INSTITUTO DE COMPUTAÇÃO

Universidade Federal do Rio de Janeiro
Centro de Ciências Matemáticas e da Natureza

Terceira lista de exercícios de Programação Orientada a Objetos

Questões teóricas

1. O que é encapsulamento e qual a sua importância para POO?
2. Investigue: Qual a diferença entre o modificador de visibilidade *public* e a visibilidade padrão (*default*) sobre um atributo de classe? OBS: Lembre-se que ***atributo de classe*** (isto é, um atributo estático) e ***atributo de objeto*** são coisas distintas.
3. O que são pacotes em Java?
4. Qual o propósito da palavra-chave ***this***? Exemplifique.

Questões práticas

5. Considere as duas classes abaixo:

Agenda
- meusContatos: Array<Contato> - total : int = 0 - MAX : int = 100
+listarContatos() : void +ehDuplicado() : boolean +adicionarContato(nome : Contato) : boolean +removerContato(c : Contato) : boolean

Contato
- nome : String - telefone : String
+getNome() : String +setNome(nome : String) +getTel() : String +setTel(telefone : String) +equals(o : Object) : boolean

Ambas as classes foram parcialmente implementadas no código abaixo. **Copie todo o código para a sua IDE de preferência e faça as atividades propostas em seguida.**

Classe Agenda:

```
public class Agenda {
    private Contato[] meusContatos;
    private final int MAX = 100;
    private int total = 0;

    //Construtor:
    public Agenda(){
        meusContatos = new Contato[MAX];
    }

    public boolean ehDuplicado(Contato c) {
        for (int i = 0; i < total; ++i) {
            if (c.equals(meusContatos[i])) { // Chama o nosso "equals()"
                return true;
            }
        }
        return false;
    }

    public void listarContatos(){
        for (int i = 0; i < total; ++i) {
            System.out.println(meusContatos[i].getNome());
        }
        System.out.println("Total de " + total + " contatos listados.");
    }

    public boolean adicionarContato(Contato c){
        if(total == MAX) //Agenda lotada?
            return false;
        meusContatos[total] = c;
        ++total;
        return true;
    }

    public boolean removerContato(){
        return false; //IMPLEMENTAR!
    }
}
```

Classe Contato:

```
public class Contato {
    private String nome;
    private String tel;

    //Método construtor:
    public Contato(String nome, String tel){
        //Note que o 'this' é necessário aqui!
        this.nome = nome;
        this.tel = tel;
    }

    @Override
    public boolean equals(Object obj){ //Criaremos nossa definição de "igualdade"
        if(obj == this) // comparação de referência!
            return false; //Isso mesmo, falso! Entende o motivo?
        if(!(obj instanceof Contato)) //"Se obj *NÃO* é da classe Contato, então...
            return false; //...obviamente não é uma duplicata.

        Contato c = (Contato)obj; //Convertemos o objeto genérico obj para o tipo Contato.

        //A classe String tb herda de Object, daí tb tem seu próprio "equals()"!
        return  this.nome.equals(c.getNome());
    }
    public String getNome(){
        return nome; //O mesmo que "return this.nome"
    }
    public void setNome(String nome){
        this.nome = nome;
    }
}
```

Também precisaremos da *driver class*, isto é, a classe (com um nome qualquer de sua escolha) que conterá o método principal ***public static void main(String args[]),*** dentro do qual criaremos nossa agenda de contatos e realizaremos operações básicas. **Para sua conveniência, já é fornecida a *driver class* abaixo.** Basta copiá-la e daí modificá-la quando necessário.

IMPORTANTE: O nome da classe abaixo (no caso, “Teste”) deve ser o mesmo do arquivo *.java criado para ela (no caso, Teste.java). Renomeie a classe ou o arquivo conforme sua conveniência.

```
import java.util.Scanner;

//Driver class - ponto de entrada da execução do código.
public class Teste{
    public static void main(String[] args) {
        Agenda ag = new Agenda();

        Contato c1 = new Contato("João", "+55 21 99999-9999");
        Contato c2 = new Contato("João", "+55 21 92222-2222");
        Contato c3 = new Contato("Maria", "+55 21 98888-8888");
        Contato c4 = new Contato("José", "+55 21 97777-7777");
        Contato c5 = new Contato("", "+55 21 91111-1111");


        ag.adicionarContato(c1);
        ag.adicionarContato(c2);
        ag.adicionarContato(c3);
        ag.adicionarContato(c4);
        ag.adicionarContato(c5);

        ag.listarContatos();
        String d = (ag.ehDuplicado(c1)) ? "" : " NÃO ";
        System.out.println("\nO contato " + c1.getNome() + d + " é duplicado.");
    }
}
```

a) **Execute o código acima.** Neste item você deve apenas **procurar entender com atenção o que foi feito no método *equals()*** (**da classe Contato)**, **e como o método *ehDuplicado()* (da classe Agenda) se aproveitou disso.** Estamos avaliando a ***igualdade semântica*** (e não a igualdade de ***referência***) de objetos por meio da reescrita do método equals(). Em Java, toda classe obrigatoriamente ***herda*** da classe genérica ***Object*** (ainda iremos falar detalhadamente sobre herança no curso!). O método equals() (que pertence à classe Object) pode ser *reescrito* nas classes herdeiras, se quisermos. Foi exatamente o que fizemos na classe Agenda! Bastou implementarmos nela o método equals() **com a mesma assinatura** declarada em *Object*, adicionando a diretiva **@*Override***

b) Agora sim, hora de escrever código: implemente os métodos getTel() e setTel() da classe Contato;

c) Modifique o ***construtor*** da classe Contato para que um contato **nunca possa ser criado sem um nome**, ou seja, não admitir uma String vazia (“”). Para isso, se nenhum nome for fornecido na criação de um novo contato (como foi o caso do contato ***c5*** na *driver class*), cria-lo com o nome “*Anônimo*”.

d) Modifique o método ehDuplicado(): agora, contatos devem ser considerados duplicados se **tanto o nome quanto o telefone dos mesmos forem iguais** (e não somente o nome). Crie novos contatos de teste para validar seu código.

e) Modifique o método adicionarContato() para que o mesmo só adicione um novo contato caso ele não seja uma duplicata de nenhum registro já existente na agenda.

f) Implemente o método removerContato(). Retornar ***true*** se o contato foi removido com sucesso e ***false*** caso contrário (lembre-se de atualizar o atributo ***total*** caso a remoção tenha ocorrido com sucesso!).

6. Considere a classe Funcionario **conforme o diagrama abaixo.**
   a) Crie a classe, implementando todos os seus métodos *getters* e *setters* (um *get* e um *set* para cada atributo). ***Não* crie um método *set* para o atributo "chefe" (mas crie o *get*).**
   b) Crie um método construtor, o qual deve receber todos os atributos como parâmetros de entrada e então associá-los aos campos respectivos do objeto em construção. Para cada funcionário que não possui chefe, utilizar o valor ***null*** em sua criação.

      Por exemplo, se a construção de um novo objeto Funcionario precisasse apenas do nome do Funcionario e também da referência para o chefe, mas não existe um chefe, um exemplo de construção seria

      Funcionario bill = new Funcionario("Bill Gates", null);
   c) Implemente o método ***mesmaChefia,*** que verifica se o chefe de um dado funcionário é o mesmo de um outro, retornar ***true*** se a ***matrícula*** dos chefes é a mesma ***ou*** se os chefes possuem a ***mesma referência***. Retornar ***false*** caso contrário.

Funcionario
- nome : String - cargo : String - matricula : int - idade : int - chefe : Funcionario - salário : double
+mesmaChefia(f : Funcionario) : boolean

Dica: para tal verificação, reescreva a função equals() e então a utilize dentro do método *mesmaChefia(),* de modo análogo ao que fizemos no exercício 5 (agenda de contatos).

Execute o programa e verifique sua corretude.