Curso de Java

aula 08

Prof. Anderson Henrique

TECH
soluções em ti

Tratamento de Exceções

Este é um recurso para construir programas robustos e tolerável a falhas na linguagem Java. Uma exceção é uma indicação de um problema que ocorre durante a execução de um programa.

O nome exceção significa que o problema não ocorre frequentemente, ou seja, é uma exceção a regra. Com o tratamento de exceções, um programa pode continuar executando em vez de encerrar.

```
Ex.: int x = 1 / 0; // não é uma operação válida
Scanner s = new Scanner(System.in);
System.out.print (“Número: ”);
int a = s.nextInt( );
System.out.print (“Divisor: ”);
int b = s.nextInt( );
//InputMismatchException string
//ArithmeticException
System.out.println(a / b);
```

```
Scanner s = new Scanner(System.in);
boolean continua = true;
do{
        try{
                System.out.print (“Número: ”);
                int a = s.nextInt( );
                System.out.print (“Divisor: ”);
                int b = s.nextInt( );
                System.out.println(a / b);
                continua = false;
        }catch(InputMismatchException e1){
                System.out.println(“Erro: Numeros devem ser inteiros”);
                s.nextLine( ); //descarta a entrada que deu erro
        }
```

```
        catch(ArithmeticException e2){
                System.out.println("Erro: Divisor deve ser diferente de zero");
        }
while(continua);
```

O finally é um bloco que é executado, independente se teve ou não erro dentro do bloco.

```
Ex.: finally{
        System.out.println("Finally executado");
}
```

Exercício

Quando criamos um array, definimos o tipo de dados que os elementos do array deverá ter, e o tamanho desse array. Quando percorremos o array com o laço de repetição e procuramos índices que não existem nesse array, a linguagem Java lança a seguinte exceção: ArrayIndexOutOfBoundsException.

Crie uma estrutura de tratamento para esta exceção específica.

Multicatch, é a possibilidade de tratar vários tipos de exceções em um mesmo bloco (Java7).

```
Ex.: catch(InputMismatchException | ArithmeticException e1){
        System.out.println("Número Inválido");
        s.nextLine( );
}
```

stackTrace, é a sua pilha de erros. Um erro pode começar em uma determinada classe e pode propagar entre várias classes.

```
Ex.: catch(InputMismatchException | ArithmeticException e1){
        System.out.println("Número Inválido");
        e1.printStackTrace; //imprime a pilha de erros na exceção
        s.nextLine( );
}
```

Ex.: catch(InputMismatchException | ArithmeticException e1){

System.out.println(“Número Inválido”);

e1.**getTrace( )**; //retorna array com a pilha de erros na exceção

s.nextLine( );

}

Ex.: catch(InputMismatchException | ArithmeticException e1){

System.out.println(“Número Inválido”);

e1.**getMessage( )**; //retorna mensagem da pilha de erros na exceção

s.nextLine( );

}

A cláusula **throws**, o tratamento de exceções pode ser lançado sem eventualmente tratar essas exceções.

Ex.: public static void dividir(Scanner s) throws InputMismatchException, ArithmeticException {

```
System.out.print (“Número: ”);
int a = s.nextInt( );
System.out.print (“Divisor: ”);
int b = s.nextInt( );
System.out.println(a / b);
```

}

Exceções Comuns

NullPointerException, quando tentamos acessar um atributo de um objeto Null.

Ex.: static int[ ] arrayNull;

System.out.println(arrayNull.length);

ArithmeticException, quando tentamos realizar operações aritméticas impossíveis.

Ex.: int x = 5 / 0;

ArrayIndexOutOfBoundsException, quando tentamos acessar uma posição [índice] que não existe no array.

ClassCastException, quando tentamos realizar um cast de um tipo de objeto que não passa no teste do (É Um).

NumberFormatException, quando tentamos converter uma String em um número, a String está no formato desconhecido da classe de conversão.

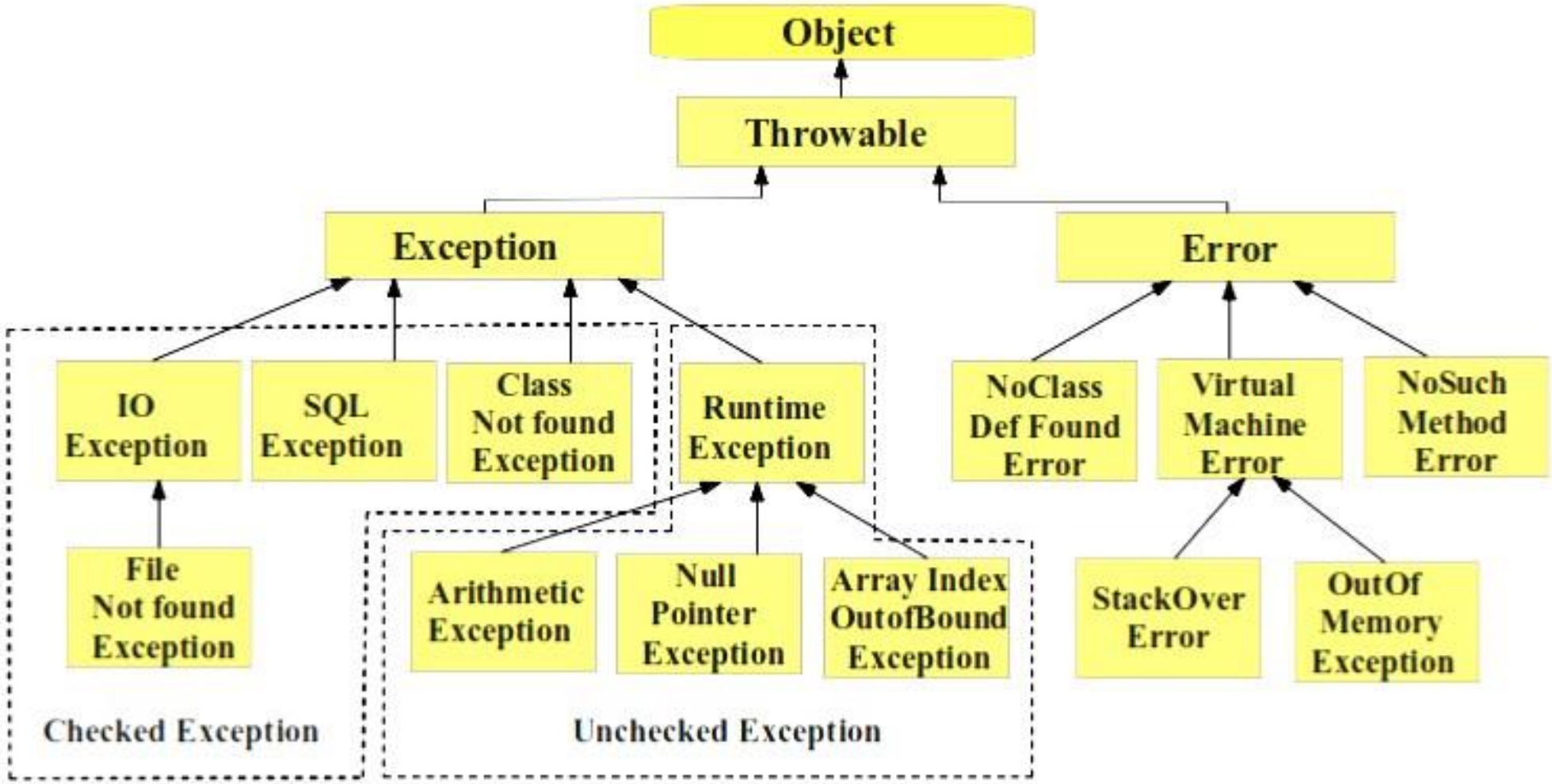
Ex.: int i = Integer.parseInt(“X”);

double d = Double.parseDouble(“Y”);

Exceções – Hierarquia de classes

Vamos conhecer alguns tipos de exceções e construir a nossa própria exceção. Todas as exceções são objetos, elas devem se estender da classe **Throwable**, que por sua vez se estende da classe **Object**.

Por convenção, as exceções que devemos criar, devem se estender da classe **Exception**, que é uma subclasse de **Throwable**. Essas exceções são verificadas.



Exercício

Construir uma exceção que verifica a senha de um usuário e retorna se ele foi autenticado ou se deu erro na autenticação.

Ex.: public class SenhaInvalidaException extends Exception{

```
public SenhaInvalidaException(String mensagem){
        super(mensagem);
}
```

}

Como testamos essa exceção?

Ex.: public class SenhaTeste{

```
public static void autenticar(String senha){
        if("123".equals(senha)){
                System.out.println("Autenticado");
        }else{
                throw new SenhaInvalidaException("Senha Incorreta");
        }
}
```

```
public static void main(String[ ] args){
        try{
                autenticar(“123”);
        }catch(SenhaInvalidaException e){
                e.printStackTrace( );
                System.out.println(e.getMessage( ));
        }
}
```

Prosseguiremos no próximo slide...

Professor: Anderson Henrique

Programador nas Linguagens Java e PHP