Alisson RS

www.alissonrs.com.br

Moodle

Linguagem de programação II

Exceções em Java

...

Introdução

- Mundo ideal: dados estão sempre na forma certa, arquivos desejados sempre existem, etc.
- Mundo real: dados ruins e bugs podem arruinar o programa.
- Necessidade de mecanismos para tratamento de erros.


TODA REGRA
TEM UMA EXCEÇÃO.
E ISSO É UMA REGRA.
LOGO...
ISSO QUER DIZER QUE...
TAMBÉM TEM UMA EXCEÇÃO.
NEM TODA REGRA TEM UMA EXCEÇÃO!
Taverna
www.imagemwhats.com.br

Introdução

```c
#include <stdio.h>

int main() {
    int n = 4000000000;
    printf("%d\n", n); /
    return 0;
}
```

return 0 - sem erro.

return 1 - com erro.

Cada valor representando um tipo de erro.

Antes da POO:

- Variável global inteira com valores de 0 até n.
- A ocorrer uma exceção:
  - Variável assumia um valor.
  - Remetia uma mensagem de erro.
  - Encerrava o programa.

Introdução

- Depois da POO:
  - Classes de erros.
  - Possíveis tipos de erros e seus tratamentos são agrupados.
  - Não há necessidade de interromper o programa.
  - O mesmo erro é tratado quantas vezes for necessário.

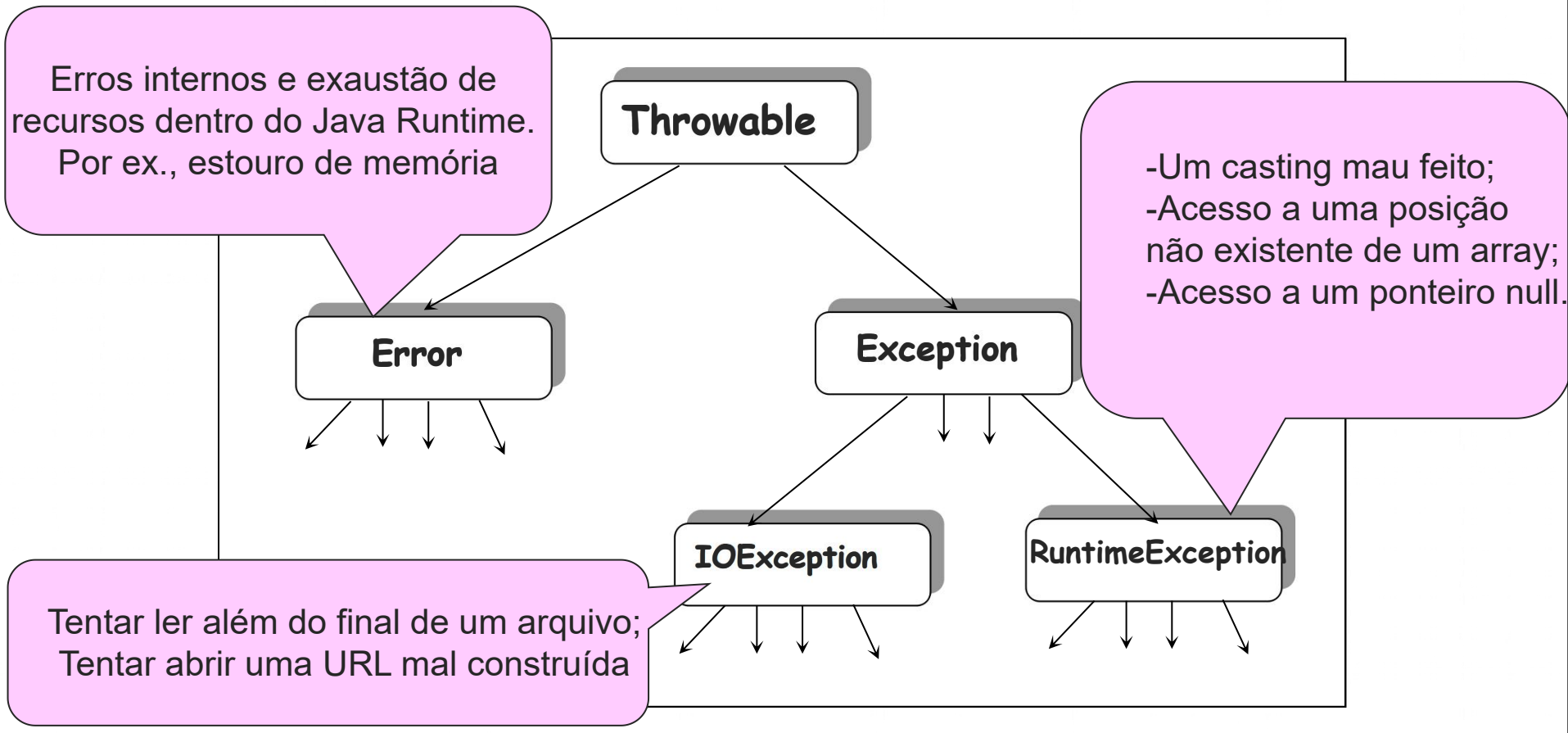
Introdução

- Idéia básica:
  - “código ruim não será executado”.
- Nem todos os erros podem ser detalhados em tempo de compilação.
- Os que não podem devem ser tratados em tempo de execução são o alvo da manipulação de exceções.

Introdução

- Premissa básica:
  - separar o processamento normal da manipulação de erros.
- Vantagens desse mecanismo:
  - Permite concentrar em lugares diferentes o “código normal” do tratamento do erro.
  - Simplifica a criação de programas grandes usando menos código.
  - Torna o código mais robusto
    - ao garantir que não há erros sem tratamento.

Hierarquia de exceções de Java



Introdução

- Exceção:
  - problema que impede a continuação do método ou escopo em execução.
- Importante:
  - exceção != problema normal.
- Problema normal:
  - há informação suficiente no contexto atual para lidar com ele.
- Exceção:
  - não há informação suficiente.
- Disparar uma exceção:
  - sair do contexto atual e relegar a solução a um contexto mais abrangente.

Exceção - Funcionamento

- Ao disparar-se uma exceção:
  - Um objeto exceção é criado.
  - A execução é interrompida.
  - O mecanismo de manipulação de exceções assume o controle
    - procura o manipulador de exceção adequado.
  - O manipulador da exceção trata o problema.

Exceção - Funcionamento

- Exemplo:
  - seja t uma referência para um objeto, que pode não ter sido inicializado.

- if (t == null)
  throw new NullPointerException();

- A palavra chave throw dispara uma exceção
  - dá início à sequência de eventos citada anteriormente.

Exceção - Funcionamento

- Outra versão:

- ```
  if (t == null)
    throw new NullPointerException
          ("t = null");
  ```

- Este construtor permite colocar informações pertinentes na exceção, que posteriormente podem ser extraídas usando outros métodos.
  ```

Exceção - Funcionamento

- Em resumo, disparar uma exceção é fácil:
  - 1) Escolha uma classe de exceção apropriada.
  - 2) Instancie um objeto dessa classe.
  - 3) Dispare-o.

```
throw new EOFException();
```

(3) **(2)** **(1)**

Capturando Exceção

- Quando uma exceção é disparada, em algum lugar ela deve ser capturada.

- Região protegida:
  - trecho de código que pode gerar exceções.

- Manipuladores de exceções:
  - tratam as exceções que ocorreram dentro da região protegida.
  - Vêm imediatamente após a mesma.

Tratamento de exceção em Java

- Palavras Chaves:
  - try
  - catch
  - finally
  - throws
  - throw

Capturando Exceção

- Em Java:
  - `try`:
    - indica a região protegida.
  - `catch`:
    - manipula uma exceção.

Formato básico:

```
try {
    // Código
} catch (ClasseDeExceção e) {
    // Manipula aquele tipo de erro
}
```

Capturando Exceção

- Pode-se usar vários manipuladores:

```
try { ...
}
catch(ClasseDeExcecao1 c1) { ...
}
catch(ClasseDeExcecao2 c2) { ...
}
catch(ClasseDeExcecao3 c3) { ...
}
...
```

Capturando Exceção

- Processo:
  - A exceção é disparada dentro de um bloco try.
  - O mecanismo de manipulação de exceção procura o primeiro catch cujo tipo de exceção bata com a exceção disparada.
  - O mecanismo entra no bloco do catch e o erro é tratado.

Tratamento de exceção em Java

- Exemplo de código:

```
1  package br.com.alissonrs.aula7;
2
3  public class Exemplo1_Excecao {
4
5      public static void main(String[] args) {
6
7          int[] array = new int[10];
8
9          for (int i = 0; i <= 10; i++) {
10             array[i] = i;
11             System.out.println(i);
12         }
13         System.out.println("Fim!");
14
15     }
16
17 }
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10

Seu programa será encerrado aqui e a linha 13 nunca será executada.

Tratamento de exceção em Java

- Exemplo de código:

```
1  package br.com.alissonrs.aula7;
2
3  public class Exemplo2_Excecao {
4
5      public static void main(String[] args) {
6
7          int[] array = new int[10];
8          try {
9              for (int i = 0; i <= 10; i++) {
10                 array[i] = i;
11                 System.out.println(i);
12             }
13         } catch (Exception e) {
14             System.out.println("Erro" + e.getMessage());
15         } finally {
16             System.out.println("Fim!");
17         }
18
19     }
20
21 }
```

Tratamento de exceção em Java

- Exemplo de código:

```
1  package br.com.alissonrs.aula7;
2
3  public class Exemplo2_Excecao {
4
5      public static void main(String[] args) {
6
7          int[] array = new int[10];
8          try {
9              for (int i = 0; i <= 10; i++) {
10                 array[i] = i;
11                 System.out.println(i);
12             }
13         } catch (Exception e) {
14             System.out.println("Erro" + e.getMessage());
15         } finally {
16             System.out.println("Fim!");
17         }
18
19     }
20
21 }
```

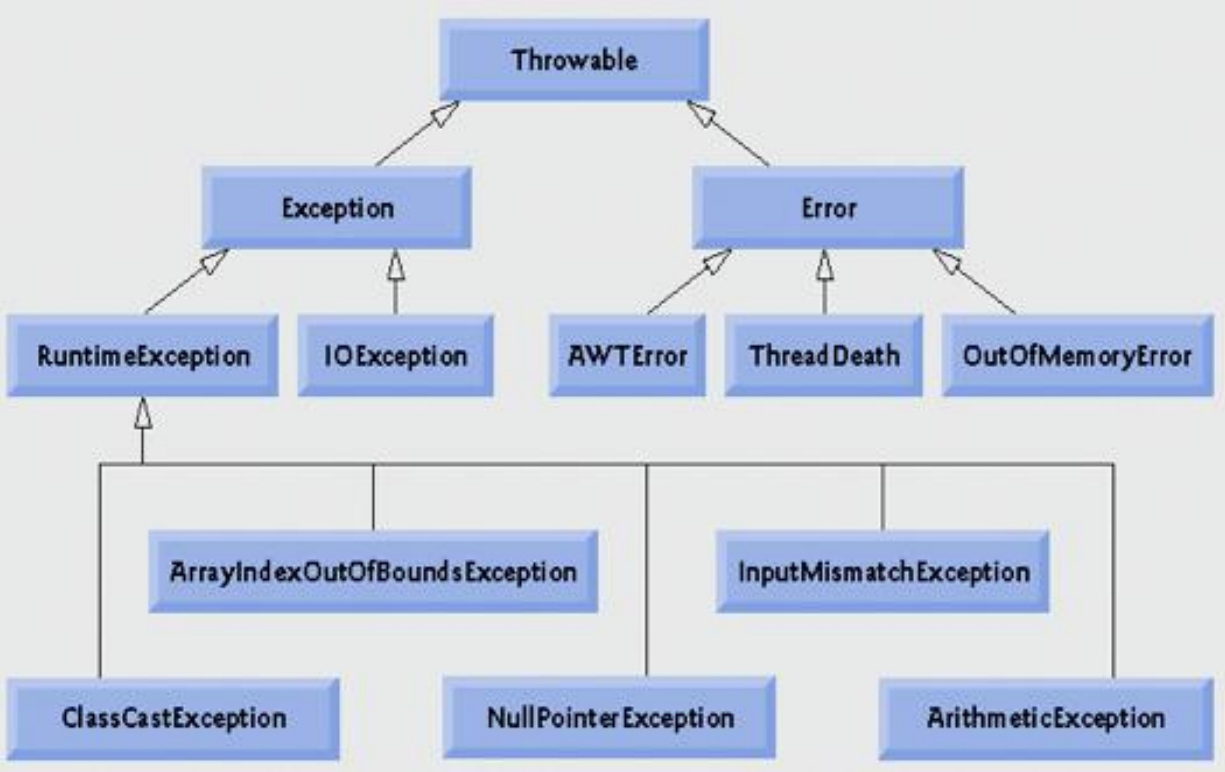
Tratamento de exceção em Java

- Uso de throws:
  - Passa a exceção para o método chamador:

```
1  package br.com.alissonrs.aula7;
2
3  public class Exemplo3_1_Excecao {
4
5      public static void main(String[] args) throws Exception {
6          ContaArray();
7      }
8      static void ContaArray() throws Exception{
9          int[] array = new int[10];
10         for (int i = 0; i <= 10; i++) {
11             array[i] = i;
12             System.out.println(i);
13         }
14         System.out.println("Fim!");
15     }
16 }
17
```

Tratamento de exceção em Java

– Hierarquia de exceções Java (mais completo)



Tratamento de exceção em Java

– Mais um exemplo:

```java
package br.com.alissonrs.aula7;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Exemplo4_Excecao {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Digite um número: ");
            try {
                System.out.println(1000/entrada.nextInt());
            } catch (InputMismatchException e) {
                System.out.println("Mensagens de erro.");
                //System.out.println(e.getMessage());
                //e.printStackTrace();
            }finally{
                entrada.close();
                System.out.println("Saindo do programa");
            }
        System.out.println("Fim!");
    }
}

```

Finally Exemplo

```
1  class DemoFinally {
2      static void procA() throws Exception {
3          try {
4              System.out.println("dentro de procA");
5              throw new Exception("demo");
6          } finally {
7              System.out.println("finally de procA");
8          }
9      }
10
11     static void procB() {
12         try {
13             System.out.println("dentro de procB");
14             return;
15         } finally {
16             System.out.println("finally de procB");
17         }
18     }
19
20     public static void main(String args[]) {
21         try {
22             procA();
23         } catch (Exception e) {
24             System.out.println("Tratando excecao que"
25                     + " ocorreu no método procA");
26         }
27         procB();
28     }
29 } // da class
```

```
        }
    } // da class
```

Problems | Javadoc | Declaration | Console

<terminated> DemoFinally [Java Application] C:\Program Files\Java\jre1.

```
dentro de procA
finally de procA
Tratando excecao que ocorreu no método procA
dentro de procB
finally de procB
```

Criando suas próprias exceções

- Classes que estendem Exception

  (extends Exception)

```
class ArgumentoException extends Exception{
```

- Funções que informam a possibilidade de lançamento de uma exceção.

  (throws)

```
int multiplicar(int x, int y) throws ArgumentoException {
```

- Lançamento da exceção caso o erro ocorra.

  (throw)

```
    throw new ArgumentoException();
```

– Um Exemplo Completo

Tratamento de exceção em Java

Criando a minha classe de exceções

```
1  package minhaexcecao;
2
3  public class MinhaExcecao extends Exception {
4      private int detalhe;
5
6      public MinhaExcecao(int a) {
7          detalhe = a;
8      }
9
10     public String toString() {
11         return "MinhaExcecao [" + detalhe + "]";
12     }
13 } // da class MinhaExcecao
```

Usando throw para lançar exceções

```
1  package minhaexcecao;
2
3  public class DemoExcecao {
4      public static void main(String args[]) {
5          try {
6              int a = 11;
7              if (a > 10) {
8               MinhaExcecao minhaExc = new MinhaExcecao(a);
9               throw minhaExc;
10             }
11         } catch (MinhaExcecao e) {
12             System.out.println("Excecao capturada: " + e);
13         }
14     }
15 } // da class DemoExcecao
```

throws

```
1  package minhaexcecao;
2
3  public class DemoThrows {
4      public static void proced() throws MinhaExcecao {
5          System.out.println("No Procedimento.");
6          throw new MinhaExcecao(1);
7      }
8
9      public static void main(String args[]) {
10         try {
11             proced();
12         } catch (MinhaExcecao e) {
13             System.out.println("Aconteceu exceção do"
14         + " tipo MinhaExcecao.");
15         }
16     }
17 }
```

– Outro Exemplo Completo

Tratamento de exceção em Java

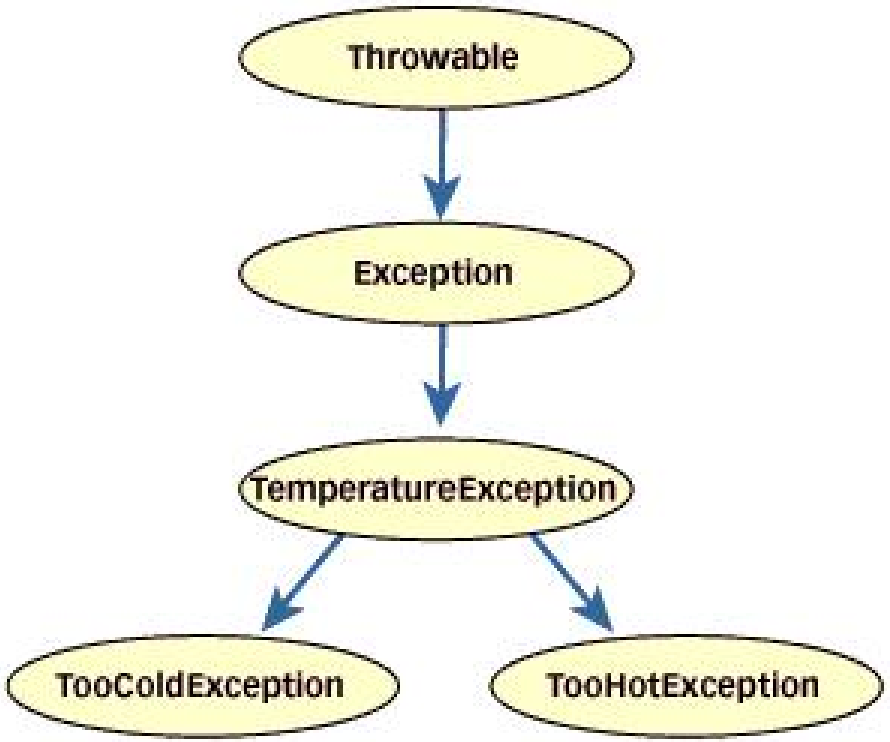
Tra

```java
class ArgumentoMuitoGrandeException extends Exception {

    ArgumentoMuitoGrandeException() {
    }

    ArgumentoMuitoGrandeException(String erro) {
        super(erro);
    }
}

public class MultiplicacaoComTratamento {

    public static void main(String[] args) throws ArgumentoMuitoGrandeException {
        int x, y, res = 0;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);

        // Somente se x e y < 100
        // Colocar if s antes?

        res = multiplicar(x, y);

        System.out.println("Resposta = " + res);

    }

    public static int multiplicar(int x, int y) throws ArgumentoMuitoGrandeException {
        if (x > 100 | y > 100) {
            throw new ArgumentoMuitoGrandeException("X ou Y muito grandes");
        }

        int resultado = x * y;
        return resultado;
    }

}
```

– Outro Exemplo Completo

Tratamento de exceção em Java

Criando a minha classe de exceções



Criando a minha classe de exceções

```
class TemperatureException extends Exception {
      public String toString() {
         return "Tem algum problema com a temperatura!";
      }
}

class TooColdException extends TemperatureException {
      public String toString() {
         return "A temperatura está gelada demais!";
      }
}

class TooHotException extends TemperatureException {
      public String toString() {
         return "A temperatura está quente demais!";
      }
}
```

Gerando exceções

```
class VirtualPerson {
   private static final int tooCold = 65;
   private static final int tooHot = 85;

   public void drinkCoffee(CoffeeCup cup) throws TooColdExceptio,TooHotException {
        int temperature = cup.getTemperature();
        if (temperature <= tooCold) {
                throw new TooColdException();
        } else if (temperature >= tooHot) {
                throw new TooHotException();
        }
        // ...
   }
   // ...
}
class CoffeeCup {
   private int temperature = 75;
   public void setTemperature(int val) {
        temperature = val;
   }
   public int getTemperature() {
        return temperature;
   }  // ...
}
```

Testando a exceção

```
class Excecao2 {
    public static void main(String[] args) {
        int temperature = 0;
        if (args.length > 0) {
            try {
                temperature = Integer.parseInt(args[0]);
            }
            catch(NumberFormatException e) {
                System.out.println(
                    "Tem que passar um inteiro como argumento.");
                return;
            }
        }
        else {
            System.out.println(
                "Tem que passar uma temperatura.");
            return;
        }
// continua ...
```

Testando a exceção

```
        // Criando copo de café
         CoffeeCup cup = new CoffeeCup();
         cup.setTemperature(temperature);
         // cria um cliente
         VirtualPerson cust = new VirtualPerson();
         try {
                 // cliente bebe café
             cust.drinkCoffee(cup);
             System.out.println("Coffee is just right.");
         }
         catch (TooColdException e) {
             System.out.println("Coffee is too cold.");
             //lidar com cliente muito bravo! :-)
         }
         catch (TooHotException e) {
             System.out.println("Coffee is too hot.");
             //lidar com cliente muito bravo! :-)
         }
    }
}
```

Tratamento de exceção em Java

- Material online:
  - https://docs.oracle.com/javase/tutorial/essential/exceptions/
  - http://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601
  - http://pt.slideshare.net/regispires/java-13-excecoes-presentation

www.alissonrs.com.br

Alisson RS