04

Problema 42: Pensando na condição de parada

Transcrição

Para resolver o problema, tivemos que prestar atenção em diversos pontos. A começar pela leitura completa do enunciado, e saber que há situações que são explícitas – como escrito no enunciado "os números são inteiros". Se não estivesse assim, poderiam ser decimais também.

Será que o nosso programa funciona com decimais? Por exemplo, o `12,5`. Vamos testar? Criaremos um novo arquivo, clicando com o botão direito no diretório e em `New > File`, chamado `entrada3.txt` para fazer esse teste.

```
12,5
```

Vamos agora para o terminal rodar esse teste.

```
Alura-Azul:bin alura$ java Test < entrada3.txt
Error: Could not find or load main class Test
```

Erramos o nome da classe, que agora é `Main`. Corrigindo:

```
Alura-Azul:bin alura$ java Main < entrada3.txt
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:846)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nexInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Main.main(Main.java:7)
```

Nós pedimos um `int`, mas temos um `double`, e por isso não funcionou. Mas, na verdade, isso não importa. Porque o nosso cliente especificou que eram números inteiros. Assim, não precisamos nos preocupar com `entrada3.txt`. Mas, se ele não falasse que era um número inteiro, poderiam ser números decimais. Mesmo que no exemplo dele só aparecessem números inteiros.

Se o cliente não especificar, fica aberto. Lembre-se que ele não usará todas as palavras possíveis para descrever todas as situações do mundo. Geralmente ele tentará explicar tudo, mas ficam alguns buracos. Nesse caso, não há dúvidas na questão do decimal; está explícito que é inteiro.

Mais uma questão: e se o número `42` se repetir? Criaremos a `entrada4.txt` para testar.

```
1
2
88
42
99
54
43
23
```

```
42
43
53
12
```

Temos dois números `42` aqui. Qual será o resultado disso? Vamos testar no terminal:

```
Alura-Azul: bin alura$ java Main < entrada4.txt
1
2
88
```

O resultado mostra só até o primeiro 42. Isso é um problema? Depende do que o cliente pediu. Como no pedido está escrito "Pare de processar o input depois de ler o número 42", não importa se há `42` novamente depois. Então não há problema, pois podemos assumir que o programa deve parar no primeiro `42` – independentemente de quantos `42` houver. E isso é corroborado pelo fato de o nosso programa ter sido aprovado no site.

Mas poderia ser que o cliente não queria dizer isso. Então cabe a nós, programadores, estarmos atentos a isso, às condições que o cliente nos dá. Será que é isso mesmo que ele quer dizer? Será que ele está pedindo para parar no primeiro `42` ou no último? Às vezes os clientes esquecem de falar algo para a gente, e precisamos confirmar com ele. Ele pode não tem considerado que poderia haver outro `42` . Então, perguntamos para ele, porque, se for para parar no último, o código será totalmente diferente.

Temos que fazer esse tipo de questionamento e de teste sempre que lemos um novo problema. Nesse caso está tudo bem definido, mas veremos outro que tem coisas em aberto. Criaremos a `entrada5.txt` , e ele será um arquivo vazio.

Relendo o enunciado, não vemos nada sobre um arquivo vazio. Ele já presume que haverão números no arquivo. O que fazer se não há números? O que o nosso código faz se não tem números? Vamos para o terminal.

```
Alura-Azul: bin alura$ java Main < entrada5.txt
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:862)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nexInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Main.main(Main.java:7)
```

Deu erro! Será que era isso que o cliente esperaria? Será que esse teste é válido para o nosso cliente? Parece que sim, dado o enunciado. Mas o que ele esperava que fizéssemos? O programa deveria imprimir nada em vez de dar erro? Ou ele esperava esse erro mesmo?

Ao ler o pedido do cliente, o programador deve extrapolar todas as condições mencionadas pelo cliente. Quando ele diz "reescreva números pequenos da entrada para a saída", tínhamos que ter pensado "Quantos números?". Mais de um? Só um? Nenhum? Infinito? Não existe quantidade negativa, então isso não é um problema. De qualquer forma, temos que fazer essas questões para o cliente. Aparentemente o juiz do Spoj não considerou esse caso de arquivo vazio, até porque ele roda em um único arquivo. Pelas definições do enunciado, o cliente poderia ter tido esse problema, mas não deu instruções do que fazer no caso.

Nas maratonas de programação, se não está implícito nem explícito o que fazer nesses casos, é porque não será cobrado. Mas é preciso ter muita atenção para o caso de estar implícito, que aí precisamos perceber. Mas, na vida real,

pode não estar implícito nem explícito, e você precisará perguntar para o cliente. Eu costumo fazer da seguinte maneira: se penso em um caso que o cliente não mencionou, implemento a solução mais simples possível, que exija o mínimo do meu código. E antes de implementar, mando um email perguntando a ele "Será que pode haver um caso de arquivo vazio?". Como eu não sei se ele vai responder o email em um minuto ou uma hora, faço a solução mais simples. Do contrário, posso acabar perdendo tempo.

Para encontrar a solução mais simples, vamos rever o código.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);
      if(scanner.hasNext()) break;

    }

  }

}
```

Nós pedimos para o programa ler o próximo número inteiro. Podíamos primeiro pedir para ele verificar se existem números para ler, com a linha `if(scanner.hasNext()) break;` .

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      if(scanner.hasNext()) break;

      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);
      if(scanner.hasNext()) break;

    }

  }

}
```

Com essa decisão que tomamos, o programa não fará nada com o arquivo vazio. Vai perceber que ele é vazio e parar. Olhando novamente o código, vemos que está sujo. O começo e o fim do laço são a mesma linha. Podemos remover a última linha:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      if(scanner.hasNext()) break;

      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);

    }

  }

}
```

Vamos testar no terminal.

```
Alura-Azul: bin alura$ java Main < entrada5.txt
```

Não deu erro e não retorna nada: está funcionada. E os demais casos?

```
Alura-Azul: bin alura$ java Main < entrada5.txt
Alura-Azul: bin alura$ java Main < entrada4.txt
1
2
88
Alura-Azul: bin alura$ java Main < entrada3.txt
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:846)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nexInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Main.main(Main.java:9)
```

O caso 3 dá erro, mas não é um problema, porque o cliente explicitou que serão apenas números inteiros.

```
Alura-Azul: bin alura$ java Main < entrada5.txt
Alura-Azul: bin alura$ java Main < entrada4.txt
1
2
88
Alura-Azul: bin alura$ java Main < entrada3.txt
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:846)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nexInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at Main.main(Main.java:9)
```

```
Alura-Azul: bin alura$ java Main < entrada2.txt
1
2
88
Alura-Azul: bin alura$ java Main < entrada1.txt
1
2
89
```

Todos os demais casos estão funcionando. Tem mais um detalhe que não gosto no código. Do jeito que o laço está escrito, existe uma possibilidade de não percebermos que ele é infinito e que o programa vai rodar para sempre. Um laço cuja condição é sempre verdadeira (`true`) exige um break no meio.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(true) {
      if(scanner.hasNext()) break;

      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);

    }

  }

}
```

Se não tiver nenhum `break` ou `return` no meio do laço, ele vai rodar eternamente, e podemos travar, por exemplo, uma thread do celular. Temos que tomar muito cuidado com o laço `while(true)` , pois se escrevermos o `break` incorretamente, o programa ficará rodando. A regra geral é nunca escrever a palavra `true` em um laço, para não correr risco. Escrevemos o `while` associado à condição pertinente para o caso. Aqui, seria ter mais alguma coisa para ler, ou seja `scanner.hasNext()` , e com isso já podemos eliminar a linha do break.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNext()) {

      int numero = scanner.nextInt();
      if(numero == 42) break;

      System.out.println(numero);

    }

  }
```

```
}
```

Se o `Scanner` tiver algo para ler, o laço continua. Do contrário, ele para. Jamais use o `while(true)`. Pode haver um caso muito específico onde ele seja pertinente, mas em geral: se você escreveu um `while(true)`, apague. É preciso tomar cuidado com a condição final de parada do laço. No nosso caso, colocamos uma condição tanto de parada quanto de início, que é a verificação de ter algo para ler. E a condição final também é ter o número `42`.

Sempre que se cria um laço, é preciso parar para pensar. Se você começou a digitar `while` ou `for`, pare e pense:

1. Qual a condição inicial? Do que eu tenho certeza quando esse laço começa?
2. Qual é a condição final? O que precisa acontecer para esse laço terminar?

É preciso ter certeza de que você não está assumindo nada errado na condição inicial, como eu fiz, assumindo que haveria números em todos os arquivos. E também na condição final, que são duas: não ter mais nada para ler e o número `42`. Esta estava escrita no enunciado, mas a primeira não. Essa tivemos que descobrir testando.

Com a solução desse problema pude mostrar diversas técnicas que podemos utilizar na criação de um laço e na leitura da entrada. São duas coisas que fazemos direto no dia a dia que exigem atenção às suas condições. Em breve resolveremos problemas mais difíceis. Até mais!