2º Teste de "Introdução à Programação" (2015/16 – 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**.
- **Não pode** usar 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:

Agendar o jantar de Natal.

Descrição:

O jantar de Natal terá que ser realizado num dia útil (segunda-feira, terça-feira, quarta-feira, quinta-feira ou sexta-feira) da semana antes do Natal. A esse jantar podem ir todos os funcionários que se inscreverem e indicarem a sua disponibilidade para o dia escolhido. Cada funcionário pode indicar as suas disponibilidades e o dia escolhido para o jantar será aquele em que haja maior número de funcionários com disponibilidade.

Interação com o utilizador:

Interpretador de comandos, onde é permitido adicionar uma disponibilidade de um funcionário ("ADF"), remover a disponibilidade de um dado funcionário ("RDF"), consultar quantos funcionários estão disponíveis para um dado dia ("QFD"), gravar num ficheiro os nomes dos funcionários disponíveis para o dia com maior número de participantes ("FJN") e sair da aplicação ("S").

Para a resolução dos grupos I, II e III, foram definidas 3 classes: AgendaFuncionario, JantarNatal e Main. As classes AgendaFuncionario e JantarNatal serão implementadas neste teste, enquanto a classe Main já está parcialmente implementada, unicamente terá que a completar.

Caderno 1

Grupo I

Este exercício visa a construção de uma classe *AgendaFuncionario* que representa um funcionário e as suas disponibilidades. Complete a seguinte implementação da classe *AgendaFuncionario* **(7,5 valores)**:

```
public class AgendaFuncionario {

    /* Constantes e Variáveis de instância */









    /* Cria um funcionário com o nome dado e com todos os dias úteis indisponíveis

     * @param nome nome do Funcionario

     * @pre nome != null
     * /

    public AgendaFuncionario(String nome){









}
```

```
/** Consulta do nome do funcionário
 * @return – nome do funcionário
 */
public String nome(){




}
/** Regista que no dia dado, o funcionário está disponível
 * @param dia 2 (segunda), 3 (terça), 4 (quarta), 5 (quinta), 6 (sexta)
 * @pre  dia >= 2 && dia <= 6
 */
public void registaDisponivel(int dia){







}
/** Regista que no dia dado, o funcionário está indisponível
 * @param dia 2 (segunda), 3 (terça), 4 (quarta), 5 (quinta), 6 (sexta)
 * @pre– dia >= 2 && dia <= 6
 */
public void registaInDisponivel(int dia){
```

```
}
/** Consulta o número de dias disponíveis do funcionário
 * @return – número de dias disponíveis
 */
public int  disponiveis(){









}
/** Consulta se num dado dia o funcionário está disponível
 * @param dia 2 (segunda), 3 (terça), 4 (quarta), 5 (quinta), 6 (sexta)
 * @return true se está disponível, false caso contrário
 * @pre– dia >= 2 && dia <= 6
 */
public boolean estaDisponivel(int dia){









}
```

}

Caderno 2

Grupo II

Este exercício visa a construção de uma classe *JantarNatal* que representa o agendamento para o jantar de Natal. Cada objecto desta classe serve para registar as disponibilidades dos funcionários para o jantar de Natal de modo a poder escolher o dia com maior número de funcionários disponíveis. Complete a seguinte implementação da classe *JantarNatal* **(7,5 valores)**:

```
public class JantarNatal{

    /** Constantes e Variáveis de instância --- JA DADAS**/

    public static final int MAX=5;


    /* vector acompanhado de funcionários */

    private AgendaFuncionario [] funcionarios;

    private int numFuncionarios;


    /* vector de 5 posições que indica o número de funcionários disponíveis por dia da semana. Por exemplo, se na segunda e sexta existem 2 funcionários disponíveis em cada dia, o vector disponiveisSemana na sua posição 0 e 4 contem 2*/

    private int [] disponiveisSemana;

    /* Métodos auxiliares - privados */
```

```
    /**
     * Constructor – Criação dum agendamento para o jantar de Natal, com o número
     * de funcionários disponíveis a zero para todos os dias úteis da semana,
     *  e sem funcionários registados.
     * @param maxFuncionarios – número máximo de funcionários que se podem registar
    * @pre maxFuncionarios > 0
     */
  public JantarNatal(int maxFuncionarios){









    }
    /* Indica se existe um funcionário com o nome dado.
     * @param nome – nome de um funcionário
     * @return - true, caso exista; false, caso contrário
     * @pre nome != null
      * /
     public boolean existeFuncionario(String nome) {









    }
/* Regista a disponibilidade dum funcionário num dado dia.
```

```
 * Note que: caso o funcionário não existir esse é adicionado ao vetor, se ainda existir
espaço
 * (maxFuncionarios dado no constructor). Caso exista ou após a sua inserção, deve
 * registar-se a sua disponibilidade no dia indicado. Note que, caso o funcionário já esteja
 *  disponível nesse dia, este método não fará nada.

 * @param nome – nome do funcionario

 * @param dia – 2 – segunda; 3- terça; 4- quarta; 5- quinta; 6- sexta
 * @return false caso não seja possível inserir um novo funcionário;
 *                 true em qualquer outro caso

 * @pre dia >= 2 && dia <= 6 && nome != null
 */
public boolean registaDisponibilidade(String nome, int dia){









}

/* Regista a indisponibilidade dum funcionário num dado dia, caso o funcionário existir.
```

```
 * . Note que, caso o funcionário já esteja indisponível nesse dia, este método não fará nada.

 * @param nome – nome do funcionario

 * @param dia – 2 – segunda; 3- terça; 4- quarta; 5- quinta; 6- sexta
 * @pre dia >= 2 && dia <= 6 && nome!= null && existeFuncionario(nome)
 */
public void registaIndisponibilidade(String nome, int dia){









}


/* Retorna o número de participantes/funcionários disponíveis para o dia indicado
```

```
 * @param dia – 2 – segunda; 3- terça; 4- quarta; 5- quinta; 6- sexta
 * @return número de participantes disponíveis

 * @pre dia >= 2 && dia <= 6

 */
public int numeroDisponiveis(int dia){









}

/* Retorna o dia com o maior número de participantes disponíveis.
 * Note que: em caso de empate, retorna o primeiro dia da semana e,
 * caso ainda ninguém registou disponibilidade, ou não existem participantes retorna 0.

 * @return  2 – segunda; 3- terça; 4- quarta; 5- quinta; 6- sexta ; o não existe maior dia

 */
public int melhorDia(){









}
```

/* Métodos para o **iterador** de funcionários

* Assuma que estão implementados e funcionam como explicado nas aulas

*/

/* Inicializa o iterador de funcionários

*/

public void iniciaIterador(){...}

/* Verifica se existem mais funcionários para iterar

* @**return** – true, se existir pelo menos mais um funcionário; false, caso contrário

*/

public boolean temSeguinte(){...}

/* Retorna o seguinte funcionário na iteração

* @return um funcionário

* @pre temSeguinte()

*/

public AgendaFuncionario seguinte(){...}

}

Caderno 3

Grupo III.

Implemente o seguinte método auxiliar que existe na classe Main, onde é testada a classe `JantarNatal` (**2,5 valores**):

```
import java.io.PrintWriter;

import java.io.FileNotFoundException;

public class Main {

  /* Escreve no ficheiro "pessoasParaJantar.txt", os nomes dos funcionários que tem
    * disponibilidade  no dia com maior número de pessoas disponíveis (método
     * melhorDia da classe JantarNatal). Caso este método retorne 0, o ficheiro fica vazio.
     * Em cada linha do ficheiro deve ser escrito o nome do funcionário disponível no dia
   * indicado (melhorDia).*/
  private static void pessoasJantar (JantarNatal natal) throws FileNotFoundException{









}
```

Caderno 4

Grupo IV.

Considere um vector de inteiros positivos, chamado *vectNatPos*, que se assume preenchido. Implemente o método peneira, sem argumentos, que coloca a zero valores não primos do vector. Considere que o vector de inteiros *primeiros30Primos* contém os primeiros 30 números primos.

Assuma que *vectNatPos* e *primeiros30Primos* são variáveis de instância da classe onde vai implementar o método peneira. (**2,5 valores**)

Sugestão: Percorra o vector *primeiros30Primos*; divida todos os elementos do vector *vecNatPos* por cada um dos elementos *de primeiros30Primos*.

public void peneira(){

}