02

Descreva o que são exceções e para que são utilizadas em Java

Imagine a situação em que tentamos acessar uma posição em um array:

```
public void fazAlgo(int[] idades) {
    System.out.println(idades[0]);
}
```

COPIAR CÓDIGO

O que acontece se o array enviado para o método é vazio? Se esse código imprimisse nulo ou um número padrão, nesse caso, teríamos sempre que nos preocupar, como em:

```
public void fazAlgo(int[] idades) {
    if(idades[0]==null) return;
    // return para caso o Java devolva nulo ao acessar
    // uma posição inválida

    System.out.println(idades[0]);
}
```

COPIAR CÓDIGO

Pense como seria difícil tratar todas as situações possíveis que *fogem do padrão* de comportamento que estamos desejando. Nesse caso, o comportamento padrão, aquilo que acontece 99% das vezes e que esperamos que aconteça é que a posição acessada dentro do array seja válido. Não queremos ter que verificar toda vez se o valor é válido, e não queremos entupir nosso código com diversos `ifs`

para diversas condições. As exceções à regra, as *exceptions*, são a alternativa para o controle de fluxo: em vez de usarmos `ifs` para controlar o fluxo que foge do padrão, é possível usar as *exceptions* para esse papel. Veremos adiante como tratar erros, como o acesso a posições inválidas, tentar acessar variáveis com valores inválidos etc.

Caso uma exception estoure e sua *stack trace* seja impressa, teremos algo como:

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
    at SuaClasse.fazAlgo(SuaClasse.java:20)
    at SuaClasse.main(SuaClasse.java:30)
```

COPIAR CÓDIGO

Note como a *stack trace* indica que método estava sendo invocado, em qual linha do arquivo fonte está essa invocação, quem invocou este método etc.

O importante é lembrar que as *exceptions* permitem que isolemos o tratamento de um comportamento por blocos, separando o bloco de lógica de nosso negócio do bloco de tratamentos de erros (sejam eles *Exceptions* ou *Errors*, como veremos adiante). O stack trace de uma `Exception` também ajuda a encontrar onde exatamente o problema ocorreu e o que estava sendo executado naquela *Thread* naquele instante.