2º Teste de "Introdução à Programação" (2009/2010)
6 de Janeiro de 2010
Duração: 2:00H

Instruções muito importantes:

- Responda a cada grupo em folhas **separadas**.
- Verifique que **todas** as folhas estão identificadas com o número de caderno.
- Antes de começar a resolver, leia o enunciado do **princípio até ao fim**.
- **Pode** usar caneta ou lápis.
- Não é permitido consultar quaisquer elementos para além deste enunciado.
- Qualquer tentativa de fraude comprovada acarretará a reprovação na disciplina.
- Não é permitido sair da sala antes que o exame termine.

I. Pretende-se programar em Java uma classe **`Book`** cujos objectos representam livros. Cada livro é caracterizado pela seguinte informação: título do livro, autor (assuma que existe um único autor por livro), editora, número de livros vendidos, e ISBN.

O construtor da classe **`Book`** é definido da seguinte forma:

- **`public`** `Book(String title, String author, String editor, int sales, String isbn)`: Constrói um objecto **`Book`** com o título `title`, autor `author`, editora `editor`, número de livros vendidos `sales`, e ISBN `isbn`;

A interface pública da classe **`Book`** é definida da seguinte forma:

- **`public`** `String getEditor()`: Devolve o editor do livro;
- **`public`** `int getSales()`: Devolve o número de livros vendidos;
- **`public`** `void sell()`: Vende um exemplar do livro;
- **`public`** `String toString()`: Devolve uma String com alguns dos dados do livro, nomeadamente o autor, título, editor e ISBN, por esta ordem, separados por ponto e vírgula. Por exemplo:

  `"Gaspar Belchior;Alguém viu o Baltazar?;Livros dos Reis Editora;978-3-16-148410-0"`

- **`public`** `boolean equals(`**`Book`**`b)`: Indica se livros são iguais. Considere que dois livros são iguais quando têm o mesmo ISBN (Nota: o ISBN é um sistema identificador único para livros e publicações não periódicas).

Programe, em Java, a classe **`Book`**.

II. Para construir um gestor de uma livraria é preciso definir uma classe **`BookShelf`** que representa um conjunto de registos de livros. Cada registo é representado por um objecto da classe **`Book`**. É importante garantir que o registo de um livro não aparece duplicado na livraria. Para esse efeito considera-se que não existem dois registos iguais na livraria, onde "iguais" é verificado com o método `equals` da classe **`Book`**. O construtor da classe **`BookShelf`** é:

- **`public BookShelf (int n)`**: Constrói uma nova **`Bookshelf`** que deve poder conter até `n` registos.

A interface da classe **`BookShelf`** deve conter os seguintes métodos:

- **`public int`** `soldBooks()`**:** Devolve o número de livros vendidos na livraria;
- **`public boolean`** `contains(`**`Book`** `b)` : Devolve **`true`** se o livro `b` pertence à livraria, e **`false`** caso contrário;
- **`public boolean`** `addBook(`**`Book`** `b)` : Adiciona o livro `b` à livraria, se possível, devolvendo **`true`**. Caso o mesmo livro já exista na livraria, ou caso a livraria não possa admitir mais livros, esta operação não faz nada e devolve **`false`**.
- **`public String`** `listBooksByEditor(String editor)`**:** Devolve uma String com as lista dos livros publicados pela editora `editor`, um por linha, no formato gerado pela operação `toString()` da classe `Book`;

a) Programe a classe **`BookShelf`** em Java.

b) Imagine que, alguns meses depois de você criar o programa original, o dono da livraria lhe pede para acrescentar uma nova funcionalidade à classe. Ele pretende saber qual a editora com mais livros publicados. Adicione à sua classe o método `editorWithMostBooks()`, sabendo que um colega seu já adicionou à classe **`BookShelf`** os métodos:

   - **`public void`** `initializeEditorIterator()`**:** inicializa o iterador;
   - **`public String`** `nextEditor()`**:** obtém o próximo editor;
   - **`public boolean`** `hasNextEditor()`**:** verifica se existe mais algum editor.

   Nesta pergunta, não precisa de implementar os 3 métodos do iterador de editores acima descritos, e não tem de se preocupar com as variáveis extra que seriam necessárias. Implemente apenas o método `editorWithMostBooks()`, **usando os métodos do iterador como métodos auxiliares.**

   - **`public String`** `editorWithMostBooks()`**:** Devolve a editora que publicou o maior número de livros. Em caso de empate deve devolver uma qualquer editora entre os empatados; caso não exista nenhum registo de livro, devolve a String vazia.

III. Escreva um programa principal em Java que leia, do ficheiro de texto "bookShelf.txt", um conjunto de livros e os insira numa livraria (representada por um objecto **`BookShelf`**).

Depois de ler a informação do ficheiro, o programa deverá pedir ao utilizador o nome de uma editora e, de seguida, apresentar na consola a lista dos livros publicados por essa editora.

O ficheiro de texto "bookShelf.txt" tem o seguinte formato: na primeira linha aparece um número inteiro positivo N que indica o total de livros. Nas seguintes linhas aparecem as informações dos vários (N) livros. Para cada livro aparece o seu título, o autor, a editora, o número de vendas e o ISBN do livro, todos estes elementos em linhas diferentes. Por exemplo:

```
5
Memorial do Convento
José Saramago
Editorial Caminho
34
978-8-52-860022-3
O Jogo do Anjo
Carlos Ruiz Záfon
Dom Quixote
20
978-9-72-203707-5
Mar me quer
Mia Couto
Editorial Caminho
6
978-9-72-211374-8
A Lâmpada de Aladino
Luís Sepúlveda
Porto Editora
12
978-9-72-004183-8
Desgraça
J. M. Coetzee
Gradiva
24
978-972-201827-2
```

IV. Um **número prático** é um inteiro positivo n tal que todos os números positivos inferiores a n podem ser definidos como uma soma distinta dos divisores de n. Por exemplo, o número 12 é um número prático, visto que todos os números de 1 a 11 podem ser expressos como uma soma dos divisores de 12 (que são, como sabe, 1, 2, 3, 4, 6 e 12).

Na resolução deste exercício, tenha em atenção a seguinte propriedade dos números práticos:

- Se n tem os divisores $1 = d_1 < d_2 < d_3 < \ldots < d_c = n$ então n é um número prático se e só se:

  $\sum_{i=1}^{r} d_i \geq d_{r+1} - 1$ para todo o r < c-1, onde c é o número de divisores.
- Tendo em conta a propriedade anterior, para determinar se o número 12 é um número prático (com os divisores $1 < 2 < 3 < 4 < 6 < 12$, ou seja, $d_1=1$, $d_2=2$, $d_3=3$, $d_4=4$, $d_5=6$ e $d_6=12$) será necessário verificar que:
  $d_1 \geq d_2 - 1$, $1 \geq 2 - 1$;
  $d_1 + d_2 \geq d_3 - 1$, $1 + 2 \geq 3 - 1$;
  $d_1 + d_2 + d_3 \geq d_4 - 1$, $1 + 2 + 3 \geq 4 - 1$;
  $d_1 + d_2 + d_3 + d_4 \geq d_5 - 1$, $1 + 2 + 3 + 4 \geq 5 - 1$;

Considere o fragmento da classe **`PracticalNumberChecker`** apresentada de seguida:

```
public class PracticalNumberChecker {
     public static final int SIZE = ...; //Grande q.b. :-)
     private int number; // número que queremos verificar
     private int[] divisors; // divisores de number
     private int nrDivisors; // número de divisores

     public PracticalNumberChecker(int number) {
          this.number = number;
          this.divisors = new int[SIZE];
          this.nrDivisors = 0;
          determineDivisors();
     }
```

Complete a classe **`PracticalNumberChecker`** com os seguintes métodos:

a) Programe o método:
   - **`private void`** `determineDivisors()`: Determina e adiciona ao vector `divisors` os vários divisores de `number`. Para simplificar, assuma que a dimensão `SIZE` do vector é suficiente para armazenar os divisores de qualquer número inteiro positivo que queiramos testar.

b) Programe o método:
   - **`public boolean`** `isPractical()`: Determina se o número em questão é ou não um número prático.