2º Teste de "Introdução à Programação" (2014/15 – 1º Semestre)

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.
- 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.

Considere o seguinte enunciado do problema que será usado nos grupos I, II e III:

Objectivo:

Gerir os sacos de prendas da árvore de Natal.

Descrição:

A árvore de Natal tem um conjunto de no máximo 30 sacos de prendas. Cada saco de prendas pertence a uma pessoa e contem um conjunto de prendas. Cada saco é identificado pelo nome da pessoa (único, não existem dois sacos com o mesmo nome de pessoa). Cada prenda foi comprada por um dado valor (número real) em Euros e contem uma pequena descrição (indica o que é a prenda).

Interação com o utilizador:

Interpretador de comandos, onde é permitido adicionar um saco de prendas à árvore de Natal ("AS"), adicionar uma nova prenda a um saco da árvore de Natal ("AP"), consultar quantas prendas tem uma dada pessoa/saco ("QS"), escrever o nome da pessoa com o saco de prendas de maior valor em Euros em prendas ("NSM"), escrever o nome da pessoa com a prenda de maior valor em Euros ("NPM"), gravar no ficheiro "pessoasComPrendas.txt" todos os nomes das pessoas que tem prendas na árvore de Natal ("LP") e sair da aplicação ("S").

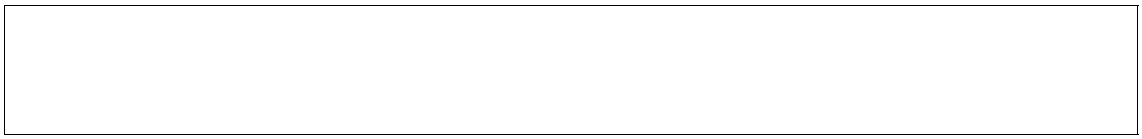
Para a resolução dos grupos I, II e III, foram definidas 3 classes: Prenda, SacoPrendas e ArvoreNatal. As classes Prenda e ArvoreNatal serão implementadas neste teste, enquanto que a classe SacoPrendas já está implementada, unicamente será usada.

Grupo I

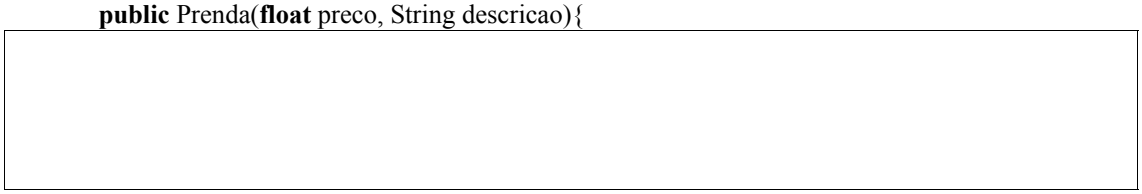
Este exercício visa a construção de uma classe `Prenda` que representa uma prenda, que custou uma dado valor em Euros e caracteriza-se por uma dada descrição

Complete a seguinte implementação da classe `Prenda` **(6 valores)**:

```
public class Prenda {
        /* Variáveis de instância */
```

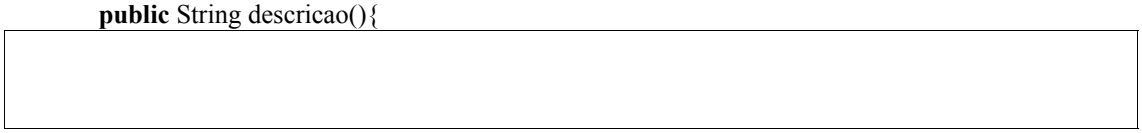


```
        /* Constructor da classe Prenda
         * @param preço valor em Euros da prenda
         * @ param descricao descrição do conteudo da prenda
         */
        public Prenda(float preco, String descricao){
```

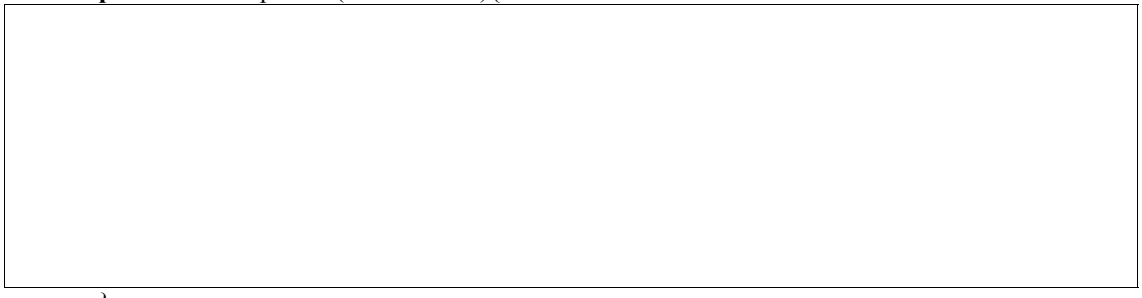


```
        }
        /** Consulta o preço da prenda
         * @return – o valor do preco da prenda
         */
        public float preco(){
```

```
        }
        /** Consulta a descrição da prenda
         * @return – a descrição do conteudo da prenda
         */
        public String descricao(){
```



```
        }
        /* Compara duas prendas com base no seu valor (this e outra)
         * @return 0 se this e outra tem preços iguais;
         *         valor menor que 0, se this tem um preço menor do que a outra;
         *         valor maior que 0 se this tem um preço maior do que a outra.
         * @pre outra != null
         */
        public int compareTo(Prenda outra){
```



```
        }
}
```

Grupo II

A classe *SacoPrendas* representa um saco de prendas de uma dada pessoa. Cada objecto desta classe (um saco de prendas) caracteriza-se pelo identificador da pessoa (nome) e o conjunto de prendas (em média existem 25 prendas por cada saco). Esta classe está implementada e contem os seguintes métodos públicos.

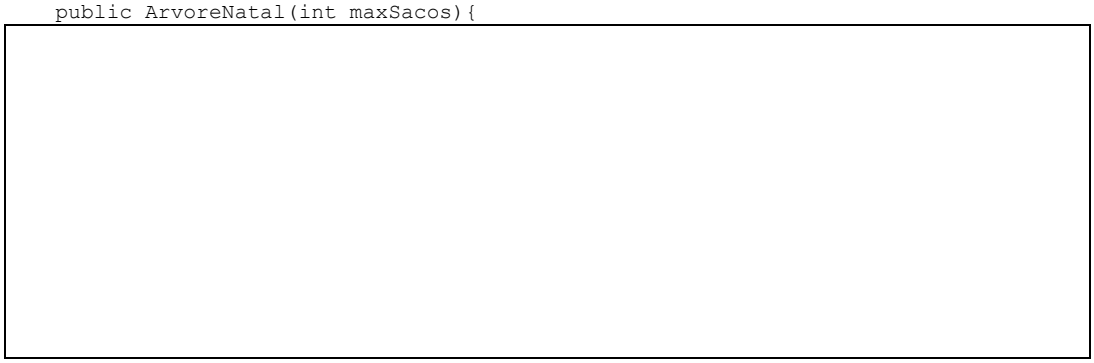
```
/**Constructor: cria um saco sem prendas
 * @param nome – nome da pessoa
 */
public SacoPrendas(String nome){...}

/* Consulta o nome da pessoa a que pertence o saco
 * @return – nome da pessoa
 */
public String nome(){...}

/* Consulta o número de prendas do saco
 * @return – número de prendas
 */
public int numeroPrendas(){...}

/* Consulta o valor em Euros das prendas no saco
 * @return – valor em Euros de todas as prendas do saco
 */
public float euros(){...}

/* Adiciona uma nova prenda ao saco
 * @param valor – valor da prenda
 * @param desc – descrição do conteudo da prenda
 */
public void adicionaPrenda(float valor,String desc){...}

/* Retorna a prenda mais cara do saco.
 * @return a prenda mais cara
 * @pre numeroPrendas() >0
 */
public Prenda maisCara(){...}
```

Este exercício visa a construção de uma classe *ArvoreNatal* que gere os sacos de prendas. Cada objecto desta classe caracteriza-se por um conjunto de sacos de prendas(objectos da classe SacoPrendas) e o valor total gasto em todas as prendas. Note que no momento de criação da árvore é definido o número máximo de sacos de prendas existentes na árvore.

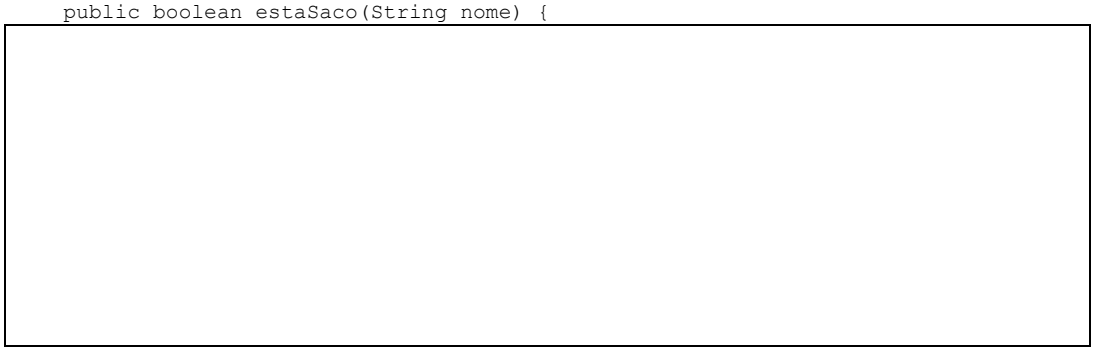
Complete a seguinte implementação da classe *ArvoreNatal* **(9valores)**:

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

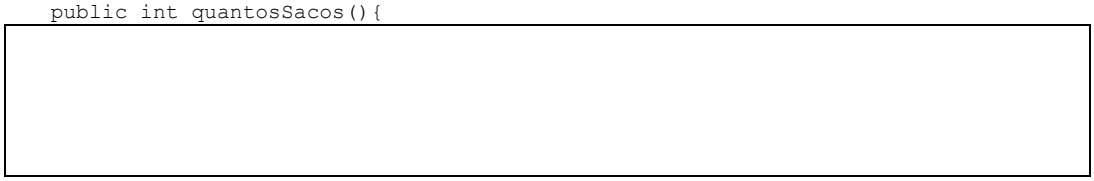
```
/**
 * Constructor - Conjunto de sacos de prendas vazio e valor em caixa 0 Euros
 * @param maxSacos - número máximo de sacos de prendas
 */
public ArvoreNatal(int maxSacos){
```



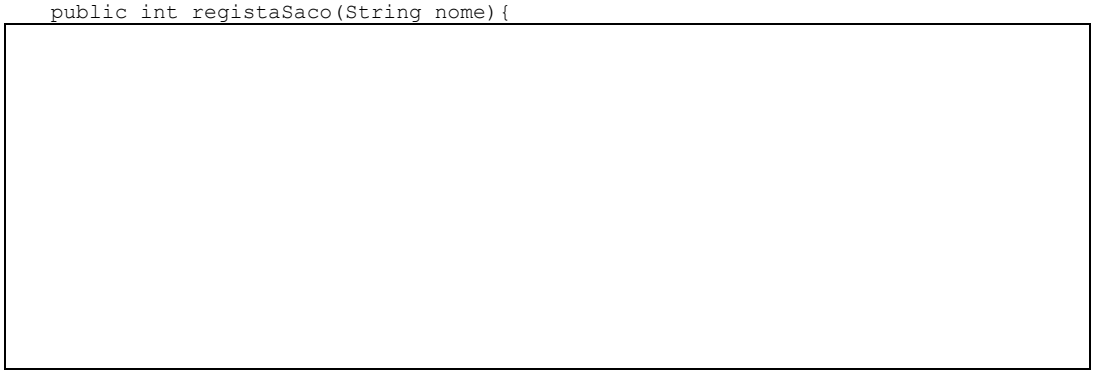
```
}
/* Indica se existe um saco para um dado nome de pessoa.
 * @param nome - nome de uma pessoa
 * @return - true, caso exista; false, caso contrário
 */
 public boolean estaSaco(String nome) {
```



```
 }
/* Consulta o número de sacos de prendas existentes
 * @return - número de sacos existentes
 */
public int quantosSacos(){
```

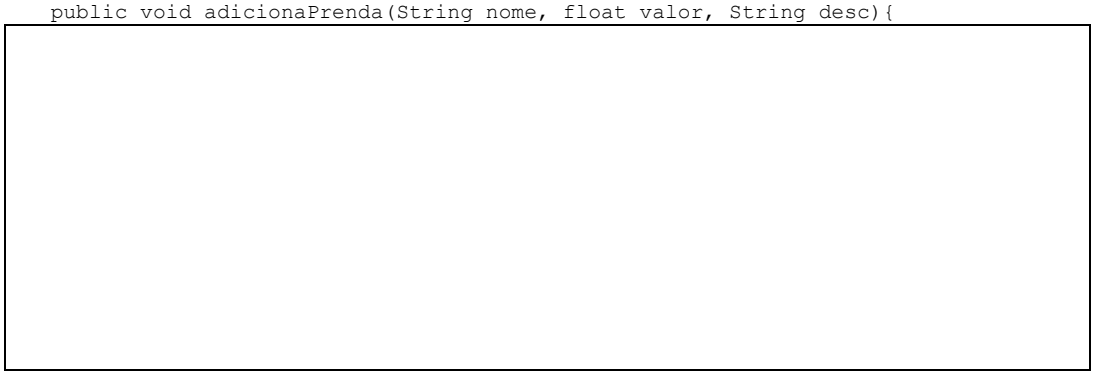


```
 }
/* Regista um novo saco de prendas. Note que o registo do saco só é efectuado
 * se ainda existe espaço na arvore (maxSacos dado no constructor).
 * @param nome - nome da pessoa
 * @return 1 caso tenha sucesso; 0 caso contrário
 */
public int registaSaco(String nome){
```

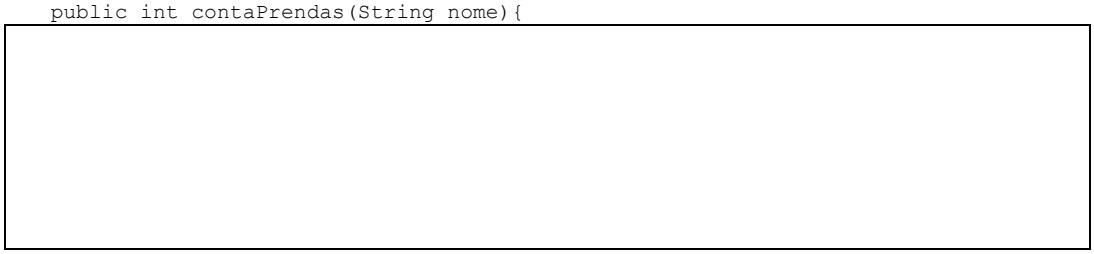


}

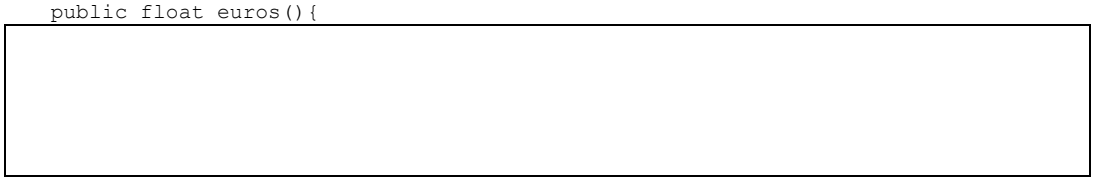
```
/* Adiciona a prenda ao saco pertencente à pessoa com o nome dado
 * @param nome  nome da pessoa
 * @param valor preço em Euros da prenda
 * @param desc descrição do conteudo da prenda
* @pre estaSaco(nome)
 */
public void adicionaPrenda(String nome, float valor, String desc){
```



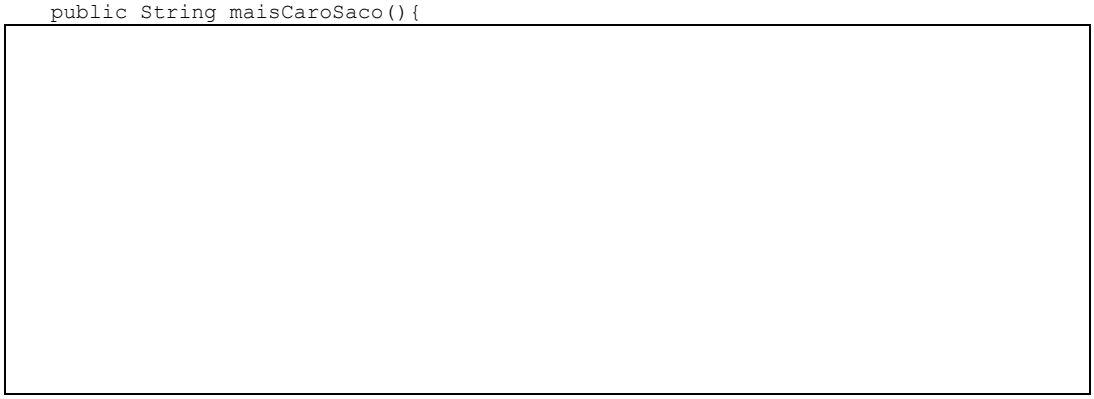
```
}
/* Consulta o número de prendas da pessoa com o nome dado
 * @param nome  nome da pessoa
 * @return número de prendas
* @pre estaSaco(nome)
 */
public int contaPrendas(String nome){
```



```
}
 /* Consulta o valor em Euros gasto nas prendas todas
  * @return - valor em Euros
 */
public float euros(){
```



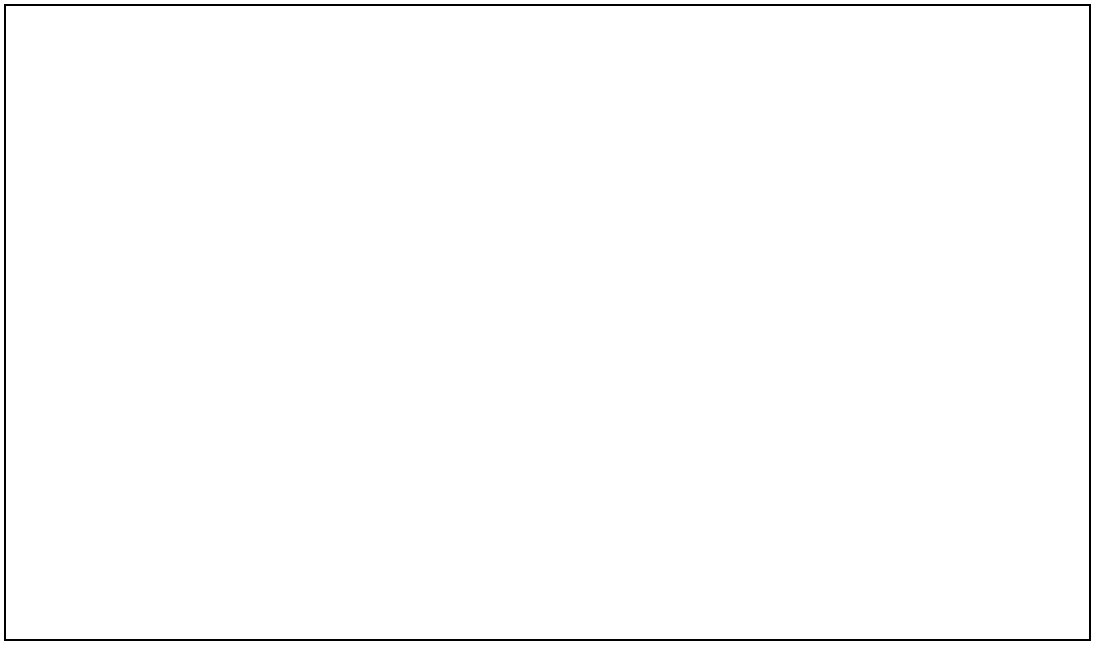
```
}

 /* Consulta o nome da pessoa com o saco de prendas mais caro
  * @pre quantosSacos() >0
  * @return nome da pessoa
 */
public String maisCaroSaco(){
```



```
    }
```

```
    /* Consulta o nome da pessoa com a prenda mais cara
     * @pre quantosSacos() >0
     * @return nome da pessoa
     */
    public String maisCaroPrenda(){
```

```
    }

    /* Métodos para o iterador  de sacos na árvore Natal
     * Assuma que estão implementados e funcionam como explicado nas aulas
     */

    /* Inicializa o iterador de sacos
     */
    public void iniciaIterador(){…}
    /* Verifica se existem mais sacos para iterar
     * @return – true, se existir pelo menos mais um saco;
                 false, caso contrário
     */
    public boolean temSeguinte(){…}
    /* Retorna o seguinte saco na iteração
     * @return um saco
     * @pre temSeguinte()
     */
    public SacoPrendas seguinte(){…}

    /* Métodos auxiliares */
```



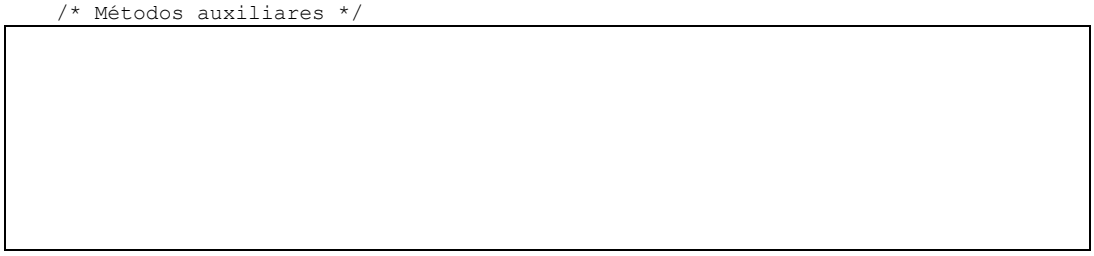
```
}
```

Grupo III

Implemente o seguinte método auxiliar que existe na classe Main, onde é testada a classe *ArvoreNatal* (**2 valores**):

```
import java.io.PrintWriter;
…
public class Main {
  …
   /* Escreve no ficheiro dado, os nomes das pessoas com sacos que contem prendas na
  arvore dada.
    * Em cada linha do ficheiro "pessoasComPrendas.txt" deve ser escrito o nome da pessoa
   */

  private static void pessoasComPrendas (ArvoreNatal arv){
```

```
        }

        …
         /* Métodos auxiliares */
```



```
}
```

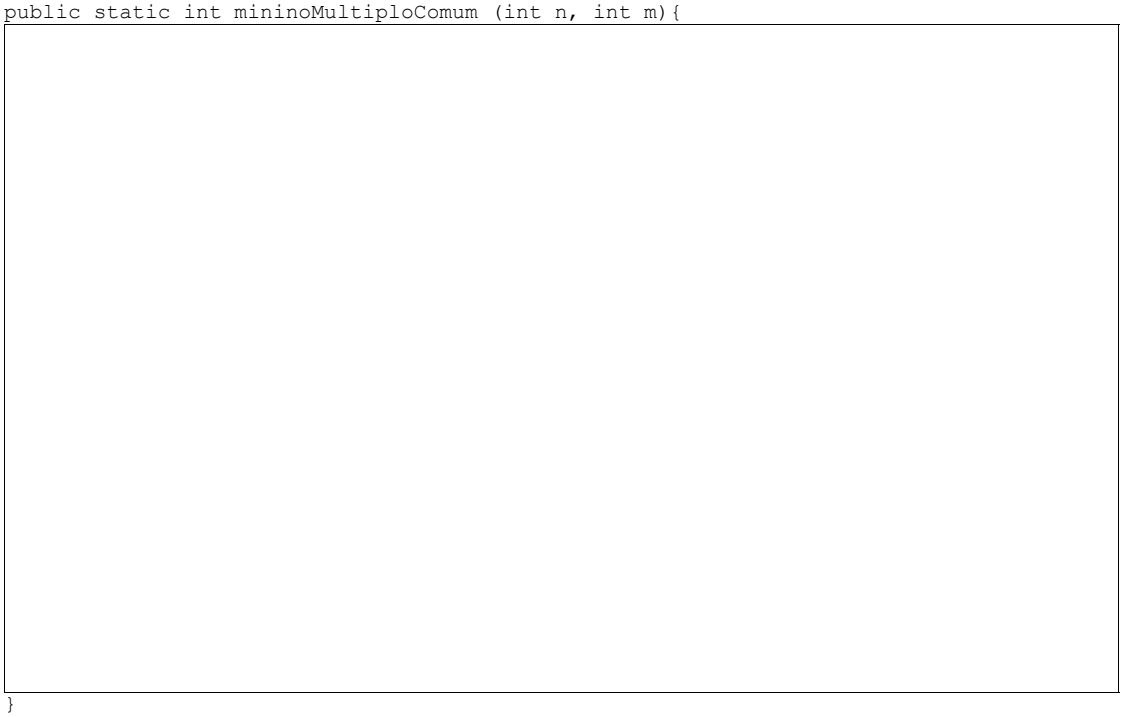
Grupo IV

Este exercício visa a implementação do método estático *minimoMultiploComum* que se encontra na biblioteca *Matematica*. (**3 valores**)

Sugestão: um algoritmo de "força bruta" é multiplicar os dois números e ver depois se algum valor inferior a esse mas superior ou igual ao maior dos dois valores dados é também múltiplo.

```
public class Matematica {
  ...
    /* Devolve o mínimo múltiplo comum entre dois naturais.
     * Exemplos:
     * se n = 2 e m = 4 entao devolve 4
     * se n = 3 e m = 5 entao devolve 15
     * se n = 4 e m = 6 entao devolve 12
     * @pre
```

```
     */


  public static int mininoMultiploComum (int n, int m){
```



```
  }
  ...
}
```