PROGRAMAÇÃO ORIENTADA A OBJETOS

Tratamento de Exceções

Prof. Emanoel Barreiros

Exceções

- Como o próprio nome diz, exceções são coisas que sinalizam um cenário fora do normal, exceções à regra
- O tratamento dessas exceções permite que seu sistema resolva exceções
- Em muitos casos (quando o tratamento é bem feito) permite que seu sistema continue executando como se nada tivesse acontecido

Exemplo

- O que acontece quanto tentamos dividir um número por zero sem nos preocupar com tratamento de exceções?
- Vamos ver rodando…

O bloco try… catch

- No código vimos alguns blocos novos
  - try: engloba um trecho de código que “pode” lançar uma exceção, ou seja, nem sempre ela será lançada. Caso ocorra uma exceção, o fluxo de controle passa para o trecho catch correspondente à exceção que foi lançada
  - catch: engloba um trecho de código que será executado caso sua exceção correspondente for lançada. Se tudo der certo, esse trecho de código não deverá ser executado.

Capturando exceções

- O bloco try no exemplo que vimos é seguido de dois blocos catch
  - InputMismatchException: sinaliza um erro de conversão devido a entrada em formato errado
  - ArithmeticException: sinaliza um erro de aritmética
- Cada um desses blocos é responsável por recuperar o programa de um erro específico
- O bloco catch é caracterizado pela palavra-chave catch, seguida do tipo da exceção que ele trata entre parênteses e uma variável para armazenar a exceção

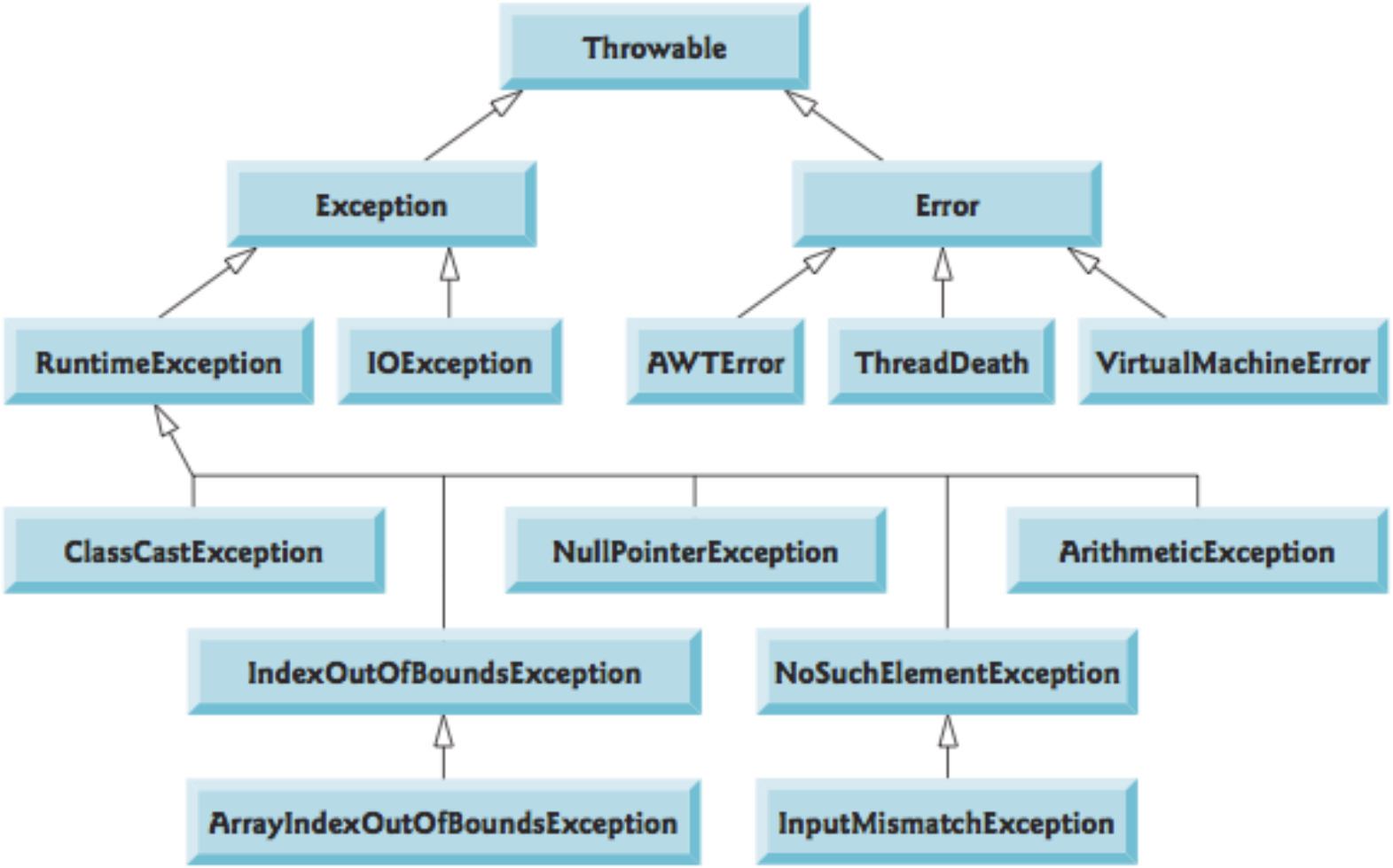
Capturando exceções

- Um bloco try deve ser seguindo por pelo menos um bloco catch (podem ter mais) ou um bloco finally (daqui a pouco falamos mais sobre isso)
- Caso ocorra uma exceção, o fluxo de controle passa para o trecho catch correspondente à exceção que foi lançada
- Depois que a exceção é tratada, o fluxo não volta para o try, ele tenta executar a primeira linha do finally (caso exista) ou continua da próxima linha após os catch
- Caso mais de um bloco catch corresponda a uma exceção sendo lançada, apenas o primeiro encontrado executa

A claúsula *throws*

- A cláusula throws é usada para sinalizar que aquele método “pode” lançar aquele tipo de exceção
- Obriga quem usa aquele método a tratar a exceção ou relançá-la
- Eventualmente, alguém precisa tratar a exceção, caso contrário, o programa será encerrado

Hierarquia de exceções em Java



Exceções checadas vs. exceções não checadas

- Java distingue entre exceções checadas e não checadas
- O tipo da exceção vai determinar se a exceção é checada ou não
  - Exceções que são subclasses de RuntimeException são exceções não checadas
  - Quaisquer outras exceções são checadas
- O programador É obrigado a tratar exceções ou relançá-las caso elas sejam CHECADAS
- O programador NÃO é obrigado a tratar exceções NÃO CHECADAS

O Bloco *finally*

- Usado para conter código a ser executado haja exceção ou não
- Muito útil para liberar recursos (fechar conexões com banco, fechar arquivos, etc.)
- É opcional, mas se existir, deve aparecer depois do último catch

Lançando exceções

- Você pode lançar exceções quando bem entender
- O lançamento de exceções é feito usando o comando throw da seguinte forma:
  - throw new Exception(“exemplo de excecao”);

Criando novas exceções

- Para criar novas exceções você deve criar uma nova classe e estender a classe de exceção correta:
  - Se quiser que sua exceção seja checada, você deve estender da classe Exception
  - Se quiser que sua exceção seja checada, você deve estender da classe RuntimeException

Exercícios

- Escreva um programa que mostre como várias exceções podem ser capturadas usando catch(Excepction e)
- Use herança para criar classes de exceção chamadas ExcecaoA, ExcecaoB e ExcecaoC, onde ExcecaoB estende ExcecaoA e ExcecaoC estende ExcecaoB. Escreva um programa que mostre que um catch para ExcecaoA consegue capturar também tanto excecoes do tipo ExcecaoB quanto Excecao C.