05

Corrigindo exceptions

Transcrição

Vamos verificar nosso código para encontrar o que pode gerar um erro? Será que estamos presumindo um comportamento do cliente que na verdade é diferente?

Analisaremos linha a linha. É assim que devemos fazer na vida real. Não olhe por cima, olhe sempre linha a linha.

```
public static void main(String[] args){

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
      int menor = scanner.nextInt();
      int maior = scanner.nextInt();

      int maiorCicloAteAgora = 1;
      for(int atual = menor; atual <= maior; atual++){
        int resultado = calculaPara(atual);
        if(resultado > maiorCicloAteAgora){
          maiorCicloAteAgora = resultado;
        }

      }
      System.out.println(i + " " + j + maiorCicloAteAgora);
    }

    private static int calculaPara(int i){
      int impressos = 1;
      //System.out.println(n);
      while (n != 1) {
              if(n % 2 == 1) n = n * 3 + 1;
            else n = n / 2;
               impressos++
              //System.out.println(n);
      }
      return impressos;
    }
```

Primeio criamos um `Scanner`. Poderia dar um `execption` no `System. in`, mas seria um erro muito grave do sistema operacional. A linha seguinte pergunta se há próxima linha `hasNextLine`. Podemos perguntar isso? A princípio, sim. Se houver próxima linha, verificamos se há dois inteiros. Vamos ver o caso de `entrada1.txt`.

```
1 10
100 200
201 210
900 1000
```

Para essa entrada tudo parece funcionar bem.

```
Alura-Azul:bin alura$ time java Main < entrada1.txt
1 10 20
100 200 125
201 210 89
900 1000 174

real  0m0.134s
user  0m0.121s
sys   0m0.031s
```

Quando nosso programa lê os dois números finais, ele lê a linha. Se tiver mais uma linha, o que acontece? Vamos criar a `entrada3.txt` para testar essa possibilidade. Vai ser igual à primeira entrada, mas com uma linha em branco no final.

```
1 10
100 200
201 210
900 1000
```

Será que o programa vai consumir a última linha depois do inteiro `1000` ? Vamos testar no terminal.

```
Alura-Azul:bin alura$ time java Main < entrada3.txt
1 10 20
100 200 125
201 210 89
900 1000 174
Exception in thread "main" java.util.NoSuchElementException
        at java.util.Scanner.thorwFor(Scanner.java:862)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.ajva:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Main.main(Main.java:16)

real  0m0.137s
user  0m0.125s
sys   0m0.032s
```

Deu o erro `NoSuchElementException` . Pode ser que estivéssemos errando na entrada. Assumimos que o programa podia verificar se tinha uma linha e então ler os inteiros. É preciso tomar muito cuidado. No último programa que atacamos, misturamos `hasNextLinext` com `nextInt` e vimos que era perigoso. E agora fizemos de novo, para verificar se havia mais elementos. Podemos usar `hasNext` em vez de `hasNextLine` .

```
public static void main(String[] args){

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNext()){
      int menor = scanner.nextInt();
      int maior = scanner.nextInt();

      int maiorCicloAteAgora = 1;
      for(int atual = menor; atual <= maior; atual++){
        int resultado = calculaPara(atual);
        if(resultado > maiorCicloAteAgora){
```

```
            maiorCicloAteAgora = resultado;
          }

        }
        System.out.println(i + " " + j + maiorCicloAteAgora);
      }

      private static int calculaPara(int i){
        int impressos = 1;
        //System.out.println(n);
        while (n != 1) {
                if(n % 2 == 1) n = n * 3 + 1;
              else n = n / 2;
                 impressos++
                //System.out.println(n);
        }
        return impressos;
      }
```

Vamos testar?

```
Alura-Azul:bin alura$ time java Main < entrada3.txt
1 10 20
100 200 125
201 210 89
900 1000 174

real  0m0.136s
user  0m0.122s
sys   0m0.032s
```

Agora funcionou. Pode ser esse o erro que o juiz online encontrou. O exemplo que ele testou pode ter rodado e funcionado, até uma quebra de linha. E o que devemos fazer como programadores? Já precisamos fazer uma mudança na leitura das duas primeiras linhas de código. Podemos ou nos dar por satisfeitos, ou continuar analisando o código em busca de outros erros.

Qual você escolhe? Programadores têm que tomar essas decisões. O cliente falou que há bug no código, e quando o verificamos, encontramos um bug. Esse bug é o mesmo que o cliente encontrou? Não temos certeza.

Vamos copiar esse código e enviar novamente para o UVa. Quando vamo em `My Submissions` verificar, vemos:

A resposta ainda está errada! Minha leitura realmente estava errada. Se olharmos a definição de input do problema:

> The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 10,000 and greater than 0.

Ele fala que haverá um par de inteiros por linha, dando a entender que não há linhas vazias. Mesmo assim, parece que tinha um erro na maneira em que estávamos lendo. Mesmo quando o cliente dá a entender certas características, no dia a dia, como programadores seniores (cuja experiência nos diferencia dos programadores básicos) temos que pensar no que está acontecendo e no que houve no passado e tentar prever os problemas do código. O sucesso do código o próprio cliente consegue prever, ao nos pedir o que o programa deve fazer.

O cliente está falando que virão dois inteiros por linha. Mas e se não vierem? É melhor cada inteiro, e quando não houver mais, podemos parar. Assim, mesmo se houver linha sobrando, não teremos problema. Você pode até comunicar o cliente falando "Estou fazendo dessa maneira, então se você colocar linhas extras, não teremos problemas". Ele provavelmente responderá que tudo bem, e enquanto isso você pode ir implementando.

Repare que esses são detalhes pequenos e simples. Se fosse algo de maior complexidade, não devemos implementar antes da aprovação do cliente. A troca do `hasNextLine` por `hasNext` é bem simples, e em vez de ler a próxima linha, lê o próximo elemento.

Conseguimos corrigir aquela `Exception`. Mas a resposta ainda está errada. Alguma entrada ainda está gerando erro. Sequer entramos no caso da `entrada2.txt`, que demora demais, e nos casos normais já temos erro.

O que pode ser esse erro? Já vimos que há o problema de demorar tempo demais, que ainda não atacamos, e esse que faz o juiz considerar nossa resposta errada. O que podemos estar lendo ou cuspindo de errado? Releia as especificações de input e output do problema e veja alguma interpretação que fiz errada de algo que está implícito. De algo que assumi por nós, mas que não estava explicitado no enunciado. Dê uma lida e veja se descobre o que eu assumi que era verdade, mas que não é. Até a próxima!