09

O que aprendemos?

Se você fez o exercício Será que o Miguel entendeu a aula? (https://cursos.alura.com.br/course/java-excecoes/task/37907), vai lembrar o que aprendemos. Para fixar ainda mais, listamos abaixo os tópicos dessa aula:

- Existe uma hierarquia grande de classes que representam exceções. Por exemplo, `ArithmeticException` é filha de `RuntimeException`, que herda de `Exception`, que por sua vez é filha da classe mais ancestral das exceções, `Throwable`. Conhecer bem essa hierarquia significa saber utilizar exceções em sua aplicação.
- `Throwable` é a classe que precisa ser extendida para que seja possível jogar um objeto na pilha (através da palavra reservada `throw`)
- É na classe `Throwable` que temos praticamente todo o código relacionada às exceções, inclusive `getMessage()` e `printStackTrace()`. Todo o resto da hierarquia apenas possui algumas sobrecargas de construtores para comunicar mensagens específicas
- A hierarquia iniciada com a classe `Throwable` é dividida em **exceções** e **erros**. Exceções são usadas em códigos de aplicação. Erros são usados exclusivamente pela máquina virtual.
- Classes que herdam de `Error` são usadas para comunicar erros na máquina virtual. Desenvolvedores de aplicação não devem criar erros que herdam de `Error`.
- `StackOverflowError` é um erro da máquina virtual para informar que a pilha de execução não tem mais memória.
- Exceções são separadas em duas grandes categorias: aquelas que são obrigatoriamente verificadas pelo compilador e as que não são verificadas.
- As primeiras são denominadas *checked* e são criadas através do pertencimento a uma hieraquia que não passe por `RuntimeException`.
- As segundas são as *unchecked*, e são criadas como descendentes de `RuntimeException`.