Exceções

POO

Prof. Marcio Delamaro

Tratamento de exceções

- Já vimos como tratar exceções (quase)
- Já vimos como lançar uma exceção
  - Com um tipo de exceção já existente
- Não vimos como criar um novo tipo de exceção

O que é uma exceção

- É um objeto, como qualquer outro em Java
- https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html

- `public class Exception extends Throwable`

- *The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch.*

Construtores

`Exception()`

Constructs a new exception with null as its detail message.

`Exception(String message)`

Constructs a new exception with the specified detail message.

Instanciação

```
if ( condicao_erro)
     throw new IllegalArgumetException(“Posição
ocupada”);
```

Instanciação

```
if ( condicao_erro)
     throw new IllegalArgumetException(“Posição
ocupada”);
```

e depois

```
catch (Exception e) {
     system.out.println(e.getMessage());
     …
}
```

Criando

- Podemos personalizar nossas exceções
- Criar uma nova classe que estende Exception
- Criar uma classe que estende uma classe que herda de Exception

BozoException

```
public class BozoException extends Exception {



}
```

BozoException

```
public class BozoException extends Exception {
   public BozoException(String message) {
      super(message);
   }
}
```

Usando BozoException

```
public void add(int posicao, int[] dados) throws
BozoException {

     if ( taken[posicao-1] )
            throw new BozoException("Posição
ocupada no placar");
```

Usando BozoException

```
try {
        System.out.print("Escolha a posição
que quer ocupar com essa jogada ===> ");
        pos = EntradaTeclado.leInt();
        pl.add(pos, values);
}

catch (BozoException e) {
     pos = 0;
}
```

Checked X unchecked

- The class Exception and any subclasses that are not also subclasses of RuntimeException are checked exceptions.
- RuntimeException and its subclasses are unchecked exceptions.
- RuntimeException and its subclasses are unchecked exceptions.
- Unchecked exceptions do not need to be declared in a method or constructor's throws clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary.

Unchecked

```
public class BozoException extends
RuntimeException {

  ...

}
```

Unchecked

```
public class BozoException extends
RuntimeException {

  ...

}

// Classe Placar

public void add(int posicao, int[] dados) throws
BozoException {

     if ( taken[posicao-1] )

        throw new BozoException("Posição ocupada no
placar");
```

Unchecked

```
public class BozoException extends RuntimeException
{

  ...

}

// Classe Placar

public void add(int posicao, int[] dados)

{

   if ( taken[posicao-1] )

   throw new BozoException("Posição ocupada no
placar");
```

Checked

- É sempre bom ter ajuda do compilador para verificar as exceções
- As vezes pode ser preciso estender uma classe que é unchecked
- Nesse caso não tem muito jeito

Tratamento seletivo

- Em um trecho de código podem ocorrer vários tipos de exceção
- Para cada um deles é possível realizar um tratamento diferente
- Vários blocos catch
- Apenas um é executado

Tratamento seletivo

```
try {
    System.out.print("Escolha a posição que quer ocupar com
essa jogada ===> ");
    pos = EntradaTeclado.leInt();
    pl.add(pos, values);
}
catch (BozoException e) {
   pos = 0;
   System.out.println("Valor inválido. Posição ocupada ou
inexistente.");
}
catch (Exception e) {
    pos = 0;
    System.out.println("Valor digitado não é um número");
}
```

Tratamento seletivo

```
try {
    System.out.print("Escolha a posição que quer ocupar com
essa jogada ===> ");
    pos = EntradaTeclado.leInt();
    pl.add(pos, values);
}
catch (BozoException e) {
   pos = 0;
   System.out.println("Valor inválido. Posição ocupada ou
inexistente.");
}
catch (Exception e) {
    pos = 0;
    System.out.println("Valor digitado não é um número");
}
```

Tratamento seletivo

```
try {
    System.out.print("Escolha a posição que quer ocupar com
essa jogada ===> ");
    pos = EntradaTeclado.leInt();
    pl.add(pos, values);
}
catch (BozoException e) {
   pos = 0;
   System.out.println("Valor inválido. Posição ocupada ou
inexistente.");
}
catch (Exception e) {
    pos = 0;
    System.out.println("Valor digitado não é um número");
}
```

Tratamento seletivo

```
try {
    System.out.print("Escolha a posição que quer ocupar com
essa jogada ===> ");
    pos = EntradaTeclado.leInt();
    pl.add(pos, values);
}
catch (BozoException e) {
   pos = 0;
   System.out.println("Valor inválido. Posição ocupada ou
inexistente.");
}
catch (Exception e) {
    pos = 0;
    System.out.println("Valor digitado não é um número");
}
```

Tratamento seletivo

```
try {
    System.out.print("Escolha a posição que quer ocupar com
essa jogada ===> ");
    pos = EntradaTeclado.leInt();
    pl.add(pos, values);
}
catch (BozoException e) {
   pos = 0;
   System.out.println("Valor inválido. Posição ocupada ou
inexistente.");
}
catch (Exception e) {
    pos = 0;
    System.out.println("Valor digitado não é um número");
}
```

BozoException é subclasse de Exception

E finalmente...

- A estrutura de tratamento de exceções tem uma forma de sempre executar código para finalizar o tratamento
- Estrututa finally

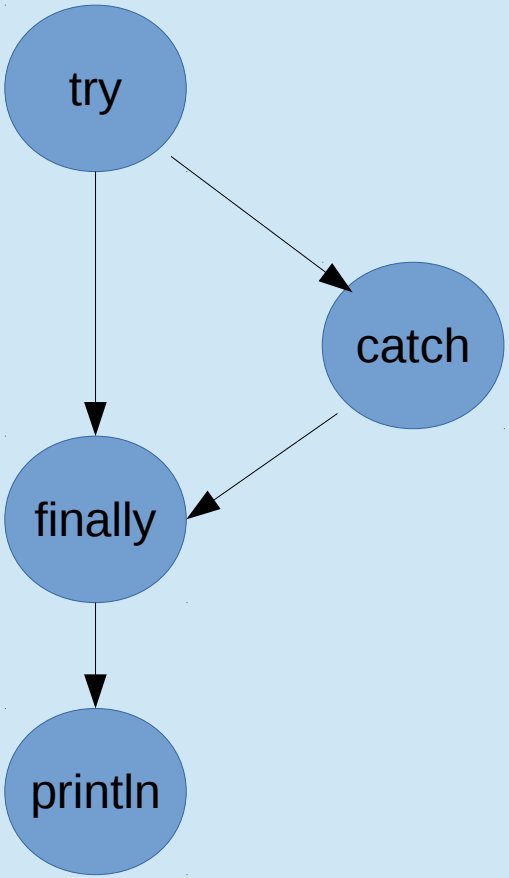
```
try { ... }
catch {... }
catch {... }
finally {...}
```

Usando finally

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

  try {
    int k = EntradaTeclado.leInt();
  }
  catch (NumberFormatException | IOException e) {
    e.printStackTrace();
  }
  finally {
    System.out.println("Não importa se houve ou não
exceção");
  }
  System.out.println("Mas não é sempre que passa aqui");
}
```

Execução



Usando finally

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

  try {
    int k = EntradaTeclado.leInt();
  }
  catch (NumberFormatException | IOException e) {
    e.printStackTrace();
    return;
  }
  finally {
    System.out.println("Não importa se houve ou não exceção");
  }
  System.out.println("Mas não é sempre que passa aqui");
}
```

Conclusão

- Não importa o que tenhamos nos blocos try/catch
- O bloco finally sempre é executado depois deles

Aproveitando

- Um pacote em Java é uma forma de organizar classes que têm funcionalidades relacionadas
- Cada pacote possui um subdiretório dentro da estrutura de diretórios do projeto
- package ssc103.bozo.main;
- Cria um diretório
  - ssc103
    - bozo
      - main

Declarando

- Primeira linha no arquivo da classe
- ```
  package ssc103.bozo.main;
  package ssc103.bozo.dados;
  ```
- Classes dentro do mesmo pacote podem acessar umas às outras
- Classes em outros pacotes precisam ser importadas
- ```
  import ssc103.bozo.dados.Placar;
  import ssc103.bozo.dados.RolaDados;
  import ssc103.bozo.exception.BozoException;
  import ssc103.bozo.util.EntradaTeclado;
  ```
- ```
  import ssc103.bozo.dados.*;
  ```
  ```

Praticando

- Modifique o seu projeto Bozo, criando pacotes e colocando as suas classes dentro dos pacotes.

- Modifique a classe EntradaTeclado para que ela, em caso de algum erro, lance sempre uma exceção "`ETException`" que é **unchecked**. Mude o seu projeto Bozo para usar essa nova versão da classe EntradaTeclado.