2º Teste de "Introdução à Programação" (2012/13 – 1º Semestre)

Duração 2H

Instruções importantes:

- Responda a cada grupo em **folhas separadas**.
- Verifique que **todas** as folhas estão identificadas com o seu nome e número.
- 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 teste termine.

Grupo I

Implemente a classe `Presente` que representa um presente de Natal. Um presente de Natal caracteriza-se por uma *descrição*, pelo *nome de quem o recebe*, pela *quantidade*, e pelo seu *preço unitário*. Por exemplo, podem-se oferecer *2* garrafas de *Vinho do Porto* ao *Toni* ao preço unitário de *€12.60* (ou seja, com um valor total de *€25.20*). Na sua classe, declare as variáveis de instância que entender necessárias e implemente as seguintes operações:

```
/**Cria um novo presente com os argumentos:
 * @param descricao – Descrição do presente
 * @param nome – Nome da pessoa que recebe o presente
 * @param quantidade – Quantidade de unidades do presente
 * @param precoUnitario – Preço de cada uma das unidades do presente
 * @pre quantidade > 0 && precoUnitario >= 0.0f
 */
public Presente(String descricao, String nome, int quantidade,
                float precoUnitario)

/**Devolve a descrição do presente
 * @return – A descrição do presente
 */
public String devolveDescricao()

/**Devolve o nome da pessoa que recebe o presente
 * @return – O nome da pessoa que recebe o presente
 */
public String devolveNome()

/**Devolve o valor total do presente
 * @return – O valor total do presente
 */
public float devolveValorTotal()

/**Acrescenta unidades ao presente.
 * @param n – número de unidades a acrescentar ao presente
 * @pre n > 0
 */
public void adicionaQuantidade(int n)

/**Compara o objecto this com o argumento outro
 * @return true se todos os membros de this forem iguais aos do outro
 */
public boolean equals(Presente outro)
```

Grupo II

Implemente a classe `PaiNatal` que representa uma colecção de objectos do tipo `Presente` oferecidos pelo Pai Natal. O número de prendas que o Pai Natal pode dar não está limitado à partida. Quando o objecto do tipo `PaiNatal` é criado a colecção de presentes está vazia. Os presentes devem ser acrescentados à colecção por ordem de chegada dos pedidos. Essa ordem é importante e deve ser mantida mesmo que algum presente seja cancelado (se a pessoa se portou mal, pode ficar sem os presentes inicialmente previstos ☺). Implemente a classe `PaiNatal`. Declare as constantes e variáveis que considerar necessárias e implemente as operações:

```
/**Cria um novo PaiNatal.
 * @param – maximo – limite máximo para os gastos
 * @pre maximo >= 0.0f
 */
public PaiNatal (float maximo)

/**Tenta dar um presente de Natal. Apenas deve ser dado um presente se o limite de gastos
 * nos presentes não for ultrapassado. Se não conseguir dar o presente, não faz nada.
 * @param descricao – Descrição do presente
 * @param nome – Nome da pessoa que recebe o presente
 * @param quantidade – Quantidade de unidades do presente
 * @param precoUnitario – Preço de cada uma das unidades do presente
 * @pre quantidade > 0 && precoUnitario >= 0.0f
 */
public void oferece(String descricao, String nome, int quantidade, float precoUnitario)

/**Devolve o número de ofertas registadas na colecção de presentes. Note que isto
 * corresponde ao número de elementos da colecção de presentes, e não ao número de
 * unidades oferecidas. Por exemplo, 2 sapatos oferecidos contam como uma oferta.
 * @return – o número de ofertas registadas
 */
public int contaOfertas()

/**Devolve os gastos totais em presentes.
 * @return – o valor total gasto em presentes.
 */
public float valorTotal()

/**Devolve o presente mais caro de todos. Em caso de empate, devolve um qualquer de entre
 * os empatados.
 * @return – o presente mais caro da colecção.
 * @pre – contaOfertas()>0
 */
public Presente maisCaro()

/**Remove todos os presente a dar a uma pessoa da lista, dado o seu nome. Se não
 * existirem presentes para essa pessoa, o método não faz nada. Esta operação deve manter
 * a ordem dos restantes presentes.
 * @param nome – nome da pessoa que fica sem presentes.
 */
public void castiga(String nome)

/**Inicializa o iterador de presentes.
 */
public void inicializaIterador()

/**Testa se há mais elementos a visitar com o iterador.
 * @return true, se houver mais elementos a visitar, ou false, caso contrário.
 */
public boolean existeProximo()

/**Devolve o próximo presente da colecção visitada com o iterador.
 * @return o próximo presente da colecção visitada com o iterador.
 * @pre existeProximo()
 */
public Presente proximo()
```

Informação para os Grupos III e IV

Considere o esqueleto da classe `TestaPaiNatal` que permite experimentar a classe `PaiNatal`.

```
package ip;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

public class TestaPaiNatal {
  /**
   * Lê os dados dos presentes do ficheiro presentes.txt, construindo uma lista de
   * presentes de Natal em que apenas podem ser acrescentados presentes num valor
   * acumulado de até 500 euros.
   * Depois, escreve na consola a lista de presentes que foram registados e tenham um
   * valor maior que 100 euros.
   * @param args – argumentos do programa principal – não usado neste método.
   */
  public static void main(String[] args) {
    try {
      PaiNatal santa = carrega("presentes.txt", 500.0f);
      listaPresentesCaros(santa, 100.0f);
    } catch (FileNotFoundException e) {
      System.out.println("Ficheiro inexistente");
    }
  }

  /**
   * Cria e devolve um objecto do tipo PaiNatal, que preenche com os presentes lidos do
   * ficheiro cujo nome é passado como argumento. O objecto do tipo PaiNatal começa por
   * ter uma colecção vazia de presentes e acrescenta os presentes enquanto não esgotar
   * o limite máximo de custo dos presentes.
   * @param fich – nome do ficheiro que contém informação sobre os presentes.
   * @param maximo – limite máximo do custo dos presentes a carregar na colecção de
   *                 presentes a carregar.
   * @return – Um objecto do tipo PaiNatal, contendo os presentes lidos do ficheiro que
   *           puderam ser oferecidos.
   * @throws – FileNotFoundException – excepção lançada se o ficheiro não estiver
   *           disponível.
   */
  private static PaiNatal carrega(String fich, float maximo) throws FileNotFoundException

  /**
   * Escreve na consola os presentes com valor maior que o limite.
   * @param coleccao – Objecto do tipo PaiNatal, com a colecção de presentes.
   * @param limite – valor a partir do qual um presente é considerado caro.
   */
  private static void listaPresentesCaros(PaiNatal coleccao, float limite)
```

Grupo III

Implemente o método `carrega`, assumindo a seguinte estrutura de ficheiro:

- Na 1ª linha, indicam-se *quantos presentes o ficheiro contém* (um inteiro maior ou igual a 0)
- Para cada presente, apresentam-se 3 linhas:
  - Na primeira, a *descrição* do presente.
  - Na segunda, o *nome* da pessoa que vai receber o presente.
  - Na terceira linha, a *quantidade* (inteiro) e o *preço unitário* do presente (float), separados por um espaço em branco.

Por exemplo, o ficheiro `presentes.txt` tem *4 presentes*, dos quais apenas 3 entram, de facto, na lista. O presente destinado ao *Gabriel Alves* ultrapassaria o limite máximo de *500 euros*, não sendo por isso inserido na lista de presentes que o Pai Natal vai guardar (recorde a descrição do método `oferece`, da classe `PaiNatal`):

Ficheiro `presentes.txt`:

```
4
Consola espectacular
João Silva
1 200.0
Meias cosidas à mão
Doroteia Cinderela
2 1.5
LCD
Gabriel Alves
1 400.49
Perfume Caríssimo
Doroteia Cinderela
2 85.99
```

Grupo IV

Implemente o método `listaPresentesCaros`. O formato para a apresentação de cada um dos presentes é semelhante ao encontrado nos ficheiros de texto mas, neste caso, o resultado é escrito na consola e não é dada indicação sobre quantos presentes são apresentados. No programa principal apresentado na página anterior deste enunciado, usando o ficheiro `presentes.txt`, descrito no **Grupo III**, há dois presentes que ultrapassam o limiar – a *consola espectacular* do *João* e os *perfumes caríssimos* da *Doroteia* (repare que, neste caso, embora o preço unitário de cada perfume seja inferior a €100.00, o valor total do presente é de €171.98, por serem 2 perfumes – é o valor total da prenda que conta). Deste modo, sendo chamada no programa da página anterior este método deveria escrever o seguinte resultado na consola:

```
Consola espectacular
João Silva
200.0
Perfume Caríssimo
Doroteia Cinderela
171.98
```