Usando as palavras-chave Throws e Throw

Como uso as palavras `Throws` e `Throw` num código Java?

java exceção

editada 27/05/14 às 13:32 — bigown ♦ — 197mil 34 443 698

perguntada 25/05/14 às 17:11 — Beto Barroso — 81 1 1 3

1 Dê uma olhada no tour. Você pode aceitar uma resposta se ela resolveu seu problema. Você pode votar em todos os posts do site também. Alguma lhe ajudou mais? Precisa que algo seja melhorado? – bigown ♦ 16/07/15 às 0:52

4 Respostas

Cheked Exception

`throws` faz parte da **declaração do método**, da sua assinatura. Ele define parte da API. Indica que um trecho de código que chame este método deve obrigatoriamente capturar uma possível exceção que ele lance. Mesmo que não deseje fazer nada com a exceção, ela deve ser capturada e relançada. Ou deve indicar que o método que usa outro com uma exceção possível tem um `throw`, como abaixo:

```
public void M1() throws IOException {
    FileReader f = new FileReader("notExist.txt");
}
```

Neste caso, qualquer método que chame `M1` deve obrigatoriamente tratar a exceção `IOException` com um `try` ou indicar que ele lança a mesma exceção especificada em `M1`.

Pode parecer que não há nenhuma exceção neste método, mas o `FileReader` lança uma exceção `FileNotFoundException` que é derivada do `IOException`. A assinatura do método construtor `FileReader` exige um tratamento dela. A inclusão do `throws IOException()` na assinatura do método garante o tratamento delegando para o chamador de `M1` tratar adequadamente.

Uma das formas de tratar (exemplo tosco):

```
try {
    M1();
} cath (IOException e) {
    System.out.println("Deu erro no arquivo");
}
```

Um outra froma do método que não compila:

```
public void M1() {
    FileReader f = new FileReader("notExist.txt");
}
```

Nenhum tratamento foi dado, não capturou a exceção e não indicou ao compilador que deve ser responsabilidade do chamador de `M1`.

Ele pode ser considerado uma **diretiva para compilador** verificar se o devido tratamento está sendo dado. É tratado em tempo de compilação.

Há muita controvérsia se isto é bom ou ruim para uma aplicação. Frequentemente se usa mais `throws` do que devia e é extremamente comum programadores "fingirem" que estão tratando, já que é obrigatório um tratamento real (não tem como um compilador forçar isto). Em muitos casos isso é algo ruim para a aplicação, mas há casos que realmente não há nada correto e viável que possa ser feito, mas a API exige que a exceção seja tratada.

É recomendado que alguns tipos de exceções sejam tratadas obrigatoriamente. Outras não. Erros de programação ou qualquer erro irrecuperável não precisam e não devem ser tratados.

Exceções em que o programa pode realmente tomar alguma ação e resolver o problema (muito comum em acesso a recursos externos) são passíveis de fazer a chamada *checked exception* com a declaração de `throws`. Uma `IOException` normalmente deve ser tratada, é um **erro recuperável**. Outro exemplo é um `SQLException`. Já um `NullPointerException` ou as exceções derivadas de `RunTimeExcepetion` seria absurdo exigir um tratamento, já que é erro de programação e não há nada seguro que possa ser feito para contornar o problema.

Lançar uma exceção

`throw` é um *statement*, ele **manda a exceção ser lançada**.

```
public void M2() {
    throw new IOException();
}
```

Este método lança uma exceção mas não exige que ela seja tratada por seus chamadores. Ele **transfere o controle do fluxo** para os métodos chamadores. Ele usa o que se chama *unckecked exception*, ou seja, uma exceção é lançada mas nada obriga ela ser tratada. É tratado em tempo de execução.

Exemplos de código completo aqui, aqui, aqui e aqui.

Mais exemplos:

```
import java.io.*;

public class Estudos{
  public static void main(String[] args){
    try{
      DataInputStream in = new DataInputStream(
        new BufferedInputStream(
          new FileInputStream("conteudo.txt")));

      while(in.available() != 0)
        System.out.print((char) in.readByte());
    }
    catch(IOException e){
      System.out.print(e.getMessage());
    }

    System.exit(0);
  }
}
```

Note que `DataInputStream`, `BufferedInputStream` não obrigam tratar exceções, mas `FileInputStream` precisa ter `FileNotFoundException` tratada obrigatoriamente, conforme documentação. O tratamento foi feito de forma generalizada para qualquer `IOException`.

```
import java.util.*;

class Estudos {

    public static void main(String[] args) {
        String palavra = "Java";
        Scanner in = new Scanner(System.in);
        System.out.print("Informe um inteiro: ");
        int indice = in.nextInt();
        try {
            System.out.println("O caractere no índice informado é " +
palavra.charAt(indice));
        }
        catch(StringIndexOutOfBoundsException e) {
            System.out.println("Erro:" + e.getMessage());
        }
    }
}
```

Veja funcionando no ideone. Também coloquei no GitHub para referência futura.

Neste exemplo se tratou uma exceção `StringIndexOutOfBoundsException` que poderia ser gerado em `charAt()`, mas nada obriga que isto fosse feito.

Conclusão

Um faz parte da assinatura do método o outro executa o lançamento de uma exceção desviando o fluxo de execução do algoritmo.

Com estes exemplos e os outros das outras respostas, acho que já te ajuda bastante. Informe na pergunta se precisa de mais informações específicas. Vá se acostumando ler documentação. Vai fazer isto a vida toda :)

editada 10/03 às 13:01

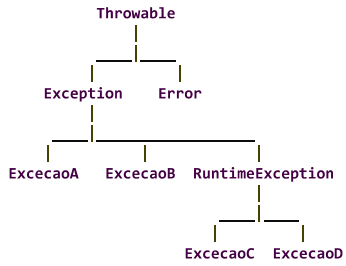
respondida 25/05/14 às 17:22

bigown ♦
197mil 34 443 698

A palavra-chave `throw` serve para lançar uma exceção (mais precisamente um `Throwable`, mas em 99,9% dos casos uma exceção). A palavra-chave `throws` serve para declarar que um método pode lançar exceções de um determinado tipo.

Para entender melhor o que significa "poder lançar exceções", vejamos.

Esta é a hierarquia das classes `Throwable` (as classes `ExcecaoA`, `ExcecaoB`, `ExcecaoC` e `ExcecaoD` são inventadas e foram colocadas apenas como exemplos):



Vamos focar nas subclasses de `Exception` e `RuntimeException`, que são os casos mais comuns.

Em Java há dois tipos de exceções:

1. *Checked exceptions:* as que precisam ser tratadas em tempo de compilação (são subclasses de `Exception` mas não de `RuntimeException`), tendo como exemplos em nossa hierarquia as classes ExcecaoA e ExcecaoB;
2. *Unchecked exceptions:* as que não precisam ser tratadas em tempo de compilação (são subclasses de `RuntimeException`), tendo como exemplos em nossa hierarquia as classes ExcecaoC e ExcecaoD.

Quando digo "tratadas em tempo de compilação", quero dizer que se uma determinada exceção é lançada dentro de um método por meio da palavra-chave `throw`, esse mesmo método deve **capturar** a exceção ou então **declarar que a lança por meio da cláusula** `throws` (o que significa efetivamente postergar a captura da exceção para outros métodos que chamem esse método). No caso de subclasses de `RuntimeException`, esse tratamento não é necessário, e a exceção em questão pode ser lançada por qualquer parte do código em qualquer situação.

Quando o código em execução chega a uma linha contendo a palavra-chave `throw`, note que essa linha está dentro de um método que está dentro de outro método e assim por diante, isto é, durante a execução do código existe uma pilha de execução de métodos. No momento em que o `throw` é executado e a exceção é lançada, ela é propagada ao longo da pilha de execução de métodos até chegar ao fim da pilha ou então ser capturada por um bloco `catch`, que permite tratar essa exceção, relançá-la, ou lançar uma nova exceção de um tipo diferente de volta para a pilha de execução de métodos.

Para mais detalhes, consulte a lição Exceptions no Oracle Java Tutorial (em inglês).

editada 25/05/14 às 17:59 respondida 25/05/14 às 17:18


Piovezan
7.185 1 12 41

1 Valeu @Piovezan mas seria possível mandar um código pronto com as palavras, é que sou iniciante em java. Desde já, agradeço. – Beto Barroso 25/05/14 às 18:06

Suponha que tu quer criar um método cuja entrada não pode ser menor que zero.

Use `throw` para lançar uma exceção nesse caso:

```
if (numero < 0) {
  throw new Exception("Número não pode ser menor que zero!");
}
```

A declaração `throws` é usado em um método para indicar que ele lança uma determinada Exceção:

```
public void fazAlgo(int numero) throws Exception {
  if (numero < 0) {
    throw new Exception("Número não pode ser menor que zero!");
  }
  // resto do método
}
```

É claro que o recomendado é criar suas próprias Exceções:

```
class NumeroMenorQueZeroException extends Exception {
  // sua classe
}
```

Assim a declaração ficaria desse jeito:

```
class MinhaClasse {

  // o "throws" no método abaixo indica que ele lança
  // a exceção "NumeroMenorQueZeroException", que criamos acima

  public void fazAlgo(int numero) throws NumeroMenorQueZeroException {
    if (numero < 0) {
```

```
        throw NumeroMenorQueZeroException("Numero não pode ser menor que zero!");
      } else {
        // faz algo com o número maior ou igual a zero
      }
    }

}
```

O `try/catch` ficaria assim:

```
try {
  minhaClasse = new MinhaClasse();
  minhaClasse.fazAlgo(-1);

} catch (NumeroMenorQueZeroException e) {
  JOptionPane.showMessageDialog(null, "Um erro aconteceu: " + e);
}
```

Você pode ler mais sobre Exceptions aqui: *http://www.caelum.com.br/apostila-java-orientacao-objetos/excecoes-e-controle-de-erros/*

editada 25/05/14 às 18:19 respondida 25/05/14 às 17:19

user7261

1 Obrigado pela explicação mas seria possível dentro de um código pronto. É que sou iniciante em java. @Andrey – Beto Barroso 25/05/14 às 18:05

1 @BetoBarroso Aprofundei um pouco minha resposta. Se não é bem isso que tu quer, edite sua pergunta sendo mais específico. =) – user7261 25/05/14 às 18:20

- **Throws** Você está lançando uma exceção, exemplo

```
public void acordar() throws Exception {
    throw new Exception("opa, deu erro");
}
```

Ou seja, você está "dizendo" para quem chamar este método que ele **PODE** (não quer dizer que vai, como no exemplo) explodir uma exceção.

- **Throw** Você está "tentando" uma exceção, como no exemplo 1

```
throw new Exception("opa, deu erro");
```

Como você pode ver ele está estourando uma exceção

editada 26/12/15 às 18:38 respondida 23/05/15 às 0:15

deFreitas

623 3 9
