Programação para Web I
Aula 3

Profa. Silvia Bertagnolli

Tratamento de Exceções

Exceção

Exceção significa “condição excepcional” e é uma ocorrência que altera o fluxo normal do programa

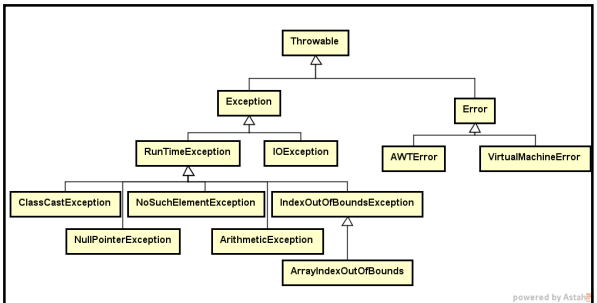
Exceções: problemas mais comuns

Os problemas mais comuns são:

- falha na aquisição de um recurso (new, open..)
- tentativa de fazer algo impossivel (divisão por zero, indice inválido..)
- outras condições inválidas (manipular objetos nulos, lista vazia, overflow..)

Algumas exceções:

- NullPointerException
- ArrayIndexOutOfBounds
- FileNotFoundException



Tipos de Exceções: Error

Error = classe base que descreve os erros não esperados ou erros que não devem ser tratados em circunstâncias normais

Indica um problema grave ocorrido em tempo de execução, tal como VirtualMachineError, OutOfMemoryError

Erros são de difícil recuperação, se não for impossível

Este tipo de problema é raro de acontecer e não pode ser tratado pelo programador

Tipos de Exceções: Exception

Exception = classe base que descreve uma condição anormal que deve ser tratada pelo programa

RuntimeException descreve as exceções que podem ocorrem em tempo de execução

Exemplos: ArithmeticException, IndexOutOfBoundsException, entre outras

Instruções para Tratar/Manipular Exceções

try – identifica um bloco de comandos que pode disparar uma exceção

catch – captura as exceções e implementa os tratadores de exceções

finally – usado para códigos de liberação de recursos

throw – usado para causar uma exceção

throws – usado para propagar uma exceção causada em um método

try/catch

try

Bloco try indica a região do programa que deve ser monitorada pelo sistema de tratamento de exceções

Dicas:

Incluir dentro do bloco try o código que pode gerar uma exceção

O bloco try é imediatamente seguido por zero ou mais blocos catch

Capturar exceções: catch

Um comando catch tem as seguintes funções:

- capturar um (determinado) tipo de exceção

- implementar um tratador para aquele tipo de exceção

SEMPRE colocar uma mensagem ou associar alguma informação no bloco catch, porque uma exceção pode ocorrer e você não perceber

Exemplo

Execução das Exceções

```
1.public class Exemplo{
2.    public static void main(String args[]) {
3.          int vetor[] = {1, 2, 3, 4};
4.          System.out.println(vetor[4]);
5.    }
6.}
```

A posição 4 existe ? Não.
Então uma exceção será gerada:
ArrayIndexOutOfBoundsException.

Como tratar a(s) exceção(ões) no código abaixo?

```
1.public class Exemplo{
2.    public static void main(String args[]) {
3.          int vetor[] = {1, 2, 3, 4};
4.          System.out.println(vetor[4]);
5.    }
6.}
```

Execução das Exceções

```
1.public class Exemplo{
2.    public static void main(String args[]) {
3.        int vetor[] = {1, 2, 3, 4};
4.        System.out.println("INÍCIO");
5.        try{
6.             System.out.println(vetor[4]);
7.             System.out.println("APÓS VETOR");
8.        }catch(ArrayIndexOutOfBoundsException e){
9.             System.out.println("Ocorreu uma exceção");
10.       }
11.       System.out.println("FIM");
12.   }
13.}
```

Saída:
INÍCIO
Ocorreu uma exceção
FIM

Execução das Exceções

```
1.public class Exemplo{
2.    public static void main(String args[]) {
3.          int vetor[] = {1, 2, 3, 4};
4.          System.out.println("INÍCIO");
5.          try{
6.               System.out.println(vetor[4]);
7.               System.out.println("APÓS VETOR");
8.          }catch(ArrayIndexOutOfBoundsException e){
9.                System.out.println("Ocorreu uma exceção");
10.         }
11.       System.out.println("FIM");
12.   }
13.}
```

Linhas que foram executadas estão marcadas em vermelho

EXERCÍCIOS

Exercícios

Abra o arquivo ListaExercicios_Aula3 que está disponível no Moodle

Agora faça os exercícios de 1 a 8

FINALLY

FINALLY

Após o último bloco catch pode ser definido um bloco finally (opcional)

Fornece o código, que é sempre executado independente de uma exceção ocorrer

O bloco finally é ideal para códigos de liberação de recursos

Se não houver blocos catch seguindo o bloco try, o bloco finally é requerido

FINALLY: SINTAXE

```
try{
   // código que pode gerar exceção
}catch(Exceção e1){
         //tratamento exceção 1
}catch(Exceção e2){
        //tratamento exceção 2
}finally{
   // sempre executa
}
```

EXECUÇÃO DAS EXCEÇÕES

```
//…
4.          System.out.println(“INÍCIO”);
5.          try{
6.               System.out.println(vetor[3]);
7.               System.out.println(“APÓS VETOR”);
8.          }catch(ArrayIndexOutOfBoundsException e){
9.                System.out.println(“Ocorreu uma exceção”);
10.         }finally{
11.               System.out.println(“DENTRO DO FINALLY”);
12.         }
13.         System.out.println(“FIM”);
//…
```

```
Saída:
INÍCIO
APÓS VETOR
DENTRO DO FINALLY
FIM
```

EXECUÇÃO DAS EXCEÇÕES

```
//…
4.          System.out.println(“INÍCIO”);
5.          try{
6.               System.out.println(vetor[4]);
7.               System.out.println(“APÓS VETOR”);
8.          }catch(ArrayIndexOutOfBoundsException e){
9.                System.out.println(“Ocorreu uma exceção”);
10.         }finally{
11.               System.out.println(“DENTRO DO FINALLY”);
12.         }
13.         System.out.println(“FIM”);
//…
```

```
Saída:
INÍCIO
Ocorreu uma exceção
DENTRO DO FINALLY
FIM
```

EXECUÇÃO DAS EXCEÇÕES

```
//…
4.          System.out.println(“INÍCIO”);
5.          try{
6.               System.out.println(vetor[4]);
7.               System.out.println(“APÓS VETOR”);
8.          }catch(ArrayIndexOutOfBoundsException e){
9.                System.out.println(“Ocorreu uma exceção”);
10.         }finally{
11.               System.out.println(“DENTRO DO FINALLY”);
12.         }
13.         System.out.println(“FIM”);
//…
```

Linhas que foram executadas estão marcadas em vermelho

Execução das Exceções

```
//…
4.          System.out.println(“INÍCIO”);
5.          try{
6.               System.out.println(vetor[3]);
7.               System.out.println(“APÓS VETOR”);
8.          }catch(ArrayIndexOutOfBoundsException e){
9.                System.out.println(“Ocorreu uma exceção”);
10.         }finally{
11.               System.out.println(“DENTRO DO FINALLY”);
12.         }
13.         System.out.println(“FIM”);
//…
```

Linhas que foram executadas estão marcadas em vermelho

Execução das Exceções: passos

O controle do programa deixa o bloco **try**

Os blocos **catch** são pesquisados em ordem a procura do tratador apropriado

Se um tipo de exceção disparada corresponder ao tipo de parâmetros em um dos blocos **catch**, o código desse bloco é executado

Se nenhuma exceção for disparada pelo bloco **try** os tratadores **catch** são ignorados

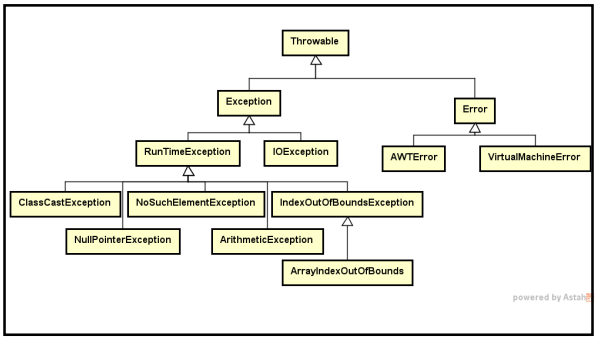
Se um bloco **finally** aparece após o último **catch**, ele é executado independentemente de uma exceção ter sido disparada

EXERCÍCIOS

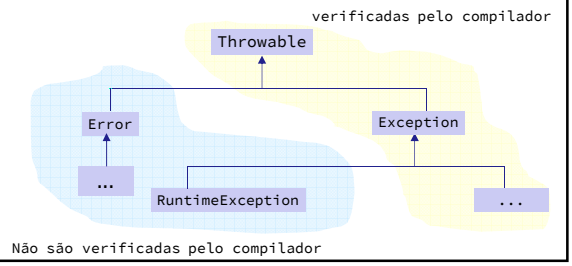
Exercícios

Usando o arquivo ListaExercicios_Aula3 que está disponível no Moodle Faça os exercícios 11 e 12

Hierarquia de Classes



Hierarquia de Classes



Tipos de Exceções: Exception

Exception = classe base que descreve uma condição anormal que deve ser tratada pelo programa

Durante a compilação, as exceções são divididas em dois grupos (i) exceções verificadas, e (ii) exceções não verificadas

Exceções verificadas pelo compilador devem ser tratadas, pois, caso contrário, a classe não é compilada

try/catch: sintaxe x hierarquia de classes

```
try{
   // código que pode gerar exceção
}
catch(Exceção e1){                    → subclasse
        //tratamento exceção 1
}
catch(Exceção e2){                    → superclasse
       //tratamento exceção 2
}
```

Capturando Qualquer Exceção

Para capturar qualquer exceção basta fazer um catch que capture a classe Exception

Vantagem: mais rápido

Desvantagem: construção muito vaga, porque não se sabe ao certo qual foi a exceção gerada

```
catch (Exception e) {
    //...
}
```

Métodos Classe Exception

String getMessage()

Retorna mensagem passada pelo construtor

String toString()

Retorna nome da exceção e mensagem

void printStackTrace()

imprime detalhes sobre exceção

EXERCÍCIOS

Exercícios

Usando o arquivo ListaExercicios_Aula3 que está disponível no Moodle Faça os exercícios 9 e 10

Como tratar as exceções verificadas pelo compilador?

Quais são as exceções que devem ser tratadas ?

Passo 1: localizar na documentação do Java a classe que deseja utilizar

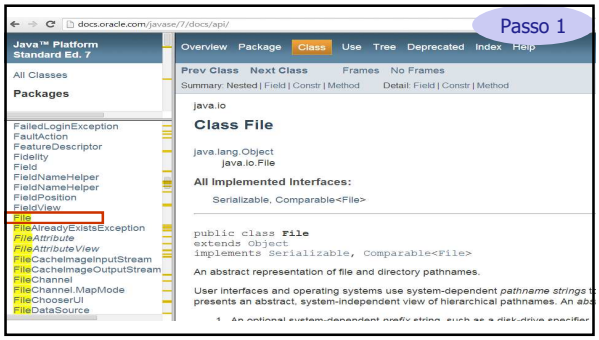
Passo 2: localizar o método que deseja utilizar no programa

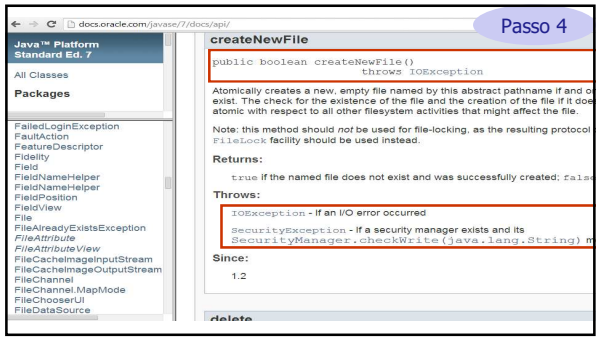
Passo 3: ativar a documentação do método

Passo 4: verificar se o método propaga algum método (instrução **throws** na assinatura do método)

Obs.:

Caso o método possua a instrução throws você deve tratar a exceção que aparece, caso contrário não precisa






```
public class Arquivos{
  public static void main(String args[]){
     String nomeArq = JOptionPane.inputDialog("Informe o
                                              nome do arquivo:");
     String op = JOptionPane.inputDialog(
                 "1 - Criar arquivo
                 2 - Excluir arquivo
                 Informe uma opção:");
     int opcao = Integer.parseInt(op);
     File f= new File(nomeArq);
     switch (opcao){// converte String em int
          case 1:// cria arquivo
                 if (f.createNewFile()==true)
                     System.out.println("Arq. criado");
                 else
                     System.out.println("Arq. já existe");
                break;
```

THROWS

THROWS

Através da cláusula throws é possível listar, na declaração de um método, as exceções que podem ser disparadas por este

A sintaxe para usar essa instrução é dada por:

```
<modificadores> <tipo_retorno> <nome_método>

    (<lista_parâmetros>) throws <nome_exceção1>

                                        ,... <nome_exceçãoN>{

         // código do método

}
```

THROWS

throws declara que o método pode provocar exceções

Considerando que:

- a unidade de execução pode detectar problemas mas geralmente não sabe como tratá-lo

- a unidade requisitante não pode detectar problemas mas geralmente sabe como tratá-los

- é conveniente saber quais são as exceções que um método pode disparar para providenciar um tratador

- Uma declaração throws é obrigatória em métodos e construtores que deixam de capturar uma ou mais exceções que ocorrem em seu interior

```
public void m1() throws Excecao1, Excecao2 {...}
```

THROWS: EXEMPLO

```
public class Exemplo{
     public static void m1() throws Exception{
           System.out.println("Executou m1()!" );
           //.....
      }
      public  static void main(String args[]){
            try{
                 m1();
            }catch(Exception e){
                  System.out.println("Erro!" );
            }
      }
}
```

EXERCÍCIOS

EXERCÍCIOS

Usando o arquivo ListaExercicios_Aula3 que está disponível no Moodle faça o exercício 13

THROW

THROW

Comando throw:

- dispara explicitamente um tipo de exceção
- pré-definido ou definido pelo usuário

É possível que o tratador catch que capturou uma exceção decida que não é capaz de processá-la, neste caso o tratador pode dispará-la novamente usando a instrução throw

THROW

Sintaxe da instrução throw

```
throw new <nome classe de exceção>();
```

Para forçar a ocorrência de uma exceção, utiliza-se a palavra reservada **throw** (no singular)

DISPARAR UMA EXCEÇÃO: THROW

```
public void metodo1( ) {
  try {
        metodo2( );
  }catch(IOException e){System.out.println(e);}
}
```

throws (plural) propaga exceção

```
public void metodo2( ) throws IOException {
    if (true)
        throw new IOException( );
}
```

throw (singular) gera exceção

THROW: EXEMPLO

```
//…
4.          System.out.println(“INÍCIO”);
5.          try{
6.             throw new ArrayIndexOutOfBoundsException();
8.          }catch(ArrayIndexOutOfBoundsException e){
9.              System.out.println(“DENTRO DO CATCH”);
10.         }finally{
11.             System.out.println(“DENTRO DO FINALLY”);
12.         }
13.         System.out.println(“FIM”);
//…
```

```
Saída:
INÍCIO
DENTRO DO CATCH
DENTRO DO FINALLY
FIM
```

EXCEÇÕES: RESUMO

O tratamento de exceções não pode ser usado para substituir testes

Em um mesmo método procure agrupar todas as exceções que ocorrem em um mesmo bloco try-catch

Quando capturar uma exceção sempre exibir uma mensagem indicando o erro que ocorreu NUNCA: catch(Exception e){}

Procure não propagar as exceções, quanto mais perto da instrução que gerou a exceção, mais preciso será o tratamento do erro

TRABALHO

TRABALHO

Usando o arquivo ListaExercicios_Aula3 faça as questões 14 a 16.