UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Excessões

Carlos Arruda Baltazar
UNIP – Cidade Universitária

Exceções e Erros

Uma exceção (exception) é um objeto especial, de uma classe específica pertencente a hierarquia da API Java, a classe Exception do pacote java.lang, que é destinada a sinalização de situações que podem ser tratadas pelo programador.

Exceções e Erros

Isto significa que uma exceção indica a ocorrência de uma anormalidade no código, a qual poderia ser solucionada pelo próprio programa, por meio do uso de um dado alternativo, de uma outra estratégia de cálculo, pela obtenção de um novo dado junto ao usuário do programa, ou eventualmente outra ação considerada apropriada.

Exceções e Erros

Um erro (error) é também um objeto especial, de outra classe específica pertencente a hierarquia Java que é java.lang.Error, mas reservada para indicar situações severas ou complexas, que não deveriam ser tratadas pelo programador.

Exceções e Erros

Assim, um erro ocorrido durante a execução do código muito provavelmente não pode ser solucionado pelo próprio programa, pois pode ter origem num defeito do hardware, num problema junto ao sistema operacional ou ainda na própria máquina virtual Java (JVM).

Exceções e Erros

Enquanto uma exceção pode estar relacionada a um problema previsível, que pode ser resolvido com algum cuidado extra no projeto do sistema; um erro é uma situação rara, imprevista, severa e crítica, cuja prudência recomenda apenas informar o ocorrido e parar imediatamente o programa.

Exceções e Erros

As exceções (exceptions) geralmente estão relacionadas a situações que, embora anormais, são previsíveis no projeto como o recebimento de argumentos inválidos, entradas de dados inconsistentes, recursos (arquivos ou urls) não encontrados ou indisponíveis, assim como o time-out das operações executadas.

Exceções e Erros

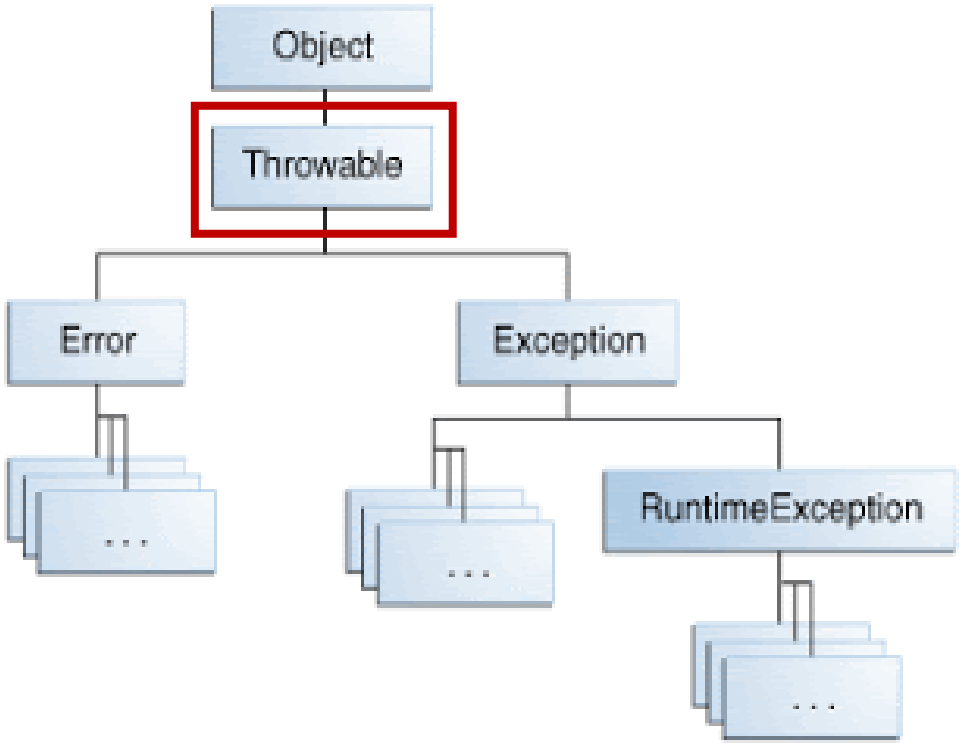
Já os erros (errors) estão sempre associados a situações anormais, atípicas e não previsíveis pelo programador, tais como erros internos da JVM, falhas oriundas do mal funcionamento do hardware (processador e memória), defeitos (bugs) no sistema operacional, assim inconsistências decorrentes de ataques ao sistema.

Exceções e Erros

Por conta disso, o termo tratamento de exceções é visto com frequência, sendo uma necessidade comum da programação. Já o tratamento de erros é raro e, por esta razão, não será tratado neste material.

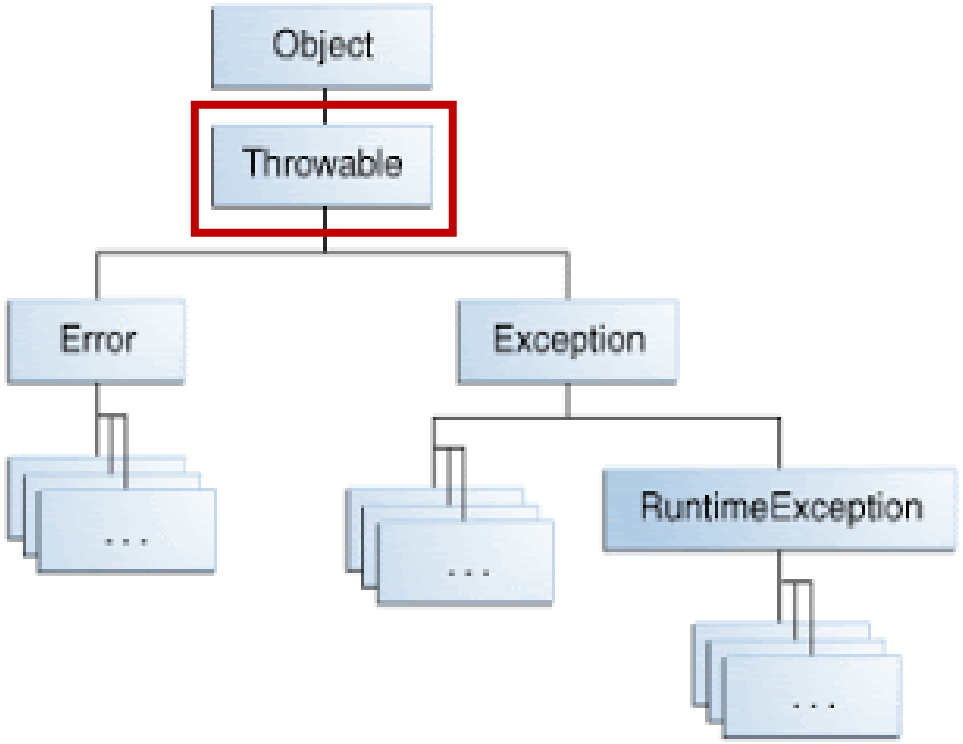
A classe Throwable

Na hierarquia de classes da API Java existe a classe Throwable, do pacote java.lang, que é a superclasse de todos os erros e exceções da linguagem Java, como mostra a figura que segue. Apenas objetos deste tipo ou de suas subclasses são lançados pela JVM e podem ser lançados pelo uso da diretiva throw.



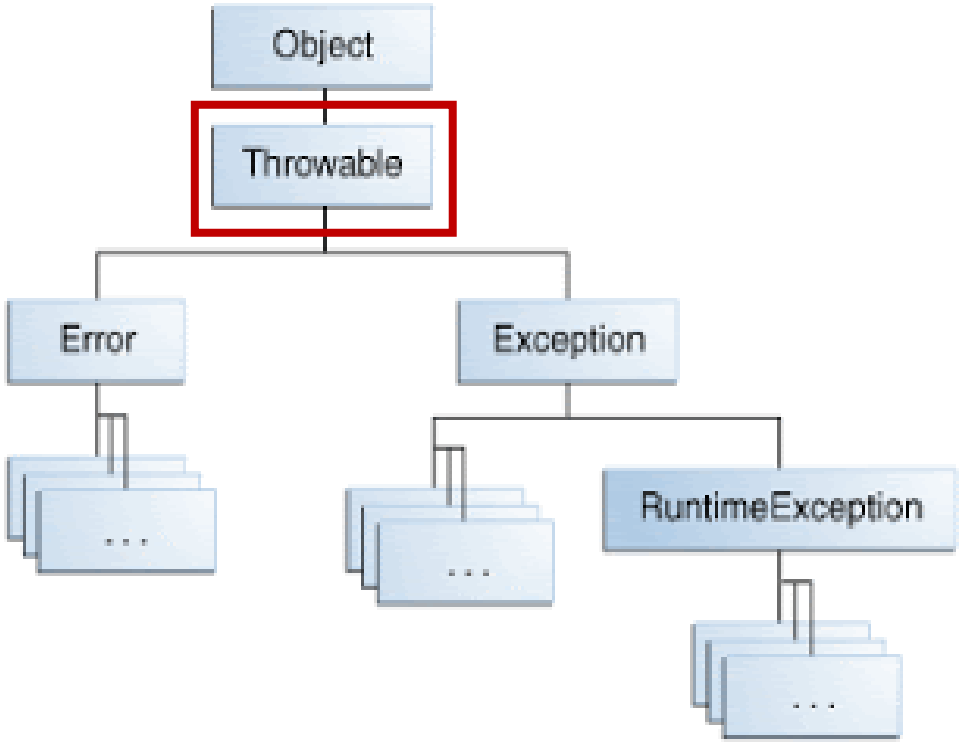
A classe Throwable

De maneira análoga, apenas objetos deste tipo, ou de suas subclasses, podem ser apanhados por cláusulas catch, que fazem parte da diretiva try.



A classe Throwable

A diretiva try e a cláusula catch, ou simplesmente try/catch, compõem uma das formas mais comuns do tratamento de exceções. Já a diretiva throw serve para que um programa sinalize a ocorrência de uma situação anormal por meio de do lançamento de uma exceção.



Classes de Exceção

Existem muitas classes de exceção no Java, mas a classe básica é java.lang.Exception, que indica um problema de natureza geral. Todas as demais classes de exceção são derivadas de Exception, por isso tem sempre o sufixo Exception em seus nomes, como nos exemplos que seguem, onde os nomes dos tipos das exceções também indicam seu propósito específico:

Classes de Exceção

- *ArrayIndexOutOfBoundsException*, sinaliza o uso de índices inválidos em *arrays*;
- *IOException*, indica problemas em operações de entrada e saída;
- *NullPointerException*, ocorre quando uma referência nula (*null*) é usada ao invés de um objeto válido;
- *NumberFormatException*, aponta problemas de representação (formato) de valores numéricos;
- *RuntimeException*, serve para indicar erros gerais durante a execução de um trecho do código do programa.

Classes de Exceção

A exceção java.lang.RuntimeException é bastante importante, pois toda a família derivada desta exceção tem tratamento opcional, ou seja, são exceções consideradas não monitoradas (unchecked exceptions).

Tipos de Exceção

Existem dois tipos de exceções: as exceções não monitoradas (unchecked exceptions) e as exceções monitoradas (checked exceptions).

As exceções não monitoradas (unchecked exceptions) são aquelas em que o tratamento com try/catch não é obrigatório. Estas exceções são implicitamente lançadas por diversos métodos presentes na API Java (o compilador não faz menção a sua ocorrência), assim como por métodos que podem ser criados pelos programadores. De alguma forma, sinalizam problemas eventuais, considerados de menor severidade.

Tipos de Exceção

Já as exceções monitoradas (checked exceptions) são aquelas em que o tratamento com try/catch é obrigatório, pois sua severidade é maior, requerendo tratamento. Estas exceções são explicitamente lançadas por muitos métodos presentes na API Java, de modo que o compilador indica como erro a ausência de tratamento próprio. Como antes, o programador pode criar métodos que lancem este tipo de exceção.

Tipos de Exceção

Independentemente de serem monitoradas ou não monitoradas, todas as exceções do Java, quando ocorrem, são lançadas para o contexto superior e, se alcançam a JVM, provocam a interrupção do programa. Apenas o tratamento é considerado opcional no caso de exceções não monitoradas (unchecked exceptions).

Blocos Try e Catch

Java fornece os blocos try e catch para nos permitir tratar exceções, de modo que nosso código possa continuar executando. Coloca-se o código que possa gerar uma exceção dentro do bloco try e o código para recuperação da exceção é colocado dentro de um bloco catch. Se uma exceção for lançada pelo código dentro do bloco try, então a execução pulará para o bloco catch, onde existe o código para tratar aquela exceção. Se nada de ilegal acontecer dentro do bloco try, o código do bloco catch será ignorado.

Blocos Try e Catch


```
public double converteESoma(String s1, String s2)
{
    try
    {
        double d1 = Double.parseDouble(s1);
        double d2 = Double.parseDouble(s2);

        return d1 + d2;
    }
    catch(NumberFormatException error)
    {
        System.out.println(error.toString());
        return 0;
    }
    catch(NullPointerException error)
    {
        System.out.println(error.toString());
        return 0;
    }
}
```

Cláusula Throws

A cláusula throws pode ser aplicada em um método para indicar explicitamente as exceções possivelmente lançadas em seu código, desobrigando seu tratamento local, quaisquer sejam seus tipos (checked ou unchecked). O mesmo método converteESoma(String, String) poderia ser modificado como:

```
public double converteESoma(String s1, String s2)
throws NumberFormatException, NullPointerException
{
        double d1 = Double.parseDouble(s1);
        double d2 = Double.parseDouble(s2);
        return d1 + d2;
}
```

Cláusula Throws

A cláusula throws, que segue a lista de argumentos do método, lista as exceções, que podem ser lançadas pelo método. Isto auxilia o programador a prever os problemas decorrentes do uso do método, além de desonerar o tratamento das exceções listadas no código do próprio método.

Cláusula Throws

No caso de exceções não monitoradas (unchecked exceptions), sua indicação com a cláusula throws não é obrigatória, mas torna explícito seu lançamento possível, auxiliando o programador.

No caso de exceções monitoradas (checked exceptions), sua indicação com a cláusula throws é obrigatória quando não são tratadas no código do método.

OBRIGADO