Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
AD1 de Programação III
1° semestre de 2014

Nome:
Pólo:

Exercício (ENTREGAR OS ARQUIVOS EM MÍDIA, PARA FINS DE TESTE, JUNTAMENTE COM A AD IMPRESSA):
Redes sociais online conectam pessoal de todo o mundo. São poucos os que utilizam computador ou outro dispositivo de comunicação e que não pertençam a pelo menos uma rede social. Inclusive, o mais comum é a mesma pessoa estar cadastrada em diversas redes ao mesmo tempo e, com isso, surge a necessidade de organizar contatos redundantes vindos de origens diferentes.

Nesta questão você deverá criar classes Java que represente contatos em uma dada rede social, e uma classe que agrupa contatos redundantes vindos de diferentes origens. Para isso, assuma que estejam disponíveis as classes `RedeKorkute` e `RedeFakebook`, que mantém listas de usuários cadastrados, respectivamente, nas redes sociais Korkute e Fakebook.

De todos os métodos disponíveis na classe `RedeKorkute`, os únicos relevantes são:

```
int quantidadeDeUsuarios(),
```

que retorna a quantidade de usuários cadastrados na rede social Korkute; e o método

```
UsuarioKorkute obterUsuario(int indice),
```

que retorna o usuário identificado pelo argumento `indice`. Dentre os métodos disponíveis na classe `RedeFakebook`, os únicos relevantes são:

```
int quantidadeDePerfis()
```

e

```
PefilFakebook obterPerfil(int ind),
```

que possuem comportamento análogo aos métodos definidos para `RedeKorkute`. Você deverá:

a) Definir a interface `IUsuarioRedeSocial`, que propõe os método

```
String obterNome()
```

e

```
String obterRedeSocialDeOrigem(),
```

que retornam, respectivamente, o nome do usuário e o nome da rede social onde o usuário está cadastrado.

b) Escrever a classe `UsuarioKorkute` que implementa a interface `IUsuarioRedeSocial`. Esta classe deve armazena o nome (`String`), apelido (`String`) e idade (`int`) do usuário Korkute e fornecer métodos para escrita e leitura desses dados.

c) Escrever a classe `PerfilFakebook` que implementa a interface `IUsuarioRedeSocial`. Esta classe armazena o nome (`String`), endereço (String) e opção de lazer (`String`) do usuário da rede Fakebook e fornece métodos para escrita e leitura desses dados.

d) Escrever a classe `OrganizadoraDeContatos` que implementa o método estático

```
IUsuarioRedeSocial[] agrupar(final String nome, final
  RedeKorkute korkute, final RedeFakebook fakebook).
```

Este método recebe o nome de um usuário que pode ou não estar cadastrado nas redes sociais Korkute e Fakebook, e retorna um array contendo o registro deste usuário nas redes sociais.

Seu programa deve funcionar com a seguinte classe de teste:

```
public class AD1_2014_1 {
    public static void main(String[] args) {
        RedeFakebook fakebook = new RedeFakebook();
        RedeKorkute  korkute  = new RedeKorkute();

        UsuarioKorkute u1 = new UsuarioKorkute();
        u1.atribuirNome("Carlos Bazilio");
        u1.atribuirApelido("Bazilio");
        u1.atribuirIdade(38);
        korkute.adicionaUsuarioKorkut(u1);

        UsuarioKorkute u2 = new UsuarioKorkute();
        u2.atribuirNome("Isabel Rosseti");
        u2.atribuirApelido("Rosseti");
        u2.atribuirIdade(39);
        korkute.adicionaUsuarioKorkut(u2);

        PerfilFakebook p1 = new PerfilFakebook();
        p1.atribuirNome("Isabel Rosseti");
        p1.atribuirEndereco("Passo da Patria, 156");
        p1.atribuirLazer("Assistir desfile");
        fakebook.adicionaPerfilFakebook(p1);

        PerfilFakebook p2 = new PerfilFakebook();
        p2.atribuirNome("Carlos Bazilio");
        p2.atribuirEndereco("Rua Recife, s/n");
        p2.atribuirLazer("Ouvir samba");
        fakebook.adicionaPerfilFakebook(p2);

        IUsuarioRedeSocial [] isabel =
OrganizadorDeContatos.agrupar("Isabel Rosseti", korkute,
fakebook);

        IUsuarioRedeSocial [] bazilio =
OrganizadorDeContatos.agrupar("Carlos Bazilio", korkute,
fakebook);
```

```
        System.out.println("Dados da Isabel:");
        for (int i = 0; i < isabel.length; i++) {
            if(isabel[i] != null)
                System.out.println( "Contato " + i + ":\n" +
isabel[i].toString() );
            System.out.println("");
        }

        System.out.println("Dados do Bazilio:");
        for (int i = 0; i < bazilio.length; i++) {
            if(bazilio[i] != null)
                System.out.println( "Contato " + i + ":\n" +
bazilio[i].toString() );
            System.out.println("");
        }
    }
}
```

RESPOSTA:

```
interface IUsuarioRedeSocial {
    public String obterNome();
    public String obterRedeSocialDeOrigem();
}

class PerfilFakebook implements IUsuarioRedeSocial{
    String nome;
    String endereco;
    String lazer;

    public String obterEndereco() {
        return endereco;
    }

    public void atribuirEndereco(String endereco) {
        this.endereco = endereco;
    }

    public String obterLazer() {
        return lazer;
    }

    public void atribuirLazer(String lazer) {
        this.lazer = lazer;
    }

    public String obterNome() {
        return nome;
    }

    public void atribuirNome(String nome) {
        this.nome = nome;
    }
```

```
    public String obterRedeSocialDeOrigem() {
        return "RedeFakebook";
    }

    public String toString() {
        String saida = "";
        saida = obterNome() + "\n" + obterEndereco() + "\n" +
obterLazer();

        return saida;
    }
}


class UsuarioKorkute implements IUsuarioRedeSocial{
    private String nome;
    private String apelido;
    private int idade;

    public String obterApelido() {
        return apelido;
    }

    public void atribuirApelido(String apelido) {
        this.apelido = apelido;
    }

    public int obterIdade() {
        return idade;
    }

    public void atribuirIdade(int idade) {
        this.idade = idade;
    }

    public String obterNome() {
        return nome;
    }

    public void atribuirNome(String nome) {
        this.nome = nome;
    }

    public String obterRedeSocialDeOrigem() {
        return "RedeKorkute";
    }
```

```
    public String toString() {
        String saida = "";
        saida = obterNome() + "\n" + obterApelido() + "\n" +
obterIdade();

        return saida;
    }
}


class RedeFakebook {
    private PerfilFakebook [] perfis = new PerfilFakebook[10];
    private int quantidadeDePerfis = 0;

    public int quantidadeDePerfis(){
        return quantidadeDePerfis;
    }

    public PerfilFakebook obterPerfil(int indice){
        if(indice >= quantidadeDePerfis()) return null;
        return perfis[indice];
    }

    void adicionaPerfilFakebook(PerfilFakebook p){
        if( quantidadeDePerfis() >= 10 ) return;
        perfis[quantidadeDePerfis++] = p;
    }
}


class RedeKorkute {
    private UsuarioKorkute [] usuarios = new UsuarioKorkute[10];
    private int quantidadeDeUsuarios = 0;

    public int quantidadeDeUsuarios(){
        return quantidadeDeUsuarios;
    }

    public UsuarioKorkute obterUsuario(int indice){
        if(indice >= quantidadeDeUsuarios()) return null;
        return usuarios[indice];
    }

    void adicionaUsuarioKorkut(UsuarioKorkute u){
        if( quantidadeDeUsuarios() >= 10 ) return;
        usuarios[quantidadeDeUsuarios++] = u;
    }
}
```

```
class OrganizadorDeContatos {
    static IUsuarioRedeSocial[] agrupar(final String nome, final
RedeKorkute korkute, final RedeFakebook fakebook){
       IUsuarioRedeSocial[] contatos = new IUsuarioRedeSocial[2];

       for (int i = 0; i < korkute.quantidadeDeUsuarios(); i++) {
            UsuarioKorkute u = korkute.obterUsuario(i);
            if( nome.equals(u.obterNome()) ) {
                contatos[0] = u;
                break;
            }
        }

       for (int i = 0; i < fakebook.quantidadeDePerfis(); i++) {
            PerfilFakebook p = fakebook.obterPerfil(i);
            if( nome.equals(p.obterNome()) ) {
                contatos[1] = p;
                break;
            }
       }

       return contatos;
    }
}
```