Programação Orientada a Objetos1

TRATAMENTO DE EXCEÇÕES

CONCEITOS

Quando ocorre um erro em tempo de execução:

- O USUÁRIO NÃO QUER VER (e nem entende) NA TELA UM STACKTRACE

- O USUÁRIO QUER VER NA TELA APENAS UMA MENSAGEM BEM ESCRITA APONTANDO O OCORRIDO

CONCEITOS

EXCEÇÃO

- É um evento não esperado que ocorre no sistema quando está em tempo de execução (Runtime)
- Geralmente, quando o sistema captura alguma exceção, o fluxo do código fica interrompido
- Podemos capturar uma exceção precedida de um tratamento
- Tratar uma exceção é importante porque auxilia o sistema em falhas como:
  - comunicação, leitura e escrita de arquivos
  - entrada de dados inválidos
  - acesso a elementos fora de índice
  - Conexão com banco de dados
  - entre outros

Tratar uma exceção ajuda a detectar e tratar possíveis erros

MAIOR ROBUSTEZ NO CÓDIGO

CLASSIFICAÇÃO

Implícitas (Não-verificadas)

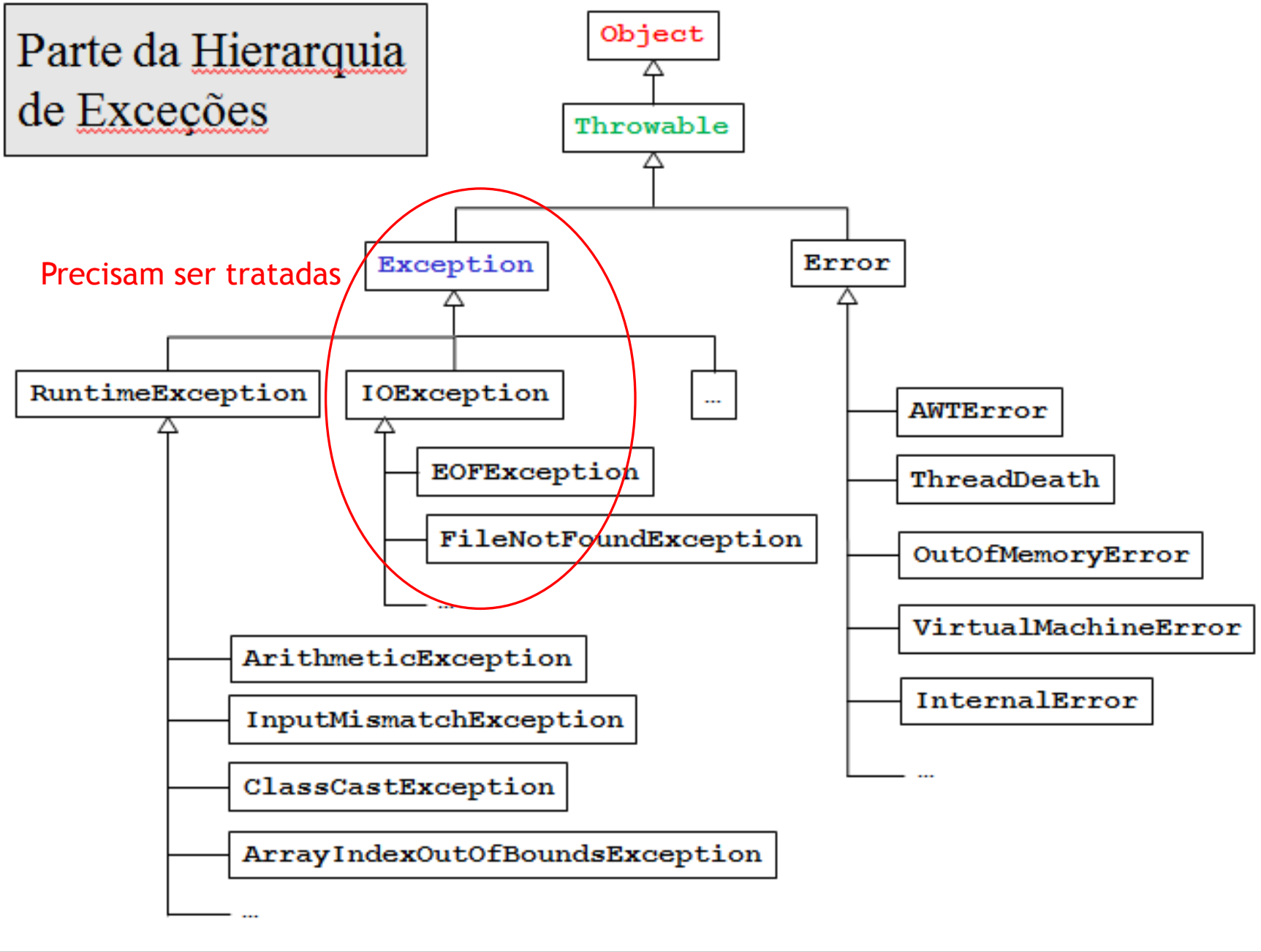
- Exceções que não precisam de tratamento e demonstram ser contornáveis. Esse tipo origina-se da subclasse **Error** ou **RunTimeException**

Explícitas (Verificadas)

- Exceções que precisam ser tratadas e que apresentam condições incontornáveis. Esse tipo origina do modelo **throw** e necessita ser declarado pelos métodos. É originado da subclasse **Exception** ou **IOException**

Classe	Objetivo	Eventos
Error	Classificada como exceção que não pode ser tratada pela aplicação.	
Exception	Trata todas as exceções da aplicação que podem ser tratadas e capturadas.	I/O Exception ou aplicar uma divisão por zero resultando na exceção **ArithmeticException**
RuntimeException	Resgata as exceções lançadas pela máquina virtual (JVM).	Referência nula (**NullPointerException**)

Parte da Hierarquia de Excecões



TRY - CATCH - FINALLY

TRY

- O bloco **try** tenta processar o código que está dentro, sendo que se ocorrer uma exceção, a execução do código pula para a primeira captura do erro no próximo bloco abaixo.

- O uso do **try** serve para indicar que o código está tentando realizar algo arriscado no sistema.

- TODO bloco **try** possui após ele pelo menos 1 bloco **catch**.

TRY - CATCH - FINALLY

CATCH

- O bloco **catch** trata a exceção lançada.
- Pode haver 1 ou mais blocos **catch**, sendo que cada um tenta tratar um tipo de exceção.
- Se ocorrer uma exceção no **try** e ela não for tratada por nenhum **catch**, então, o erro ocorrerá e, dependendo do erro, o sistema irá parar.

TRY - CATCH - FINALLY

FINALLY

- O bloco **finally** sempre finaliza a sequência de comandos do sistema, independente de ocasionar algum erro no sistema.

- Este bloco é opcional.

- É usado em ações que sempre precisam ser executadas independente da ocorrência do erro.

EXEMPLO: fechamento da conexão de um banco de dados.

TRY - CATCH - FINALLY

Indica-se try-catch em métodos que manipulam dados:

- CRUD no banco de dados
- Índices fora do intervalo de array
- Cálculos matemáticos
- I/O de dados
- Erros de rede
- Anulação de objetos
- Outros

EXEMPLO DE ESTRUTURA TRY-CATCH

```
/**
 * Classe utilizada para demonstrar o bloco try / catch.
 */
public class ExemploExcecao {
  public static void main(String[] args) {
    try {
      /* Trecho de código no qual uma
       * exceção pode acontecer.
       */
    } catch (Exception ex) {
      /* Trecho de código no qual uma
       * exceção do tipo "Exception" será tratada.
       */
    }
  }
}
```

TROW - TROWS

TROW

- É possível que o próprio programador lance uma exceção por meio do uso da palavra-chave **throw**

FORMATO

```
throw new << Exceção_desejada >> ( );
```

THROWS

- Quando a exceção é do tipo verificada, sabe-se que é necessário tratá-la. Mas há a possibilidade de passar essa tarefa para o local em que a função foi chamada. Usa-se a palavra-chave **throws** na frente da declaração da função.

FORMATO

```
<modificadores> <tipo> <nome_método>( <parametros> ) throws <classes>
{ }
```

TROW - TROWS

- Quando um método lança mais de uma exceção, coloca-se todas elas na assinatura do método separadas por vírgula

- Quando houver várias exceções cujo tratamento seja o mesmo, usa-se colocar um único **catch** contendo como argumentos as classes das exceções separadas pelo operador **|**

- É possível agregar várias exceções generalizando todos os **catch** em um único catch (Exception). É indicado que se utilize uma mensagem genérica seguida de um *stacktrace*

- É possível criar a nossa própria exceção quando se deseja ser mais específico para tratar uma dada exceção

- Nesse caso, cria-se uma classe específica para representar essa exceção