PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA

Disciplina: Fundamentos de Programação **Turma**: 147 **Data**: 27/11/2015 **Nome**: _____	G1: ☐ P1 ☐ P2 X P3 ☐ PS ☐ G2 **Nota**: _____ **Rubrica**: _____ **Profa. Luana Müller**

Instruções: Leia calmamente toda a prova. Somente após, comece a resolver as questões. A interpretação das questões faz parte da prova. As soluções deverão ser escritas em Java.

1. Na série de televisão *Game Of Thrones*, existe uma grande necessidade de figurantes para dar vida a série. Alguns episódios requerem, inclusive, centenas de figurantes. O problema é que, devido a demanda, as vezes é necessário contratar o mesmo figurante para participar de episódios diferentes e alguns espectadores tem notados que alguns destes, tem "voltado a vida", pois já haviam sido mortos na série.
Então, para resolver o problema, a produção da série está fazendo um programa que os ajude no gerenciamento dos figurantes para evitar que esse problema aconteça.
O programa é composto pelas classes apresentadas abaixo:

```
public class Pessoa {
	private String nome;
	private String email;
	private char sexo;
	private int idade;
	private String telefone;
	private String cidade;

	//construtor
	//getters e setters
}
```

```
public class Figuracao {
	private Pessoa figurante;
	private int temporada;
	private int episodio;

	//construtor
	//getters
	//setters
}
```

```
public class CadastroDePessoas {
	private Pessoa [] cadastroPessoas;
	private int proximaPosicao;
	public CadastroDePessoas(){
		cadastroPessoas = new Pessoa[1000];
		proximaPosicao = 0;
	}
	public boolean adicionar(Pessoa p){
		if(proximaPosicao >= cadastroPessoas.length){
			return false;
		} else {
			cadastroPessoas[proximaPosicao] = p;
			proximaPosicao++;
			return true;
		}
	}
	public boolean remover (String email){
		boolean troca = false;
		for(int i=0; i<proximaPosicao; i++){
			if(cadastroPessoas[i].getEmail().equalsIgnoreCase(email)){
				troca = true;
			}
			if(troca){
				cadastroPessoas[i] = cadastroPessoas[i+1];
			}
		}
		if(!troca){
			return false;
		}
		cadastroPessoas[proximaPosicao-1] = null;
		proximaPosicao--;
		return true;
	}}
```

```java
public class CadastroDeFiguracao {

	private Figuracao [] cadastro;
	private int proximaPosicao;

	public CadastroDeFiguracao(){
		cadastro = new Figuracao[10000];
		proximaPosicao = 0;
	}

}
```

É possível observar que o programa está incompleto, e precisamos de sua ajuda para finalizar o mesmo.

Nesta última classe, CadastroDeFiguracao, é onde as informações serão manipuladas afim de evitar-se o problema citado e, também, se gerar alguns relatórios que pode ajudar a produção na hora de escalar os elencos.

Desta forma, para esta classe:

1. (3.0 pts) Crie o método `public boolean adicionar(Figuracao f)` de forma que a figuração só seja adicionada ao vetor caso o figurante escalado não tenha participado da temporada a qual ele está sendo atribuído e tenha participado de menos que 4 episódios no total da série.
2. (3.0 pts) Escreva um método que retorne todas as figurações feitas na primeira temporada da série, cujos figurantes possuam determinada idade e sexo (recebidos por parâmetro).
3. (1.0 pts) Abaixo podemos observar o algoritmo de ordenação BubbleSort. Neste algoritmo, lista, corresponde a um vetor de inteiro. Adapte-o de forma que ele ordene a lista de figurações em ordem crescente pelo número da temporada.

```java
for(int i=0; i<lista.size()-1; i++){
	for(int j=0; j<lista.size()-1-i; j++){
		if(lista.get(j) > lista.get(j+1)){
			int aux = lista.get(j);
			lista.set(j, lista.get(j+1));
			lista.set(j+1, aux);
		}
	}
}
```

4. (3.0 pts) Em várias cenas da série são criados (literalmente) exércitos de figurantes, com diversas fileiras. Escreva um método que receba por parâmetro uma matriz de Figurantes (que representa este exército) e retorne o número da linha que contém a maior média de idades.

BOA PROVA E... SUMMER IS COMING!