Depois disto ouvi a voz do Senhor, que dizia:
A quem enviarei, e quem há de ir por nós?

Então disse eu: Eis-me aqui, envia-me a mim.

Isaías 6:8

Especialização em Java

Java I

- Prof. José Antonio Gonçalves

Ementa:

Conteúdo já visto

- **Orientação a Objetos em Java**: Classes, Objetos, Herança, Polimorfismo, Classes Abstratas, Interface;
- **Exceções**;
- **Manipulação de Texto e Strings**;
- **Componentes básicos de interface gráfica**;
-**Tratamento de Eventos**.

Nestes Slides:

- **Orientação a Objetos em Java:**

Exceções

Construindo nossas exceções e tratando-as

Contextualização (1/ 3)

As “Regras de Negócio” nos fornecem os subsídios para a construção de softwares. Por vezes elas apenas nos no dão parâmetros flexíveis, por outras, nos impõe condições que não permite transigências. **Por exemplo:**

- No Brasil a lei trabalhista não permite que alguém trabalhe por um salário abaixo do salário mínimo estipulado pelo Governo.

- Consoante a isso, ao construir um software de gestão de Recursos Humanos, não podemos permitir que se faça lançamentos de salário que infrinjam esta Lei. Caso isso ocorra a empresa poderá sofrer sanções legais.

Contextualização (2 / 3)

Considere que irá construir um aplicativo deste. Como criar regras internas no seu sistema que, além de mostrar o erro ao usuário (programador), permita que este trate o erro?

Com Java isto pode ser feito por meio da **criação e tratamento de exceções**.

Contextualização (3 / 3)

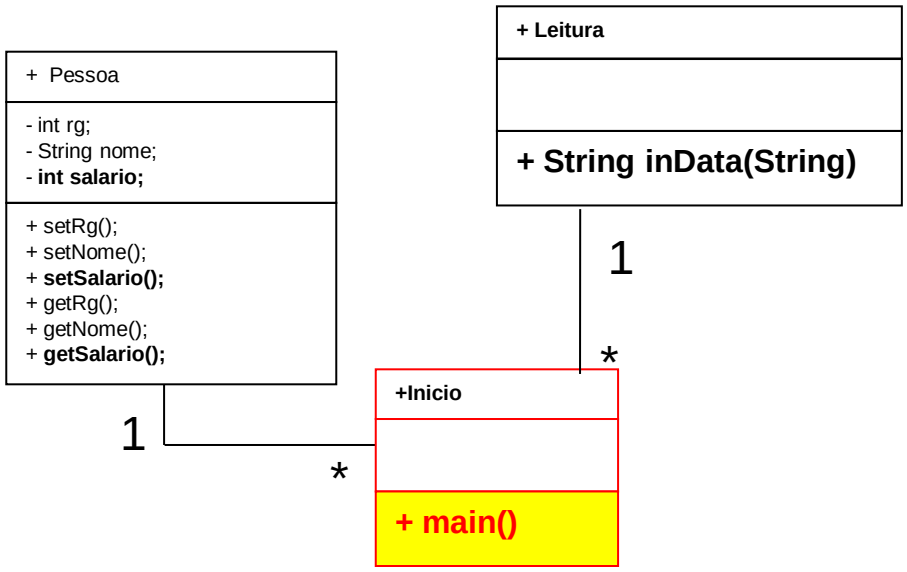
Quando for criar suas exceções, procure criá-las de forma que sejam do tipo “verificada”. Como vimos nas aulas anteriores exceções verificadas são aquelas que “estendem” a classe Exception e são verificadas em tempo de compilação

Exemplificação

Considere que deva construir um software que trate da questão salarial levantada anteriormente. Como podemos resolver?

Exemplificação

Observe o diagrama de classes a seguir. Perceba a existência do atributo salário na classe Pessoa.



Exemplificação

Nesta ordem, vamos construir:

- uma classe que será a exceção a ser lançada: **classe SalarioException** (que herda a classe Exception)**;**

- a regra de negócio que utilizará esta classe: **método setSalario() da classe Pessoa**;

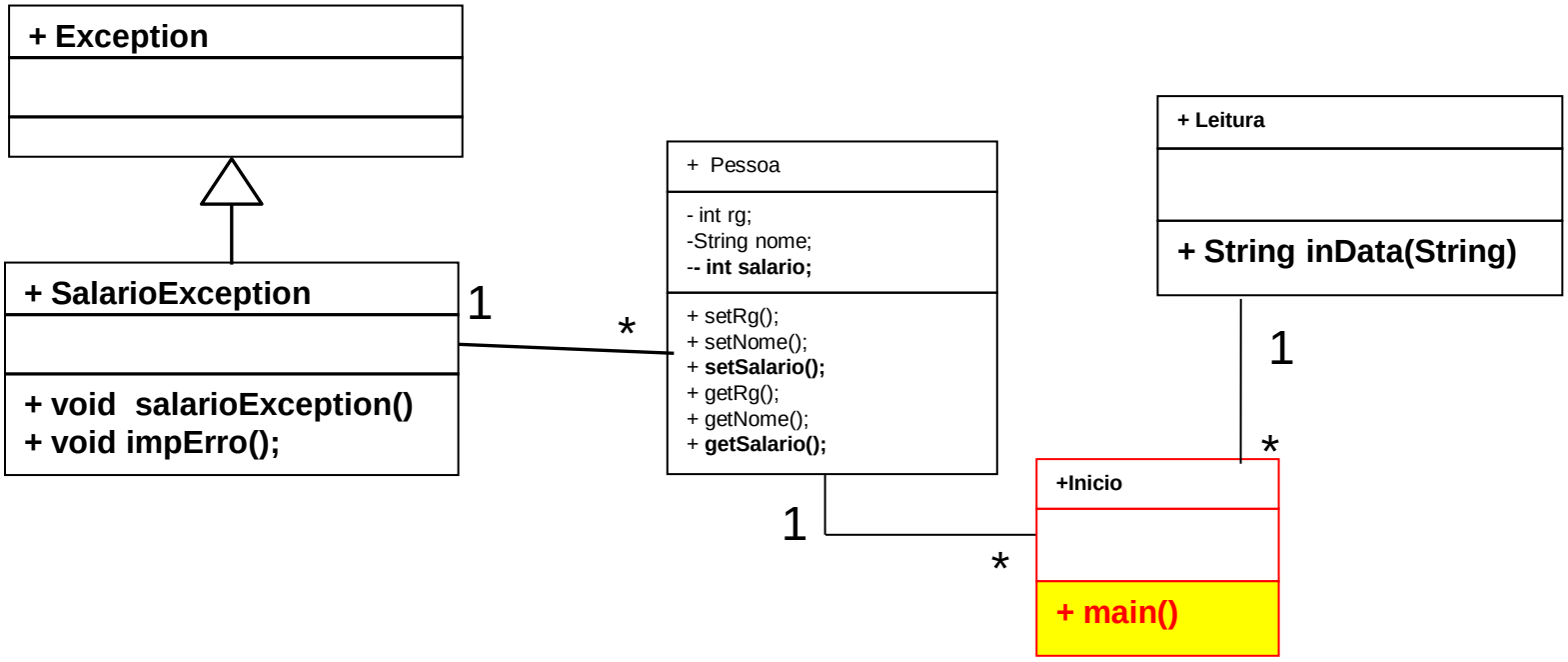
- condição de teste deste mecanismo: **instanciação de um tipo Pessoa, o objeto "p", na classe Inicio**, por meio do método **setSalario()**

Exemplificação

Observe que a classe de exceção (SalarioException) deve estar “associada” à classe que a disparará (Pessoa).

Que preferencialmente estenda a classe Exception.

Neste caso vamos utilizar o Método Construtor da classe SalarioException para apresentar seu uso e mais um método (impErro) somente para demonstrar que mais de um tratamento poderia ser implementado.

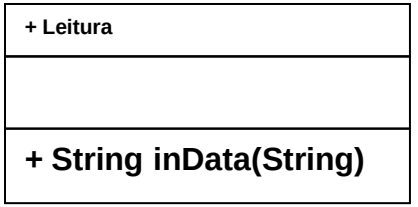


Exemplificação

A implementar a regra de negócio, vamos aproveitar o método setSalario().

Exemplificação

Para entrada de dados vamos usar a classe Leitura (*que recebe uma String e retorna uma String*)

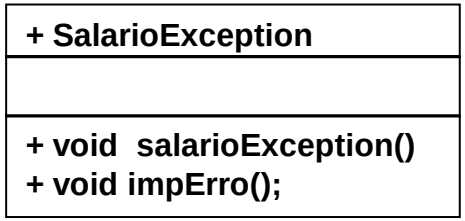



```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;

4.  public class Leitura{
5.          public static String inData(String rotulo){
6.                  InputStreamReader teclado = new InputStreamReader(System.in);
7.                  BufferedReader memoria = new BufferedReader(teclado);
8.                  System.out.print(rotulo);
9.                  String s = "";
10.                 try{
11.                         s = memoria.readLine();
12.                 }
13.                 catch(IOException e){
14.                         System.out.println("Erro de entrada");
15.                 }
16.                 return s;
17.         }
18. }
```

Exemplificação

classe SalarioException (que gerará o objeto da exceção que será “lançado”)




```
1.  public class SalarioException extends Exception{
2.      public SalarioException(){
3.          System.out.println("\n Método Construtor da Casse SalarioException");
4.      }
5.      public void impErro(){
6.          System.out.println("\n Salário abaixo do valor");
7.      }
8.  }
```

Exemplificação

classe Pessoa (*perceba as regra de negócio no método setSalario()*)

```
1.  public class Pessoa {
2.
3.   private int rg;
4.      private String nome;
5.      private int salario;

6.      public int getRg() {
7.          return rg;
8.      }
9.      public void setRg(int rg) {
10.         this.rg = rg;
11.     }
12.     public String getNome() {
13.         return nome;
14.     }
15.     public void setNome(String nome) {
16.         this.nome = nome;
17.     }
18.     public int getSalario() {
19.         return salario;
20.     }
21.     public void setSalario(int salario) throws SalarioException {
22.         int salarioMinimo = 750;
23.         if(salario >= salarioMinimo){
24.             this.salario=salario;
25.         }
26.         else{
27.             throw new SalarioException();
28.         }
29.     }
30. }
```

throws: indica que o método **pode disparar** uma exceção, que no caso é a SalarioException

throw: efetivamente dispara uma exceção, que no caso é a SalarioException

Detalhes da construção do método setSalario da classe Pessoa

```
21.  public void setSalario(int salario) throws SalarioException {
22.          int salarioMinimo = 750;
23.          if(salario >= salarioMinimo){
24.                this.salario=salario;
25.          }
26.          else{
27.                throw new SalarioException();
28.          }
29.  }
```

- Na construção de um método que pode disparar uma exceção deve-se explicitar isso em sua assinatura, adicionando a palavra ***throws*** (**com** "***s***") e a exceção que o método pode disparar, no caso SalarioException. A linha 21 apresenta esta codificação.

- Considerando que o método deve disparar a exceção, caso uma regra seja infringida, usamos ***throw*** (**sem** "s") para, efetivamente, disparar a exceção. A linha 27 demonstra isso.

- Na lógica aplicada à construção do método usou-se um "**IF/ELSE**" (linhas 22 a 27). Neste caso a exceção é disparada se o a proposição do IF não for verdadeira.

Exemplificação

classe Inicio (*perceba que o método setSalario, que pode disparar uma exceção, está "dentro" de um try e foi constuído um catch caso a exceção seja disparada*)

```
1.  public class Inicio {
2.      public static void main(String arg[]){
3.          Pessoa p = new Pessoa();
4.          Leitura l = new Leitura();
5.
6.          p.setRg(Integer.parseInt(l.inData("Entre com o RG: ")));
7.          p.setNome(l.inData("\n Entre com o NOME: "));
8.
9.           try{
10.             p.setSalario(Integer.parseInt(l.inData("\n SALARIO: ")));
11.          }
12.          catch(SalarioException se){
13.             System.out.println("\n Qual erro aconteceu? ");
14.             se.impErro();
15.          }
16.
17.         System.out.println("\n RG.....: "+p.getRg());
18.         System.out.println("\n NOME...: "+p.getNome());
19.         System.out.println("\n SALARIO: "+p.getSalario());
20.     }
21. }
```

Teste

Após executar aplicação comente o try/catch (como mostra o exemplo) e verifique que não compilará ou, se estiver usando uma IDE, o erro será apresentado, informando que não foi reportada nenhum exceção.

Isto ocorre porque na assinatura do método setSalario, na classe Pessoa, informou que o método pode disparar uma exceção do tipo verificada (a SalarioException que herda Exception).

```
1.  public class Inicio {
2.      public static void main(String arg[]){
3.          Pessoa p = new Pessoa();
4.          Leitura l = new Leitura();
5.
6.          p.setRg(Integer.parseInt(l.inData("Entre com o RG: ")));
7.          p.setNome(l.inData("\n Entre com o NOME: "));
8.
9.           //try{
10.             p.setSalario(Integer.parseInt(l.inData("\n SALARIO: ")));
11.         // }
12.          //catch(SalarioException se){
13.           //   System.out.println("\n Qual erro aconteceu? ");
14.           //   se.impErro();
15.         //  }
16.
17.         System.out.println("\n RG.....: "+p.getRg());
18.         System.out.println("\n NOME...: "+p.getNome());
19.         System.out.println("\n SALARIO: "+p.getSalario());
20.     }
21. }
```