Linguagem de Programação I

Prof. Edson Kaneshima

Tratamento de Exceções

Aula 07

Tratamento de Exceções

- Exceção representa uma situação que normalmente não ocorre e algo inesperado no programa.

- O tratamento de uma exceção permite que o programa execute uma determinada lógica caso ocorra a exceção. Permite também que o programa continue em execução.

- Exemplos:
  - ArithmeticException – divisão por zero.
  - ArrayIndexOutOfBoundsException – índice inexistente no array.

Tratamento de Exceção

```
try {
        //bloco que é monitorado para erros
}
catch (TipoDaException exception) {
        // tratamento do erro
}
```

Tratamento de Exceção

Exemplo:

```
try{
        aux[i] = arrA[i] / arrB[i];
}
catch (ArithmeticException e) {
        System.out.println("Divisão por zero");
}
```

Tratamento de várias exceções

```
try {
        //bloco que é monitorado para erros
} catch (TipoDaException-1 exception) {
        // tratamento do erro
} catch (TipoDaException-2 exception) {
        // tratamento do erro
} catch (TipoDaException-N exception) {
        // tratamento do erro
}
```

Prioridade do catch é a ordem de captura.

Tratamento de várias exceções

```
try{
        System.out.print("A: ");
        a[i] = teclado.nextInt();
        System.out.print("B: ");
        b[i] = teclado.nextInt();
        c[i] = a[i] / b[i];
}catch (ArithmeticException e){
        System.out.println("Divisão por zero");
}catch (InputMismatchException e){
        System.out.println("erro na entrada de dados");
}
```

Capturando a exceção genérica

```
try{
        System.out.print("A: ");
        a[i] = teclado.nextInt();
        System.out.print("B: ");
        b[i] = teclado.nextInt();
        c[i] = a[i] / b[i];
} catch (Exception e){
            System.out.println(“Ocorreu um erro");
}
```

try..catch..finally

```
try {
        //bloco que é monitorado para erros
} catch (TipoDaException-1 exception) {
        // tratamento do erro
} catch (TipoDaException-2 exception) {
        // tratamento do erro
} finally {
        // bloco que será sempre executado, independente se
        // houve ou não exception e se ela foi tratado ou não
}
```

try..catch..finally

```
try{
        String arquivo = "d://Unifil//Linguagem1-2017//Arquivo.txt";
        arqentrada = new BufferedReader(new FileReader(arquivo));
        String linha;
        while ((linha = arqentrada.readLine()) != null) {
                System.out.println(linha);
        }
}catch (IOException e) {
       System.out.println("erro no arquivo");
}finally{
        if (arqentrada != null)
         try {
            arqentrada.close();
         }catch (IOException ex) {
            System.out.println("erro no fechamento do arquivo");
         }
}
```

Tratamento de Exceções – classe Throwable

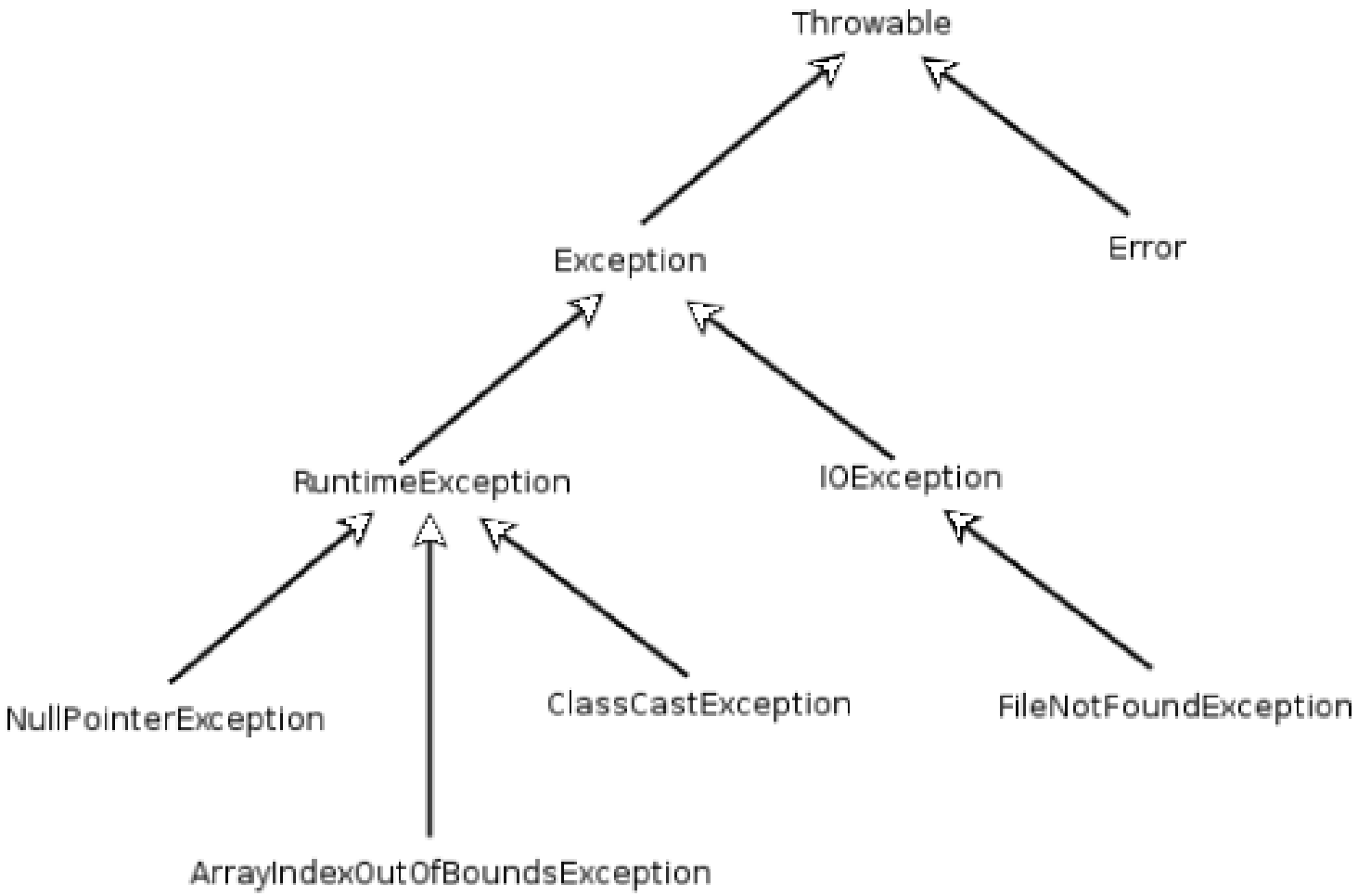
- Throwable – Super classe de todas as exceptions.
- Métodos:
  - getMessage() – retorna a descrição do erro, exemplo: / by zero.
  - printStackTrace() – imprime o stack trace do erro, exemplo:

```
java.lang.ArithmeticException: / by zero
at DivisaoArray.main(DivisaoArray.java:19)
at __SHELL12.run(__SHELL12.java:6)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
    java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessor
    Impl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at bluej.runtime.ExecServer$3.run(ExecServer.java:743)
```

Tratamento de Exceções – classe Throwable

```
try{
        System.out.print("A: ");
        a[i] = teclado.nextInt();
        System.out.print("B: ");
        b[i] = teclado.nextInt();
        c[i] = a[i] / b[i];
} catch (Exception e){
            System.out.println (e.getMessage());
            e.printStackTrace();
}
```

Hierarquia de Exceções (visão parcial)



Cláusula throws

- Para indicar que algum código no corpo do seu método pode lançar uma exceção, basta acrescentar a palavra chave throws depois da assinatura do método, com o nome ou nomes das exceções que seu método pode lançar:

```
public void carregaArray(int arr[]) throws InputMismatchException

public void carregaArray(int arr[]) throws InputMismatchException,
                                           ArrayIndexOutOfBoundsException
```

- A cláusula throws oferece informações sobre exceções em potencial e permite que o java se certifique que as outras pessoas estejam usando o método corretamente, torna explícito todos os locais onde condições excepcionais devem ser tratadas.

Cláusula throws

- throws propaga a exceção, ou seja, passa a responsabilidade para quem for utilizar o método tratar a exceção. Chamamos de checked exception, pois o compilador irá checar se ela está sendo tratada.

Trecho da classe que implementa o método com throws:

```
public void carregaArray(int arr[]) throws InputMismatchException  {
      for(int i=0; i<arr.length; i++){
         System.out.println("Informe Numero: ");
         arr[i] = teclado.nextInt();
      }
}
```

Trecho da classe que chama o método carregaArray():

```
try{
    ent.carregaArray(a);
}catch (InputMismatchException e){
    System.out.println(e.getMessage());
}
```

Lançando exceções - throw

- throw cria um novo objeto de exceção que é lançada.

```
public void saca(double valor) {
    if (valor > this.saldo) {
        throw new IllegalArgumentException("Saldo Insuficiente");
    } else {
        this.saldo = this.saldo - valor;
    }
}
```

Lançando exceções - throw

- O método getMessage() definido na classe Throwable retorna a mensagem que passamos no construtor da IllegalArgumentException.

```
try{
        minhaConta.saca(1000);
}catch (IllegalArgumentException e){
        System.out.println(e.getMessage());
}
```

Criando sua própria exceção

- Embora exista uma grande quantidade de exceções na biblioteca de classes java, pode se criar suas próprias exceções para lidar com os diferentes tipos de erros com que seus programas se deparam.

```
public class SaldoInsuficienteException extends RuntimeException {

        SaldoInsuficienteException(String message)
        {
            super(message);
        }
}
```

Criando e lançando sua própria exceção

- throw cria um novo objeto de exceção que é lançada.

```
public void saca(double valor) {
    if (valor > this.saldo) {
        throw new SaldoInsuficienteException("Saldo Insuficiente");
    } else {
        this.saldo = this.saldo - valor;
    }
}
```