Número: _____ Nome: _____

Exame de "Introdução à Programação" (2013/14 – 1º Semestre)

6 de Janeiro de 2014

Duração 2H

Instruções importantes:

- Responda a cada pergunta no espaço atribuído para o efeito.
- Verifique que **todas** as folhas estão identificadas com o seu nome e número.
- **Pode** usar caneta ou lápis.
- Não é permitido consultar quaisquer elementos para além deste enunciado.
- Não é permitido sair da sala antes que o teste termine.

Considere o seguinte enunciado:

Objectivo:

Gerir os produtos à venda em determinada loja.

Descrição:

A loja pode armazenar até 1500 produtos diferentes. No espaço de venda estão no máximo 100 produtos. Um cliente pode comprar qualquer quantidade de dado produto, desde que exista na loja (em exibição e/ou em stock).

Cada produto é identificado pelo seu código único (um valor inteiro positivo) e tem uma descrição, preço e quantidades na loja (em stock e em exibição).

Todos os dias a loja repõe produtos em exibição, retirando-os do stock.

Interação com o utilizador:

É feita por um interpretador de comandos, onde se pode vender um dado produto a partir do seu código ("V"), repor um dado produto em exibição desde que exista em stock e exista espaço na loja ("R"), repor um produto em stock na loja ("N"), listar todos os produtos em venda ("L"), gravar no ficheiro "vendas.txt" por cada produto da loja a quantidade vendida ("G") e sair da aplicação ("S").

Número: _____ Nome: _____

Grupo I

Complete a seguinte implementação da classe *Produto* **(5 valores)**:

```
public class Produto{
  /** Constantes  e Variáveis de instância **/
```

```
/**Constructor – cria produto colocando a quantidade em exibição a zero
 * @pre
```

```
 */
public Produto(int codigo, String descricao, float preco, int quantStock){
```

```
}
/* Consulta o código do produto
 */
public int getCodigo(){
    return codigo;
}
/* Consulta a descrição do produto
 */
public String getDescricao(){
    return descricao;
}
/* Consulta o preço do produto
 */
public float getPreco(){
    return preco;
}
/* Consulta a quantidade em stock do produto
 */
public int getQuantStock (){
    return quantStock;
}
/* Consulta a quantidade em exibição do produto
 */
public int getQuantExibe (){
    return quantExibe;
}
/* Altera a quantidade em stock do produto, incrementando-a com o valor dado
 * @pre
```

```
 */
public void IncQuantStock (int quant){
```

```
}
/* Incrementa o número de unidades em exibição do produto, retirando a quantidade
   indicada do stock
 * @pre getQuantStock() >= quant && quant > 0
 */
public void IncQuantExibe (int quant){
```

```
}
/* Vende uma dada quantidade do produto, se houver suficiente na loja
 * @pre
```

```
 */
public float vendeProd (int quant){
```

Número: _____ Nome: _____

```
}
```

Grupo II

Considere a classe *Loja* que gere as vendas dos produtos de dada loja. Cada objecto desta classe caracteriza-se por uma colecção de produtos e pelo valor em caixa resultante das vendas. Complete a seguinte implementação da classe *Loja* **(8 valores)**:

```
public class Loja{
    /** Constantes e Variáveis de instância **/
```

```
    /*
     Constructor – Inicializa a colecções de produtos e coloca o valor em caixa
     a zero Euros.
     */
    public Loja(){
```

```
    }
    /* Verifica se dado produto é vendido na loja
    */
    public boolean eVendido(int codigo){
```

```
    }
    /* Coloca um novo produto no stock da loja
     * @pre
```

```
    */
    public void junta(int codigo, String descricao, float preco, int stock){
```

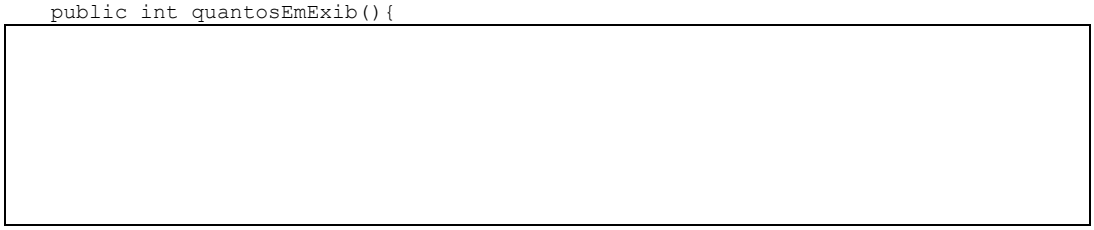
```
    }
    /* Incrementa a quantidade em stock de determinado produto vendido na loja
     * @pre
```

```
    */
public void reporStock(int codigo, int quant){
```

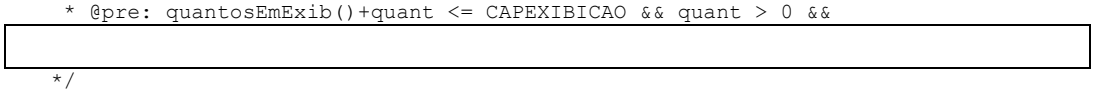
```
    }
```

Número: _____ Nome: _____

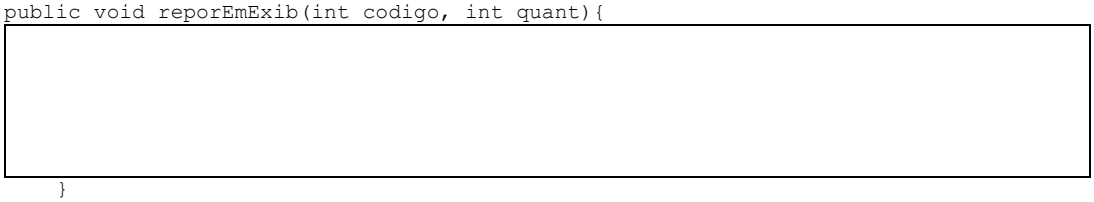
```
    /* Consulta a quantidade de produtos em exibição
     */
    public int quantosEmExib(){
```



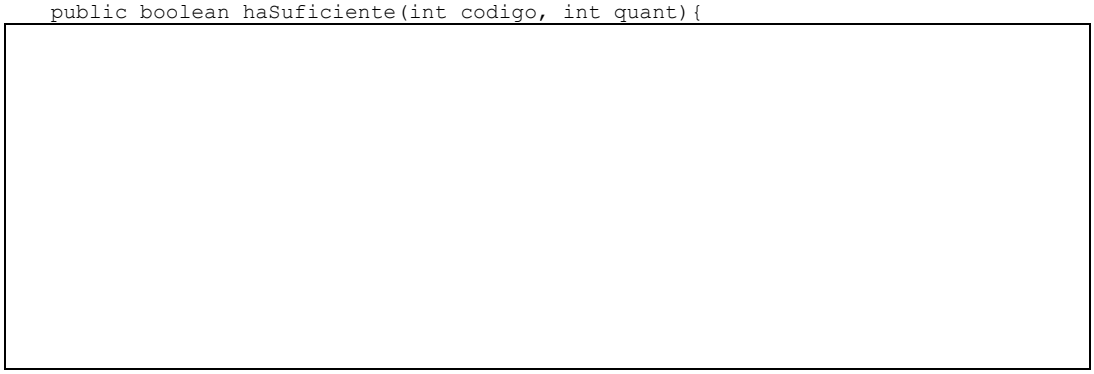
```
    }
    /* Incrementa a quantidade em exibição de determinado produto em stock vendido
       na loja
     * @pre: quantosEmExib()+quant <= CAPEXIBICAO && quant > 0 &&
```



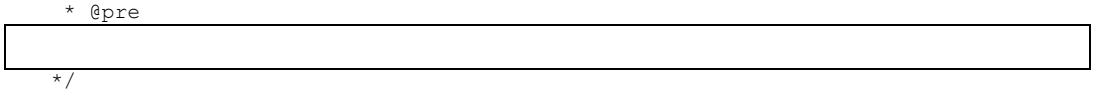
```
    */
public void reporEmExib(int codigo, int quant){
```



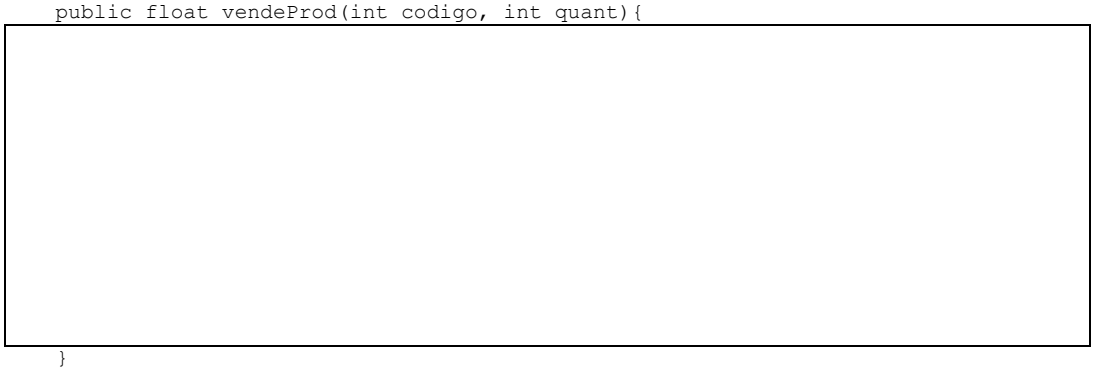
```
    }
    /* Indica se há produto suficiente para fazer venda, i.e., a quantidade em
       stock e em exibição chega para a venda
     * pre: eVendido(codigo) && quant >= 0
     */
    public boolean haSuficiente(int codigo, int quant){
```



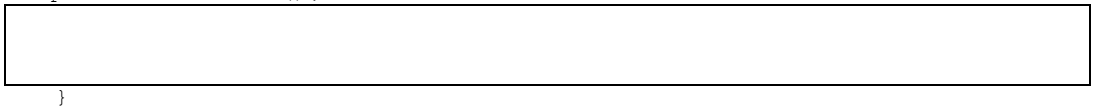
```
    }
    /* Vende determinado produto, devolvendo o valor cobrado
     * @pre
```



```
    */
    public float vendeProd(int codigo, int quant){
```



```
    }
    /* Consulta o valor em caixa
     */
    public float caixa(){
```



```
    }
```

Número: _____ Nome: _____

```
/* Métodos auxiliares */
```

Número: _____ Nome: _____

```
}
```

Número: _____ Nome: _____

Grupo III

Implemente o seguinte método auxiliar, da classe Main, que é usado na criação de objectos da classe *Loja* (4 valores):

```
import java.io.FileReader;

public class Main {

  /* Lê do ficheiro indicado os dados para preencher as colecções de produtos em stock
   * na nova loja minhaLoja.
   * Na primeira linha do ficheiro está o número de produtos vendidos na loja.
   * Cada uma das linhas seguintes contém a informação de dado produto, com o formato
   * 'codigo preco quantidade_em_stock descricao' (assumindo que a quantidade em
   * exibição é sempre de zero unidades).
   * Caso exista erro na abertura do ficheiro, deve escrever na consola "ERRO"
   * @param nomeFich nome do ficheiro a ler
  * @pre nomeFich!=null && minhaLoja == null
  */

  private static void refill (String nomeFich, Loja minhaLoja){
```

Número: _____ Nome: _____

```
   }
```

Grupo IV

Implemente o método *maiorSeqCresc,* que se encontra na classe *SemNome,* de acordo com a descrição em baixo (3 valores).

```
public class SemNome {
  ...
  /* Vector de valores inteiros, que se encontra preenchido
   * desde a posição 0 até contador-1
   */
  private int [] numeros; // vector de valores inteiros
  private int contador;
  ...
   /* Devolve o tamanho da maior sequência crescente de valores inteiros que ocorre
    * no vector numeros. Exemplos:
    * numeros = [4,3,2,1], contador = 4 → devolve 1 (a sequência "4")
    * numeros = [1,2,0,-1], contador = 4 → devolve 2 (a sequência maior é "1,2")
    * numeros = [1,2,0,-1,6,7,5], contador = 5 → devolve 3 (a sequência maior é "-1,6,7")
    * @return tamanho da maior sequência crescente de inteiros

  public int maiorSeqCresc (){
```

```
  }
}
```