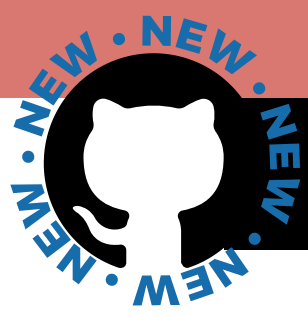


Acesse o repositório do github com exemplos.

EXCEÇÕES

EXCEPTIONS

Uma ***Exception*** é um evento que "rompe" ou "quebra" o fluxo normal de execução do programa.

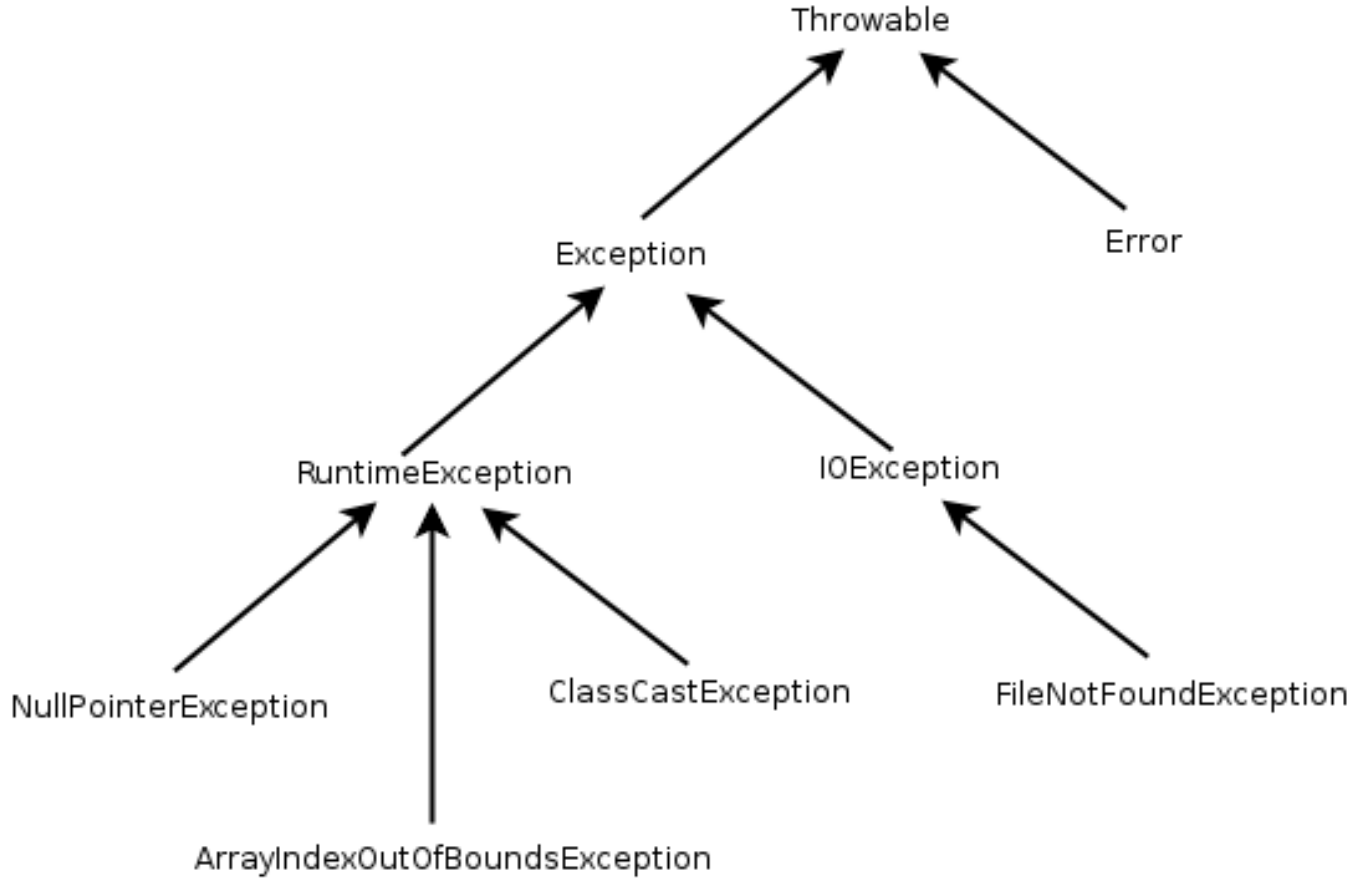
LIDANDO COM EXCEÇÕES

EXCEPTIONS HANDLING

Exception Handling é um mecanismo para lidar com as exceções e erros que ocorrem em tempo de execução.

HIERARQUIA

No ***Java***, os ***erros*** possuem uma certa classificação, que é baseada em hierarquia de classe. Observe:



SOBRE A HIERARQUIA

A principal classe dessa hierarquia é a classe ***Throwable,*** ela representa qualquer tipo de ***erro*** que pode acontecer. Logo abaixo dela, temos duas classes: ***Error*** e ***Exception.***

*Os **erros** são de execução, geralmente gerados por uma situação muito anormal que aconteceu durante a execução daquela aplicação. Ex: "**OutOfMemoryError**" que acontece quando acaba a memória disponível para sua aplicação executar.*

*Já as **Exceptions** podem até ser tratadas pela sua aplicação pois são mais "previsíveis" e podem ocorrer em alguns momentos específicos. Ex: "**SQLException**" que pode acontecer numa consulta ao banco de dados.*

JAVA

ERRORS

- Não são possíveis de capturar.
- Todos são do pacote **java.lang.error**.
- Não são conhecidos pelo compilador, pois acontecem em tempo de execução.
- São causadas pelo ambiente no qual a aplicação está sendo executando.

X

EXCEPTIONS

- São possíveis de capturar.
- Pode ser do tipo ***Checked*** ou ***Unchecked***.
- Todas são do pacote **java.lang.Exception**.
- ***Checked Exceptions*** são conhecidas pelo compilador, enquanto as ***Unchecked*** não são.
- São causadas pela aplicação.

JAVA

CHECKED

1- O compilador te obriga a fazer um certo tipo de tratamento.

2-São filhas de ***Exception*** mas não são filhas de ***RuntimeException.***

3-Não são simples de evitar, então você deve estar preparado caso elas ocorram.

4-Quando você tem uma linha de código ou uma expressão que pode jogar uma ***Exception***, você é obrigado a tratar esse erro de alguma forma, com um ***try- catch*** ou ***throws*** na assinatura do método.

UNCHECKED

1- O compilador ignora.

2-São filhas de ***RuntimeException***, ou seja ocorrem em tempo de compilação da sua aplicação.

3-São mais simples de evitar e você pode fazer alguns tipos de checagem em seu código.

4-Quando você tem uma linha de código, uma expressão que pode jogar uma ***Exception,*** você não precisa tratar com o ***try-catch*** pode somente jogar a exception com o ***throw***.

DRAWING Java

EXCEPTIONS

JAVA

PALAVRAS RESERVADAS

try

É usado para sinalizar um bloco de código que pode gerar uma ***Exception***.

catch

É usado para capturar e tratar a *Exception*.

finally

É usado para executar uma parte importante do programa, mesmo que ocorra uma ***Exception***.

throw

É usado para lançar ou "jogar" uma ***Exception***, fazendo o tratamento ali mesmo.

throws

É usado na assinatura do método para declarar que uma ***Exception*** pode ocorrer, transferindo a responsabilidade de tratá-la para quem for chamar.

DRAWING Java

EXCEPTIONS

JAVA

TOP 10 EXCEPTIONS

1. ***Arithmetic Exception:*** *é lançada em operações matemáticas.*
2. ***ArrayIndexOutOfBounds Exception e IndexOutOfBoundsException:*** *a primeira é lançada ao tentar acessar uma posição inválida de um* ***array*** *e a segunda em uma classe* ***ArrayList****, por exemplo.*
3. ***ClassNotFoundException:*** *é lançada quando uma classe não pode ser encontrada.*
4. ***FileNotFoundException****: é lançada quando um arquivo não é encontrado ou não pode ser aberto.*
5. ***IOException:*** *é lançada quando existe um erro de entrada ou saída de dados.*
6. ***InterruptedException:*** *é lançada quando uma* ***thread*** *está esperando, dormindo ou fazendo alguma outra tarefa, e é interrompida.*
7. ***NoSuchMethodException:*** *é lançada quando não encontrou o método desejado.*
8. ***NullPointerException:*** *é lançada quando o objeto referenciado está nulo, ou seja* ***null****.*
9. ***NumberFormatException:*** *é lançada quando não é possível converter um número.*
10. ***StringIndexOutOfBoundsException:*** *é lançada quando não é possível acessar um índice da String.*

DRAWING Java

EXCEPTIONS

OUTROS ERROS E EXCEPTIONS

1. ***ClassCastException:*** *é lançada quando não é possível fazer o* ***Casting*** *(conversão).*
2. ***StackOverflowError:*** *é lançada, normalmente em momento de recursividade (um método chamando ele mesmo) e a pilha de execução fica muito grande esgotando assim a memória disponível.*
3. ***NoClassDefFoundError:*** *é lançada quando uma não foi encontrada.*
4. ***ExceptionInInitializerError:*** *é lançada quando ocorre uma exception na execução de um bloco estático(**static**) ou na atribuição de valor à variável estática (**static**).*
5. ***IllegalArgumentException:*** *é lançada quando um método é invocado com um parâmetro incorreto.*
6. ***IllegalStateException:*** *é lançada quando um método foi invocado no momento incorreto, ou seja em um estado inválido*
7. ***AssertionError:*** *é lançada para indicar que uma **assertion** (afirmação) falhou.*
8. ***OutOfMemoryError:*** *é lançado quando esgotou a memória disponível.*

JAVA

RESUMINDO...

1. *Uma **Exception** é a maneira de você saber o que deu errado no seu código.*
2. *É muito importante ler a **Exception** para resolver o problema que ocorreu, a própria **API** já te ajuda.*
3. *Além da **Exception** o **Strack Trace** (informações que são mostradas abaixo da **Exception**) informa mais detalhes do seu erro, assim como o número da linha onde eles ocorreram.*
4. *Só é possível usar **try-catch** em uma **Checked Exception** se o código do bloco do **try** pode realmente lançar a **Checked Exception** em questão, caso contrário o compilador avisará com um erro: "unreachable code".*
5. *Tratar uma **Exception** de forma genérica não é uma boa prática, porque quanto mais específico for a **Exception**, mas fácil de identificar o problema.*
6. *Os **errors** não deveriam ser tratados pela aplicação, já que são de responsabilidade da JVM.*
7. *Caso aconteça uma **Exception** as demais linhas do bloco **try** não serão executadas.*
8. *Se a **Exception** que ocorre não foi definida no **catch**, a chamada do método para e volta jogando a **Exception** como se não houvesse um bloco **try-catch**.*
9. *Num bloco **try-multi-catch** (mais de um **catch**) a ordem deles é muito importante, visto que ao tratar uma **Exception** mais específica (que é filha) abaixo de uma mais genérica você terá um erro de compilação.*
10. *Cuidado ao inicializar variáveis que possam lançar uma **Exception** (ao abrir um arquivo por exemplo), nesse caso elas devem ser tratadas no método construtor.*

Gostou do meu Resumo ?

- Dê um like!
- Compartilhe...
- Salve para Depois
- Deixe seu Comentário