Tratamento de Exceção - Java

Estagiário docente: Daniel David Fernandes

Professora: Rita Suzana Pitangueira Maciel

O que são?

- Erro de execução no programa (*runtime error*)
- Uma condição inesperada durante a execução de um programa
- São objetos que sinalizam que ocorreu algum problema no tempo de execução de um programa

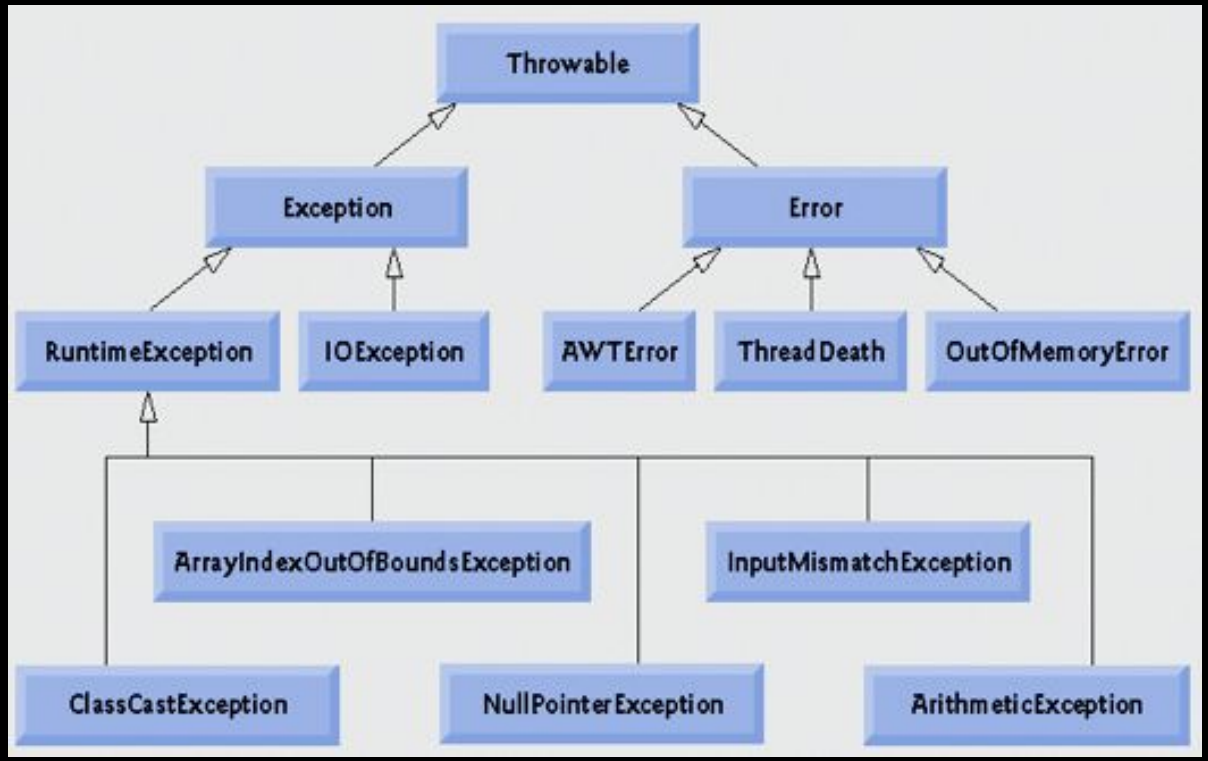
Pra quê são usadas?

- Ajudam a detectar e tratar possíveis erros que possam acontecer
- Forma de ter um controle ou auditoria do que está acontecendo durante a execução.

Onde aparecem?

- Manipulação de dados
    - Manipular um objeto que está com o valor nulo.
    - CRUD no banco de dados;
    - Índices fora do intervalo de array;
    - Cálculos matemáticos;
    - I/O de dados;
    - Erros de conexão de rede;
    - etc.

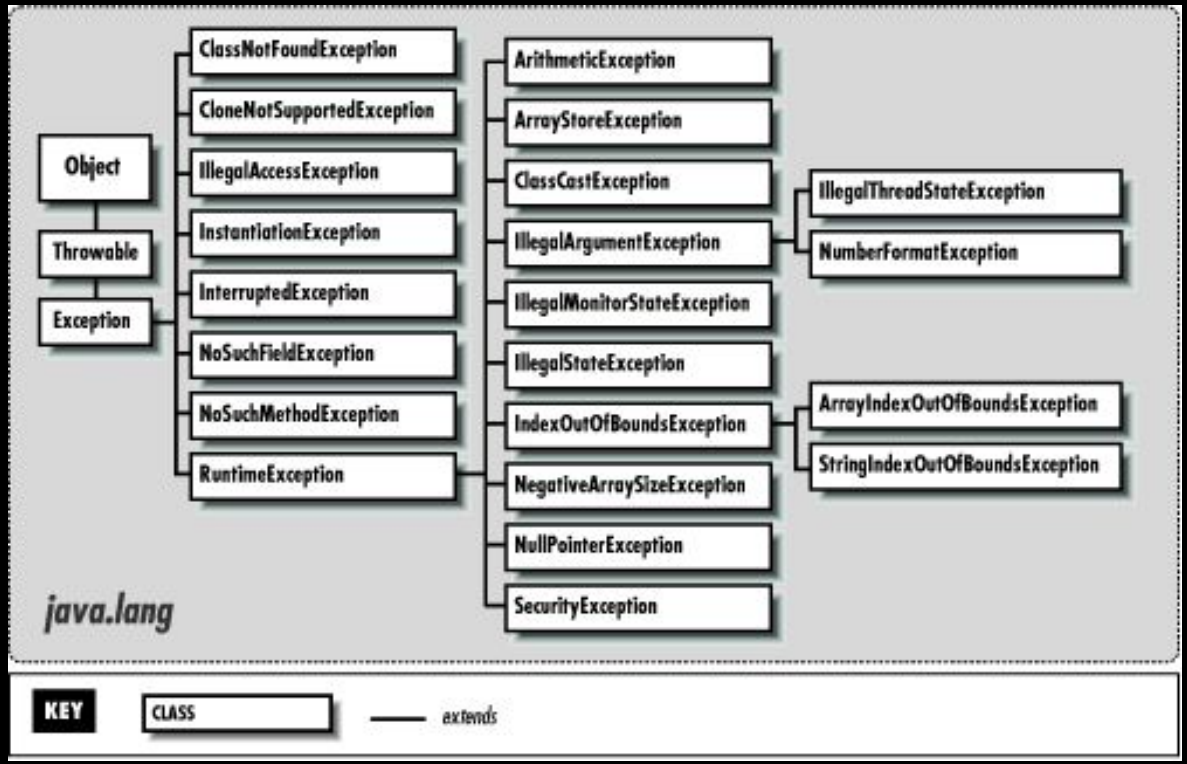
Como funcionam?



https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Como funcionam?

- Observação:
  - Não é necessário importar o pacote java.lang



https://www.devmedia.com.br/trabalhando-com-excecoes-em-java/27601

Como funcionam?

- Exception (java.lang.Exception) – mostra as situações em que a aplicação pode querer capturar e realizar um tratamento para conseguir realizar o processamento.

- Error (java.lang.Error) – indica as situações em que a aplicação não deve tentar tratar, como ocorrências que não deveriam acontecer.

Como funcionam?

- Palavras reservadas (*keywords*):
  - try,
  - catch,
  - finally,
  - throw e throws

Como funcionam?

- try
  - indica que o código está tentando realizar algo arriscado no sistema
- catch
  - trata a exceção lançada
- finally
  - realiza independente de acontecer erro ou não.

Como funcionam?

```
try {
    //trecho de código que pode vir a lançar uma exceção
}
catch(tipo_excecao1 e1) {
    //ação a ser tomada
}
catch(tipo_excecao2 e2) {
    //ação a ser tomada
}
catch(tipo_excecaoX e3) {
    //ação a ser tomada
}
finally {
    //ação que sempre será executada
}
```

Como funcionam?

- throws
  - declara as exceções que podem ser lançadas no método
  - Por que usar?
    - quando não se deseja que uma exceção seja tratada no próprio método, mas em outro que use este método
    - sinalizar para outros desenvolvedores

Como funcionam?

- throws (Exemplo03.java & Exemplo06.java)

```
int fazAlgo() throws tipo_exceção_1, tipo_exceção_2, tipo_exceção_n {

    //[...]

}
```

Como funcionam?

- throw
  - Novo objeto de exceção que é lançado
  - Permite criar sua própria exceção
  - Por que usar?
    - forçar o lançamento de uma exceção
    - exceção padrão em vez de uma específica

Como funcionam?

- throw (Exemplo04.java & Exemplo07.java)

```
try {

    //[...]

}
catch(tipoExcessão_1 e) {

    throw new novoTipoExceçao(e);

}
```

Você pode criar sua própria exceção!

(Exemplo08.java)

Exemplos de exceções

- java.lang.
  - ArrayIndexOutOfBoundsException (Exemplo01.java)
  - StringIndexOutOfBoundsException (Exemplo09.java)
  - NullPointerException (Exemplo02.java)
  - IllegalArgumentException (Exemplo04.java)
  - ClassCastException (Exemplo10.java)
  - ArithmeticException (Exemplo05.java)
  - InputMismatchException (Exemplo05.java)
  - ...

Todas as exceções

https://docs.oracle.com/javase/7/docs/api/java/lang/package-tree.html

Exercício 1

- Crie uma classe Matematica com dois métodos estáticos
  - Calculo de Operações – Recebe como parâmetros dois números reais, mais o símbolo da operação (+, -, *, /)
  - Calculo de Fatorial – Recebe como parâmetro o número inteiro que se quer calcular fatorial
- Veja quais as possíveis condições de erro e lance as devidas exceções
- Faça um programa em outra classe que acesse essas operações e faça o tratamento adequado, impedindo do programa cair e dando uma mensagem consistente ao usuário
  - Esse programa deve receber números pelo teclado para acessar os dois métodos.
  - Utilize o método *nextLine()* da classe *Scanner* para isso e faça tratamentos de possíveis erros na entrada de dados.

Exercício 2

- Crie uma classe Circulo com os seguintes atributos (números reais) com seus respectivos métodos acessores
  - Coordenada x, Coordenada y e Raio
- No método acessor ***set*** de cada um, impeça do objeto receber valores negativos para quaisquer atributos.
  - Não deixe o objeto ser instanciado – ou ter valores de atributos alterados – caso isso ocorra
- O método construtor deve receber os 3 atributos como parâmetros
- Faça um programa em outra classe que instancie objeto da classe ´Circulo e altere os seus atributos, impedindo do programa cair e dando uma mensagem consistente ao usuário
  - Esse programa deve receber números pelo teclado para instanciar objetos e alterar atributos
  - Utilize o método *nextLine()* da classe *Scanner* para isso e faça tratamentos de possíveis erros na entrada de dados.