Considerações Sobre Classes e Objetos

Prof. Ricardo P. Mesquita

Lançando Exceção

- Vamos codificar um programa simples para ilustrar o efeito do lançamento de uma exceção
- Faremos o método set lançar uma exceção por meio de uma cláusula **throw**
- Faremos um bloco **try ... catch** para capturar esta exceção e informar para o usuário
- Vamos visualizar os campos do objeto antes e depois da ocorrência da exceção
- Vamos programar...

Classe Hora_1

```
public class Hora_1 {

    private int hora;
    private int minuto;
    private int segundo;

    public void setHora( int h, int m, int s ){
        if (((( h < 0 || h >= 24 ) || m < 0 ) || m >= 60 ) ||
                s < 0 ) || s >= 60)
        {
            throw new IllegalArgumentException("horas, minutos ou segundos fora da faixa aceitável!!");
        }
        this.hora = h;
        this.minuto = m;
        this.segundo = s;
    }

    public String paraFormato24Horas()
    {
      return String.format( "%02d:%02d:%02d", hora, minuto, segundo );
    }

   public String paraFormato12Horas()
   {
      return String.format( "%d:%02d:%02d %s",
         ( ( hora == 0 || hora == 12 ) ? 12 : hora % 12 ),
         minuto, segundo, ( hora < 12 ? "AM" : "PM" ) );
   }
}
```

Lança exceção

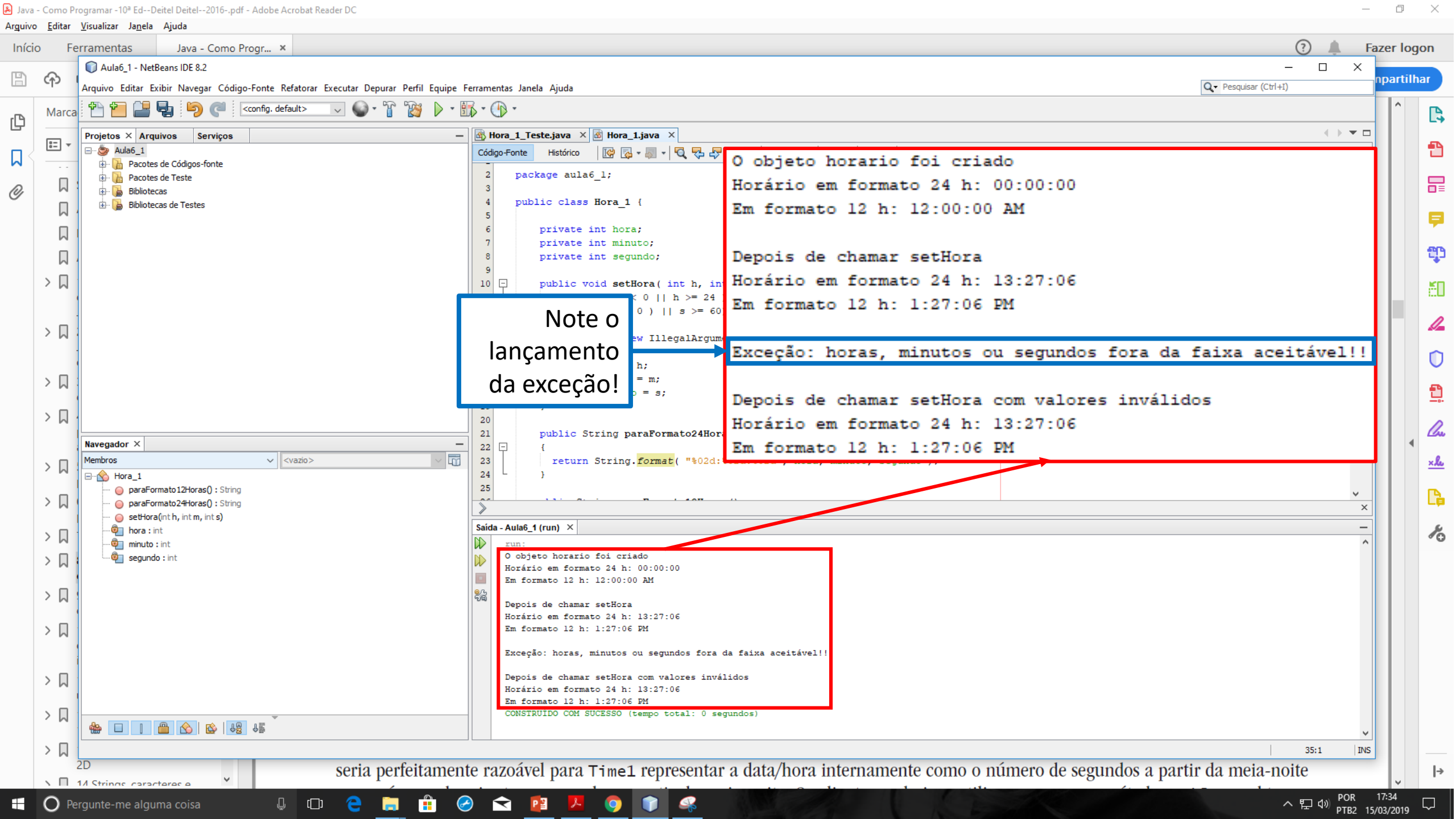
Classe Hora_1_Teste

Instancia um objeto Hora_1 com valores default

‘Seta” valores válidos para o objeto

Tenta setar valores fora da faixa para o objeto

```
public class Hora_1_Teste {
    public static void main(String[] args) {
        Hora_1 horario = new Hora_1();

        mostraHora("O objeto horario foi criado", horario);
        System.out.println();

        horario.setHora(13, 27, 6);
        mostraHora("Depois de chamar setHora", horario);
        System.out.println();

        try
        {
            horario.setHora(99, 99, 99);
        }
        catch (IllegalArgumentException e)
        {
            System.out.printf("Exceção: %s%n%n", e.getMessage());
        }

        mostraHora("Depois de chamar setHora com valores inválidos", horario);
    }

    private static void mostraHora(String cabecalho, Hora_1 t)
    {
        System.out.printf("%s%nHorário em formato 24 h: %s%nEm formato 12 h: %s%n",
            cabecalho, t.paraFormato24Horas(), t.paraFormato12Horas());
    }
}
```



seria perfeitamente razoável para `Time1` representar a data/hora internamente como o número de segundos a partir da meia-noite

Controlando o Acesso a Membros

- Os modificadores de acesso **public** e **private** controlam o acesso a variáveis e métodos de uma classe.
- O principal objetivo dos métodos **public** é apresentar aos clientes da classe uma visualização dos ***serviços*** que a classe fornece (isto é, a ***interface pública da classe***). Mas...
- Os clientes não precisam se preocupar com a forma como a classe realiza suas tarefas!
- Por essa razão, as variáveis **private** e os métodos **private** da classe (isto é, ***seus detalhes de implementação***) ***não*** **são** acessíveis aos clientes.

Controlando o Acesso a Membros

- Tente usar essa classe driver para a classe Hora_1 do exemplo anterior:

```
public class TesteDeAcesso {
    public static void main(String[] args) {
        Hora_1 h = new Hora_1();

        h.hora = 7;
        h.minuto = 15;
        h.segundo = 30;
    }
}
```

- Forçando a saída:

```
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - hora has private access in aula6_2.Hora_1
        at aula6_2.TesteDeAcesso.main(TesteDeAcesso.java:8)
C:\Users\rmesquita\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
FALHA NA CONSTRUÇÃO (tempo total: 6 segundos)
```

Uso da Referência **This**

- Cada objeto pode acessar uma ***referência a si próprio*** com a palavra-chave **this**.
- Quando um método é chamado para um objeto particular, o corpo do método utiliza ***implicitamente*** a palavra-chave this para referenciar as ***variáveis de instância*** do objeto e outros métodos.
- Isso permite que o código da classe saiba qual objeto deve ser manipulado.
- Podemos também usar a palavra-chave **this** explicitamente no corpo do método.
- Vamos programar...

Classe Testando_o_This

classe HoraSimples para demonstrar a referência "this"

se o construtor usa nomes de parâmetros idênticos aos nomes das variáveis de instância, é requerida a referência this para fazer a distinção entre os nomes!

```
public class Testando_o_This {
    public static void main(String[] args)
    {
        HoraSimples hora = new HoraSimples(15, 30, 19);
        System.out.println(hora.constroiString());
    }
}
class HoraSimples
{
    private int hora;
    private int minuto;
    private int segundo;

    public HoraSimples(int hora, int minuto, int segundo)
    {
        this.hora = hora; // seta a hora para "this" (este) objeto
        this.minuto = minuto; // seta o minuto para "this" (este) objeto
        this.segundo = segundo; // seta o segundo para "this" (este) objeto
    }

    public String constroiString()
    {
        return String.format("%24s: %s%n%24s: %s",
            "this.paraFormato24Horas()", this.paraFormato24Horas(),
            "     paraFormato24Horas()", paraFormato24Horas());
    }

    public String paraFormato24Horas()
    {
        return String.format("%02d:%02d:%02d",
            this.hora, this.minuto, this.segundo);
    }
}
```

Testando_o_This - NetBeans IDE 8.2

Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda

Pesquisar (Ctrl+I)

<config. default>

Projetos | Arquivos | Serviços

- Aula6_1
- Aula6_2
- Testando_o_This
  - Pacotes de Códigos-fonte
    - testando_o_this
      - Testando_o_This.java
  - Pacotes de Teste
  - Bibliotecas
  - Bibliotecas de Testes

Navegador

Membros <vazio>

- HoraSimples
  - HoraSimples(int hora, int minuto, int segundo)
  - constroiString() : String
  - paraFormato24Horas() : String
  - hora : int
  - minuto : int
  - segundo : int
- Testando_o_This
  - main(String[] args)

Hora_1_Teste.java | Hora_1.java | Hora_1.java | TesteDeAcesso.java | Testando_o_This.java

Código-Fonte | Histórico

```
2   package testando_o_this;
3
4   public class Testando_o_This {
5       public static void main(String[] args)
6       {
7           HoraSimples hora = new HoraSimples(15, 30, 19);
8           System.out.println(hora.constroiString());
9       }
10  }
11  class HoraSimples
12  {
13      private int hora;
14      private int minuto;
15      private int segundo;
16
17      public HoraSimples(int hora, int minuto, int segundo)
18      {
19          this.hora = hora; // seta a hora para "this" (este) objeto
20          this.minuto = minuto; // seta o minuto para "this" (este) objeto
21          this.segundo = segundo; // seta o segundo para "this" (este) objeto
22      }
23
24      public String constroiString()
25      {
26          return String.format("%24s: %s%n%24s: %s",
27              "this.paraFormato24Horas()", this.paraFormato24Horas(),
28              "     paraFormato24Horas()", paraFormato24Horas());
29      }
30
31      public String paraFormato24Horas()
32      {
33          return String.format("%02d:%02d:%02d",
34              this.hora, this.minuto, this.segundo);
35      }
36  }
37
38
39
```

this.paraFormato24Horas(): 15:30:19
     paraFormato24Horas(): 15:30:19

Saída - Testando_o_This (run)

```
run:
this.paraFormato24Horas(): 15:30:19
     paraFormato24Horas(): 15:30:19
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

39:1 INS

Pergunte-me alguma coisa

POR PTB2 18:11 15/03/2019

Sobrecarga de Construtores

- Permitem que objetos da classe sejam instanciados de diferentes maneiras.
- Para sobrecarregar construtores, simplesmente forneça múltiplas declarações de construtor com assinaturas diferentes.
- Vamos a um exemplo...

Classe Hora_2

```
public class Hora_2
{
    private int hora;
    private int minuto;
    private int segundo;

    public Hora_2()
    {
        this(0, 0, 0);
    }

    public Hora_2(int h)
    {
        this(h, 0, 0); // invoke constructor with three arguments
    }

    public Hora_2(int h, int m)
    {
        this(h, m, 0);
    }

    public Hora_2(int h, int m, int s)
    {
        if (h < 0 || h >= 24)
            throw new IllegalArgumentException("A hora deve estar entre 0-23");

        if (m < 0 || m >= 60)
            throw new IllegalArgumentException("Os minutos devem estar entre 0-59");

        if (s < 0 || s >= 60)
            throw new IllegalArgumentException("Os segundos devem estar entre 0-59");

        this.hora = h;
        this.minuto = m;
```

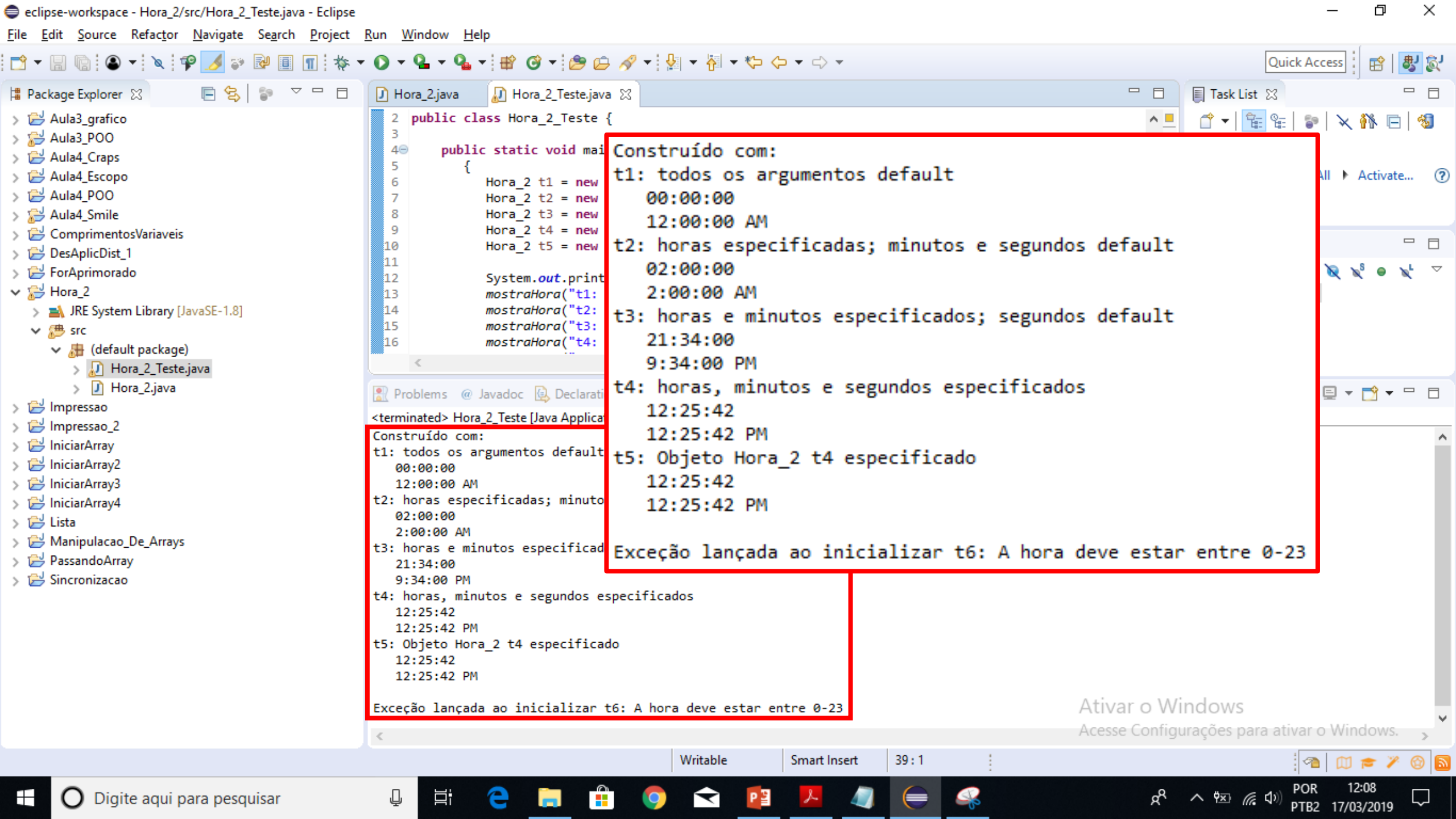
Classe Hora_2

```
    public int getHora()
    {
        return hora;
    }

    public int getMinuto()
    {
        return minuto;
    }

    public int getSegundo()
    {
        return segundo;
    }

    public String paraFormato24Horas()
    {
        return String.format(
            "%02d:%02d:%02d", getHora(), getMinuto(), getSegundo());
    }

    public String paraFormato12Horas()
    {
        return String.format("%d:%02d:%02d %s",
            ((getHora() == 0 || getHora() == 12) ? 12 : getHora() % 12),
            getMinuto(), getSegundo(), (getHora() < 12 ? "AM" : "PM"));
    }
}
```

Classe Hora_2_Teste

Classe driver

```
public class Hora_2_Teste {

    public static void main(String[] args)
       {
          Hora_2 t1 = new Hora_2();
          Hora_2 t2 = new Hora_2(2);
          Hora_2 t3 = new Hora_2(21, 34);
          Hora_2 t4 = new Hora_2(12, 25, 42);
          Hora_2 t5 = new Hora_2(t4);

          System.out.println("Construído com:");
          mostraHora("t1: todos os argumentos default", t1);
          mostraHora("t2: horas especificadas; minutos e segundos default", t2);
          mostraHora("t3: horas e minutos especificados; segundos default", t3);
          mostraHora("t4: horas, minutos e segundos especificados", t4);
          mostraHora("t5: Objeto Hora_2 t4 especificado", t5);

          try
          {
             Hora_2 t6 = new Hora_2(27, 74, 99);
          }
          catch (IllegalArgumentException e)
          {
             System.out.printf("%nExceção lançada ao inicializar t6: %s%n",
                e.getMessage());
          }
       }

       private static void mostraHora(String cabecalho, Hora_2 t)
       {
          System.out.printf("%s%n   %s%n   %s%n",
             cabecalho, t.paraFormato24Horas(), t.paraFormato12Horas());
       }
}
```


eclipse-workspace - Hora_2/src/Hora_2_Teste.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Aula3_grafico
Aula3_POO
Aula4_Craps
Aula4_Escopo
Aula4_POO
Aula4_Smile
ComprimentosVariaveis
DesAplicDist_1
ForAprimorado
Hora_2
JRE System Library [JavaSE-1.8]
src
(default package)
Hora_2_Teste.java
Hora_2.java
Impressao
Impressao_2
IniciarArray
IniciarArray2
IniciarArray3
IniciarArray4
Lista
Manipulacao_De_Arrays
PassandoArray
Sincronizacao
Hora_2.java
Hora_2_Teste.java
2 public class Hora_2_Teste {
3
4 public static void mai
5 {
6 Hora_2 t1 = new
7 Hora_2 t2 = new
8 Hora_2 t3 = new
9 Hora_2 t4 = new
10 Hora_2 t5 = new
11
12 System.out.print
13 mostraHora("t1:
14 mostraHora("t2:
15 mostraHora("t3:
16 mostraHora("t4:
Task List
Activate...
Problems Javadoc Declarati
<terminated> Hora_2_Teste [Java Applica
Construído com:
t1: todos os argumentos default
00:00:00
12:00:00 AM
t2: horas especificadas; minuto
02:00:00
2:00:00 AM
t3: horas e minutos especificad
21:34:00
9:34:00 PM
t4: horas, minutos e segundos especificados
12:25:42
12:25:42 PM
t5: Objeto Hora_2 t4 especificado
12:25:42
12:25:42 PM
Exceção lançada ao inicializar t6: A hora deve estar entre 0-23
Construído com:
t1: todos os argumentos default
00:00:00
12:00:00 AM
t2: horas especificadas; minutos e segundos default
02:00:00
2:00:00 AM
t3: horas e minutos especificados; segundos default
21:34:00
9:34:00 PM
t4: horas, minutos e segundos especificados
12:25:42
12:25:42 PM
t5: Objeto Hora_2 t4 especificado
12:25:42
12:25:42 PM
Exceção lançada ao inicializar t6: A hora deve estar entre 0-23
Ativar o Windows
Acesse Configurações para ativar o Windows.
Writable
Smart Insert
39 : 1
Digite aqui para pesquisar
POR
PTB2
12:08
17/03/2019

Composição

- Uma classe pode ter ***referências a objetos de outras classes como membros***.
- Trata-se de um **relacionamento *tem um*.**
- Vamos construir um exemplo de composição:
  - Uma classe Data – cujos objetos são datas no formato dia-mês-ano
  - Uma classe Empregado, que usa objetos Data.

Classe Data

```
public class Data {

 private int dia;
   private int mes;
   private int ano;

   private static final int[] dias_por_mes =
      {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

   public Data(int d, int m, int a)
   {
      if (d <= 0 ||
         (d > dias_por_mes[m] && !(m == 2 && d == 29)))
         throw new IllegalArgumentException("O dia (" + d +
            ") está fora da faixa especificada para mês e ano");

      if (m == 2 && d == 29 && !(a % 400 == 0 ||
           (a % 4 == 0 && a % 100 != 0)))
         throw new IllegalArgumentException("O dia (" + d +
            ") está fora da faixa especificada para mês e ano");

      if (m <= 0 || m > 12)
         throw new IllegalArgumentException(
            "O mês (" + m + ") deve estar entre 1 e 12");

      this.dia = d;
      this.mes = m;
```

Classe Empregado

```
public class Empregado {

 private String nome;
   private String sobrenome;
   private Data dataDeNascimento;
   private Data dataDeAdmissao;

   public Empregado(String nome, String sobrenome, Data dataDeNascimento,
      Data dataDeAdmissao)
   {
      this.nome = nome;
      this.sobrenome = sobrenome;
      this.dataDeNascimento = dataDeNascimento;
      this.dataDeAdmissao = dataDeAdmissao;
   }

   public String toString()
   {
      return String.format("%s, %s\n Data de Admissão: %s\n Data de Nascimento: %s",
         sobrenome, nome, dataDeAdmissao, dataDeNascimento);
   }
}
```

Classe Empregado_Teste

```
public class Empregado_Teste {

public static void main(String[] args)
    {
        Data nascimento = new Data(24, 7, 1949);
        Data admissao = new Data(12, 3, 1988);
        Empregado empregado = new Empregado("José", "Silva", nascimento, admissao);

        System.out.println(empregado);
    }

}
```

eclipse-workspace - Empregado/src/Empregado.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Explorer

- Aula3_grafico
- Aula3_POO
- Aula4_Craps
- Aula4_Escopo
- Aula4_POO
- Aula4_Smile
- ComprimentosVariaveis
- DesAplicDist_1
- Empregado
  - JRE System Library [JavaSE-1.8]
  - src
    - (default package)
      - Data.java
      - Empregado_Teste.java
      - Empregado.java
- ForAprimorado
- Hora_2
- Impressao
- Impressao_2
- IniciarArray
- IniciarArray2
- IniciarArray3
- IniciarArray4
- Lista
- Manipulacao_De_Arrays
- PassandoArray
- Sincronizacao

Hora_2.java | Hora_2_Teste.java | Data.java | *Empregado.java | Empregado_Teste.java

```
1
2  public class Empregado {
3
4      private String nome;
5      private String sobrenome;
6      private Data dataDeNascimento;
7      private Data dataDeAdmissao;
8
9      public Empregado(String nome, String sobrenome, Data dataDeNascimento,
10        Data dataDeAdmissao)
11     {
12        this.nome = nome;
13        this.sobrenome = sobrenome;
14        this.dataDeNascimento = dataDeNascimento;
15        this.dataDeAdmissao = da
16     }
17
18     public String toString()
19     {
20        return String.format("%s
21           sobrenome, nome, datal
22     }
23 }
24
25
```

Task List

Find | All | Activate...

Outline

- : String
- nome : String
- DeNascimento : Data
- DeAdmissao : Data
- egado(String, String, Data,
- ing() : String

Problems | Javadoc | Declaration | Console

<terminated> Empregado_Teste [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (17 de mar de 2019 12:50:37)

```
Construtor para o objeto Data 24/7/1949
Construtor para o objeto Data 12/3/1988
Silva, José
 Data de Admissão: 12/3/1988
 Data de Nascimento: 24/7/1949
```


```
Construtor para o objeto Data 24/7/1949
Construtor para o objeto Data 12/3/1988
Silva, José
 Data de Admissão: 12/3/1988
 Data de Nascimento: 24/7/1949
```


Writable | Smart Insert | 25 : 1

Digite aqui para pesquisar

POR PTB2 12:54 17/03/2019

Dúvidas?