     // ||
138     // ok:11, 20
139     // |
140     // ok:11, 21
141
142     //=====
143
144     int c = 10; // ==
145     int d = 20;
146     // cacador da mentira(expulsa) // Auditor
147     // && so se for tudo verdadeiro
148     if (c++ == 11 && d++ == 30) {
149         System.out.println("OK :" + c + "," + d);
150     }
151
152     System.out.println("expulso :" + c + "," + d);
153
154     // expulso : 11, 20
155 }
156 }
```

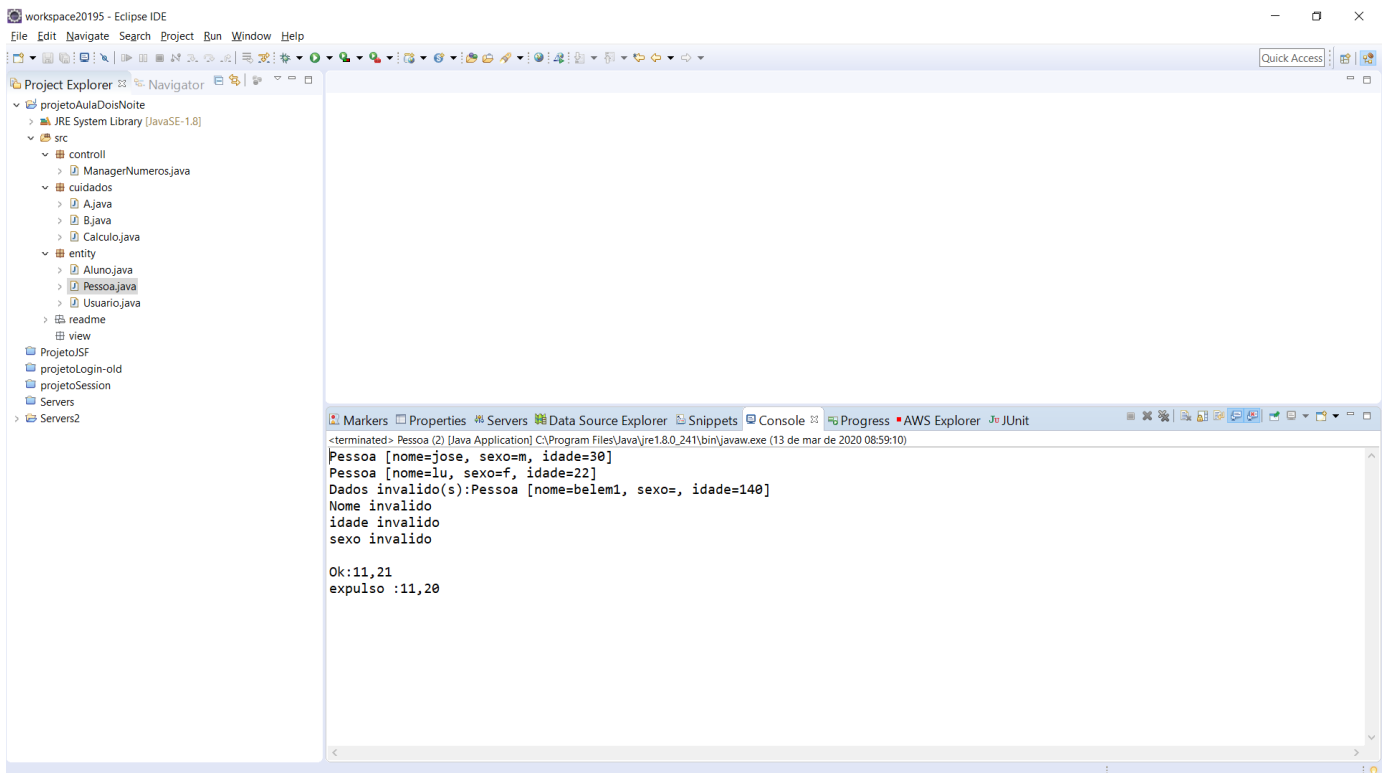


Para rodar a classe:

Clique na classe com o botão direito ➡ run as ➡ java application



👁 Resultado no console:



</> Usuario.java

```
1 package entity;
2
3 //Usuario é uma pessoa
4 //Associacao por generalizacao
5
6 public class Usuario extends Pessoa {
7
8     // Usuario é uma pessoa
9     // Herda ...
10
11     private Integer id;
12     private String login;
13     private String senha;
14
15     // quando usuario herda (busca os dados de Pessoa)
16     // Cosntrutor vazio e construtor cheio
17
18     public Usuario() {
19
20     }
21
22     // construtor dados dele e da superClasse
23
24     public Usuario(Integer id, String nome, String sexo, Integer idade,
25 String login, String senha) {
26         super(nome, sexo, idade);
27         this.id = id;
28         this.login = login;
29         this.senha = senha;
30     }
31     // toString dados dele e de cima
32
33     @Override
34     public String toString() {
35         return "Usuario [id=" + id + ", login=" + login +
36         ", senha=" + senha + ", getNome()" + getNome() + ", getSexo()" +
37         + getSexo() + ", getIdade()" + getIdade() + "]";
38     }
39
40     public Integer getId() {
41         return id;
42     }
43
44     public void setId(Integer id) {
45         this.id = id;
46     }
47
48     public String getLogin() {
49         return login;
50     }
51
52     public void setLogin(String login) {
53         this.login = login;
54     }
```



```
55  
56     public String getSenha() {  
57         return senha;  
58     }  
59  
60     public void setSenha(String senha) {  
61         this.senha = senha;  
62     }  
63 }
```

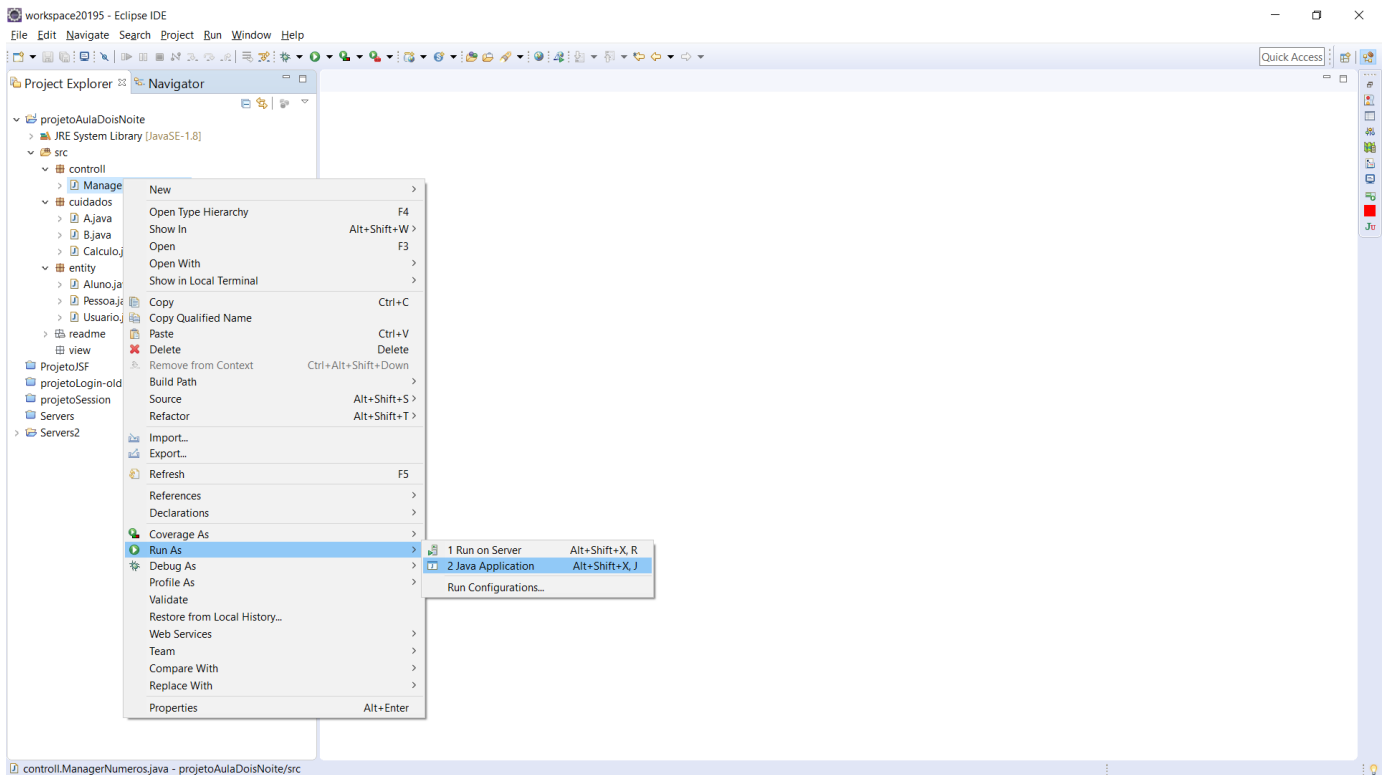
</> **ManagerNumeros.java**


```
1 package controll;
2
3 import java.util.ArrayList;
4 import java.util.Arrays;
5
6 public class ManagerNumeros {
7
8     static ArrayList<Integer> numeros = new ArrayList<Integer>();
9
10    static {
11        numeros.add(1);
12        numeros.add(10);
13        numeros.add(4);
14        numeros.add(5);
15        numeros.addAll(Arrays.asList(new Integer[] { 6, 7, 8 }));
16    }
17
18    public Integer menorNumero() {
19        // a.compareTo(b) --> ordenando ...
20        Integer menor = numeros.stream().min((a, b) ->
21        a.compareTo(b)).get();
22        return menor;
23    }
24
25    public Integer maiorNumero() {
26        Integer maior = numeros.stream().max((a, b) ->
27        a.compareTo(b)).get();
28        return maior;
29    }
30
31    public Long quantidade() {
32        Long conta = numeros.stream().count();
33        return conta;
34    }
35
36    // reduce (achar totais) => 0 valor inicial (Integer::sum)
37    // somatorio
38    public Integer soma() {
39        Integer soma = numeros.stream().reduce(0, Integer::sum);
40        return soma;
41    }
42
43    // BigData (map, reduce)
44
45    public static void main(String[] args) {
46        ManagerNumeros mn = new ManagerNumeros();
47        numeros.stream().forEach(x -> System.out.print(x + " - "));
48
49        System.out.println("");
50        System.out.println("maior :" + mn.maiorNumero());
51        System.out.println("menor :" + mn.menorNumero());
52        System.out.println("Soma :" + mn.soma());
53        System.out.println("Quantidade :" + mn.quantidade());
54    }
55 }
```

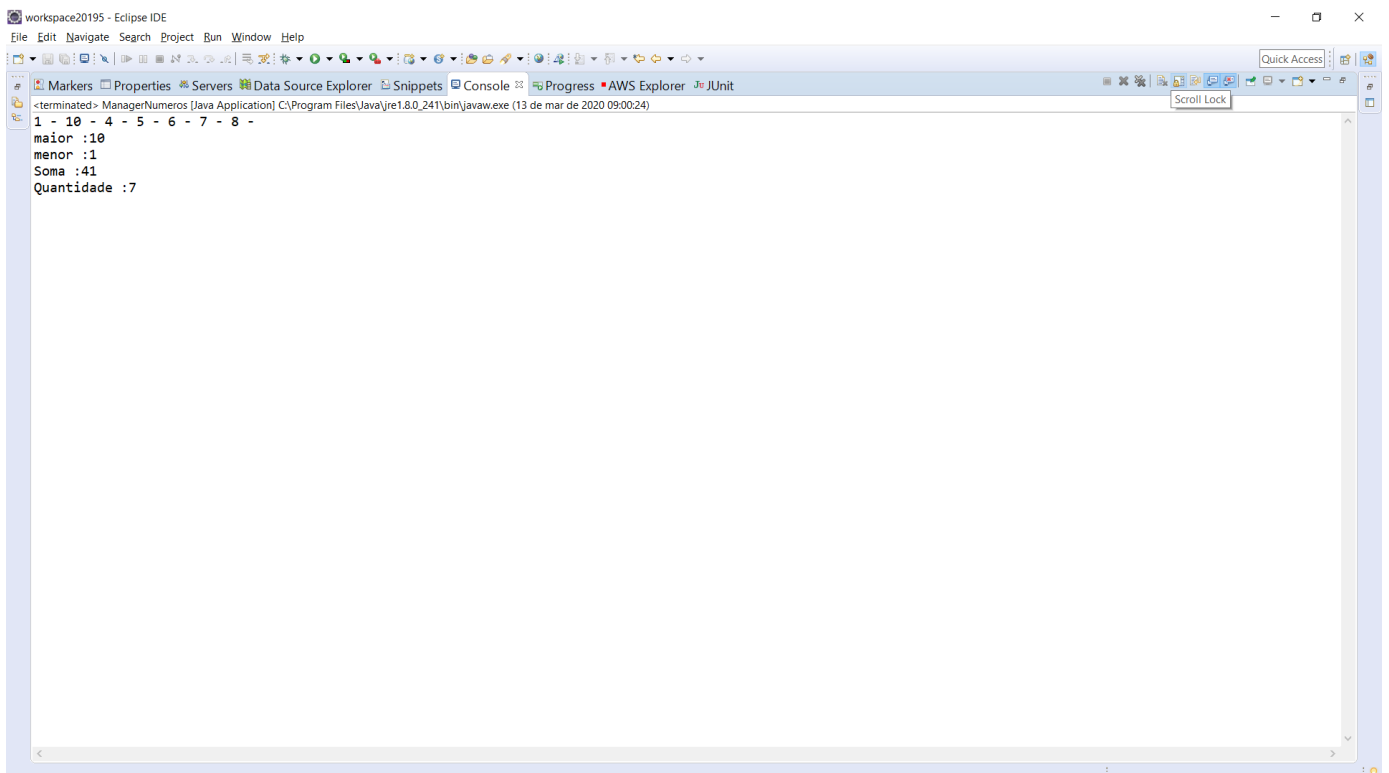


Para rodar a classe:

Clique na classe com o botão direito → run as → java application



👁 Resultado no console:



</> A.java


```
1 package cuidados;
2
3 public class A {
4     // conceito (gerar o cheio e o vazio)
5     // sobrecarga ELIMINA o IMÃ€ ...
6
7     private String nome;
8
9     public A(String nome) {
10         this.nome = nome;
11     }
12
13     public String getNome() {
14         return nome;
15     }
16
17     public void setNome(String nome) {
18         this.nome = nome;
19     }
20 }
21
```

</> B.java

```
1  package cuidados;
2
3  public class B extends A {
4
5      // Classe que herda tem um imã (construtor)
6      // imã
7      String disciplina;
8      Double nota1;
9      Double nota2;
10
11     public B(String nome, String disciplina, Double nota1, Double nota2) {
12         // subir
13         // super caça o cosntrutor da CLASSE DE CIMA
14         // dica do construtor ...
15         super(nome);
16         this.disciplina = disciplina;
17         this.nota1 = nota1;
18         this.nota2 = nota2;
19     }
20
21     public String getDisciplina() {
22         return disciplina;
23     }
24
25     public void setDisciplina(String disciplina) {
26         this.disciplina = disciplina;
27     }
28
29     public Double getNota1() {
30         return nota1;
31     }
32
33     public void setNota1(Double nota1) {
34         this.nota1 = nota1;
35     }
36
37     public Double getNota2() {
38         return nota2;
39     }
40
41     public void setNota2(Double nota2) {
42         this.nota2 = nota2;
43     }
44 }
```

</> Calculo.java

```
1 package cuidados;
2
3 public class Calculo {
4
5     // raio parta
6     // Perseguição implacavel ..
7     // Só roda dando resultado no java dos (java project)
8
9     // POG da Forte ..
10    // Nao Imprima sempre return
11    public void soma1(Double num1, Double num2) {
12        System.out.println(num1 + num2);
13    }
14
15    public Double soma(Double num1, Double num2) {
16        return num1 + num2;
17    }
18 }
```

—  Coti Informática <https://www.cotiinformatica.com.br> (<https://www.cotiinformatica.com.br>)