Programação II

Prof. Alcemir Rodrigues Santos

Tratamento de Exceções

Agenda

- O que são Exceções?
- Exceções verificáveis e não-verificáveis
- Blocos try&&catch
- Extendendo Exception

O que são exceções (Exception)?

- Uma exceção é um evento que interrompe o fluxo normal de processamento de uma classe. Esse evento é um erro de algum tipo e isso causa o término anormal da dasse.
- Quando programamos nossos sistemas desejamos que funcionem como esperado, no entanto, problemas acontecem mas eles são as exceções e não a regra.
- Portanto tratar estas exceções é fundamental para garantirmos a continuidade da execução dos nossos programas.
- Desta forma tornamos nossos programas mais robustos e tolerantes a falhas.

Throwable

- Em Java você só pode lançar e capturar objetos de tipo (ou subtipo) de `Throwable`.
- As únicas subclasses fornecidas pela API Java são `Error` and `Exception`.
- `Throwable` é considerada uma **exceção verificável**.

Error

- Um Error é lançado para indicar um problema sério que a aplicação não deveria tentar resolver.
- Error e todas as suas subclasses são tidas como **exceções não-verificáveis**
- Exemplos
  - StackOverflowError, tipicamente devido recursão infinita
  - OutOfMemoryError
  - AssertionError
  - NoClassDefFoundError, tipicamente devido a erro de configuração do classpath
  - NoSuchMethodError/NoSuchFieldError, tipicamente devido a uma versão errada da classe sendo carregada

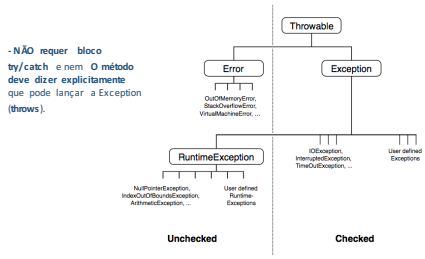
Exception

- Exceptions são utilizadas para condições que a aplicação pode querer capturar
- Exception e todas as suas subclasses (exceto RuntimeException e suas subclasses) são tidas como exceções **verificáveis**
- Exemplos
  - InterruptedException, thread foi interrompida enquanto executava uma chamada de bloco
  - IOException, supertipo de exceção para todos os erros relacionados a I/O
  - EOFException
  - FileNotFoundException
  - UnknownHostException, cheque sua conexão com a internet :)

RuntimeException

- A RuntimeException é uma subclasse de Exception.
- Ela é especial porque é um tipo de **exceção não-verificável**
- RuntimeExceptions são utilizadas para condições que uma aplicação geralmente não captura, tais como erros de programação
- **Exemplos**
  - ClassCastException, cast inválido
  - NullPointerException
  - ArithmeticException, tipicamente devido a divizão por zero
  - IndexOutOfBoundsException

Exceptions Verificáveis e Não-Verificáveis



- NÃO requer bloco **try/catch** e nem **O método deve dizer explicitamente** que pode lançar a Exception (**throws**).

- Requer bloco **try/catch** para capturar uma Exception lançada.

- **O método deve dizer explicitamente** que pode lançar a Exception (**throws**).

Exceções verificáveis

Não compila!

```
class Example {
    void method() {
        throw new Exception();
    }
}
```

"erro: Exception deve ser capturada ou declarada para ser lançada."

```
class Example {
    void method() throws Exception {
        throw new Exception();
    }
}
```

Compila!

```
class Example {
    void method() {
        try {
            throw new Exception();
        } catch (Exception e) {
            // ...
        }
    }
}
```


Exceções não-verificáveis

```
class Example {
    void method() {                    // No throws declaration
        throw new RuntimeException();  // No try/catch
    }
}
```


Exemplo real de um bloco try & catch

```
1// Arquivo: DivisaoZero.java
2// A classe DivisaoZero
3
4import java.util.Scanner;
5
6public class DivisaoZero {
7    public static void main(String[] args)
8    {
9        Scanner scanner = new Scanner(System.in);
10       float result;
11       int a, b;
12
13       try {
14           a = scanner.nextInt();
15
16           b = scanner.nextInt();
17
18           result = divisao(a, b);
19
20           System.out.print(result);
21       }
22       catch(ArithmeticException exception)
23       {
24           System.err.printf("Argumentos inválidos. " +
25                   "Divisão por zero.");
26       }
27   }
28
29   public static float divisao(int a, int b)
30   {
31       float result;
32
33       result = (float)a / b;
34
35       return result;
36   }
37}
```

Blocos try & catch

- Neste exemplo utilizamos blocos try e catch. Colocamos entre as chaves do bloco try a porção de código que queremos proteger bem como a fonte de uma possível exceção.

- No bloco catch, incluímos o código que será executado se um problema aritmético, como a divisão por zero, ocorrer.

- Dentro do bloco „try" realizamos a leitura de dois inteiros, a e b, e passamos como parâmetros para o método divisão. Este é o ponto do código onde poderá ocorrer uma falha. Observe que a operação de divisão está dentro da função divisão. A porção de código que estamos protegendo é a exibição do valor da divisão.

Blocos try & catch

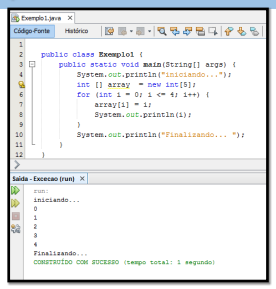
- No bloco catch, apenas exibimos uma mensagem para o usuário informando o motivo dafalha na execução do programa.
- Quando executamos este programa e informamos os valores corretos, nenhum problema ocorre e ele pode ser finalizado sem problemas.

Captura de Exceções

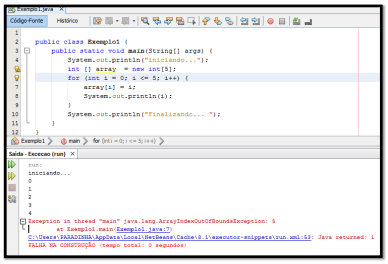
```
try {
  ...
}catch (Exceção1 varException1)  {
  ...
}catch (Exceção2 varException2)  {
  ...
}... {
}catch (Exception  e) {
  ....
}
```

```
try {
    System.out.println("digite dois numeros: ");
    a = scanner.nextInt();
    b = scanner.nextInt();
    result = divisao(a, b);
    System.out.println("O resultado é "+ result);
} catch (ArithmeticException exception) {
    System.err.printf("O valor de b não pode ser z
}
```

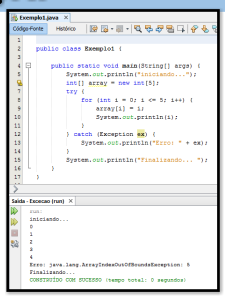
Exemplo Exceções



Exemplo Exceções



Exemplo Exceções



Extendendo Exception

```
public class MinhaExceca o extends Exception {
        public MinhaExceca o( ) {}
        public MinhaExceca o( St ri ng message) {super(messa ge ); }
        public MinhaExceca o( Th ro wa bl e cause) {super(caus e) ; }

        public MinhaExceca o( St ri ng message, Throwable cause) {
                super(messa ge , cause);
        }

        public MinhaExceca o( St ri ng message, Throwable cause, boolean
enableSuppr es si on, boolean writableStac kT ra ce ) {
                super(messa ge , cause, enableSuppr es si on , writableSta ck Tr ac e);
        }
}
```