Exampe Especial
Algoritmos e Estruturas de Dados I

Prof.: Carlos Camarão

14 de Julho de 2006

1. Escreva um programa que leia um valor inteiro positivo n, calcule o valor de $\sum_{i=1}^{n}(-(i+1)/(i*i))$, e imprima esse valor. Decomponha o seu programa em três partes, cada uma delas implementada por um método: leitura, cálculo do valor desejado a partir do valor lido, e impressão do valor calculado.

   O programa (isto é, o método de leitura) deve ler uma cadeia de caracteres e causar uma exceção se essa cadeia não representar um número inteiro positivo. Nesse caso, a exceção deve ser tratada no programa principal, e esse tratamento deve apenas emitir uma mensagem de erro apropriada, e em seguida a execução do programa deve ser terminada.

2. Faça um programa que leia dois números inteiros positivos, n e m, e n seqüências de m valores inteiros, e imprima, para i variando de 1 a m, a soma dos i-ésimos inteiros de cada uma das n sequências.

   Por exemplo, se os valores fornecidos como entrada forem:

   3
   2
   10
   5
   2
   20
   3
   30

   o programa deve imprimir o seguinte: 1: 15 2: 55

   Note que 15 é obtido pela soma (10+2+3) e 55 por (5+20+30).

   Em cada situação de entrada de dados, se um valor não esperado for fornecido com entrada (por exemplo uma cadeia de caracteres que não representa um número inteiro), uma exceção deve ser causada, e o dado incorreto deve ser fornecido novamente pelo usuário do programa.

   O processo de nova entrada de dados, depois de uma entrada de um dado incorreto, deve ser iniciado em um tratador de exceção.

Nota: o método *parseInt* causa uma exceção (*NumberFormatException*) se a cadeia de caracteres fornecida como argumento não representar um número inteiro.

Não esqueça de testar se os dois primeiros valores lidos (de n e m) são valores positivo, e causar uma exceção em caso contrário. Você pode causar e tratar apenas a exceção *Exception*, pois a exceção *NumberFormatException* é um objeto da classe *Exception*, e portanto será também tratada por um tratador que trata (recebe como argumento) uma exceção *NumberFormatException*.

3. Escreva o corpo do método:

```
static void jogadaComputador (int[] jogo)
```

tal que `jogadaComputador` realize a jogada "ótima" do jogo `Nim`, se essa jogada puder ser feita e, em caso contrário, retire um símbolo da linha que contém maior número de símbolos.

Lembre-se: como foi explicado em sala de aula e no enunciado do trabalho, i) a jogada ótima existe se e somente se a soma binária *sb* (ou-exclusivo da representação binária do número de símbolos) de todas as linhas for diferente de zero. Nesse caso, ii) a linha de onde se deve retirar símbolos é qualquer linha i na qual $x =$ `sb ^ jogo[i]` é menor do que `jogo[i]` (onde `^` faz o ou-exclusivo da representação binária dos dois operandos inteiros). Além disso, iii) o número de símbolos que se deve deixar nessa linha é `sb ^ jogo[i]`.

Por exemplo, se temos:

Linha	Número de símbolos	Em binário
0	1	1
1	2	01
2	3	11
3	4	100
4	5	101
5	6	110

A soma binária é 100, e as linhas em que o valor x decresce são as três últimas. Por exemplo, podemos deixar, na última linha, `100 ^ 110 = 010` (2 símbolos), e com isso a soma binária de todas as linhas vai passar a ser igual a zero.

Nota: o operador `^` da linguagem Java faz a operação de ou-exclusivo na representação binária dos dois argumentos inteiros.