Exceptions

Kotlin trata as exceções de um jeito muito parecido com Java, **vai jogando-a na pilha até chegar em um ponto que a trate ou explode na última execução da pilha**. Todas as **Exceptions** do **Kotlin** são herdeiras de ***Throwable***.

Kotlin usa a estrutura ***try catch finally,*** a exceção tem os atributos padrões como stack trace, mensagem e etc...:

```
fun converte (funcionarioAny : Any){
    try{
        funcionarioAny as Funcionario
    }catch ( ex: ClassCastException ){
        ex.printStackTrace();
    }finally {
        funcionarioAny.javaClass.hashCode()
    }

}
```

O **Try** no **Kotlin** **pode ser usado como uma expressão** *(try expression)* que retorna valor. Por exemplo, ***pegar um Input e tratar ele quando cair na Exceção e devolver o valor tratado***. A ideia por trás da **Try expression** é tratar um possível erro e já **devolver o valor desejado**:

```
val valorRecebido : BigDecimal? = try{
    BigDecimal(entrada)
}catch (ex : NumberFormatException){
    println("Erro na conversão")
    ex.printStackTrace()
    null
}
```

*****O if também pode ser usado como uma expression*****

Na criação de uma **Exception personalizada** é importante fazer ela herdar de **Exception** e não de ***Throwable***, isso porque **trhowlable é pai tanto de exception quanto de erro (erro é relacionado a JVM):**

```
class SaldoInsuficienteException (
    mensagem : String = "Saldo insuficiente"
) : Exception(mensagem) {

}
```

```
class FalhaAutenticacaoException (
    mensagem: String = "Falha na autenticação"
): Exception (mensagem) {
}
```

```
try {
    marcioConta.transfere(BigDecimal( val: 191), mariaConta, senha: "mario")
}catch (ex:SaldoInsuficienteException){
    ex.printStackTrace()
}catch (ex : FalhaAutenticacaoException){
    ex.printStackTrace()
}catch (ex : Exception){
    println("Erro desconhecido")
    ex.printStackTrace()
}
```