Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
AD2 de Programação III
2° semestre de 2015.

Nome:
Matrícula:
Pólo:

Obs: *A solução para o exercício proposto deve ser entregue em formato digital.*

Considere que fomos contratados para implementar um sistema simples de agendamento de pacientes. O código abaixo inicia a implementação desse sistema:

```
enum PLANO {
	AMIL, UNIMED, AMS, ASSIM, GOLDEN, OUTRO, PARTICULAR
}

class Contato {
	String nome;
	PLANO plano;
}

class Clientes {
	Map<Contato, GregorianCalendar> clientes;

	public Clientes() {
		this.clientes = new HashMap<Contato, GregorianCalendar>();
	}

	public Clientes(Map<Contato, GregorianCalendar> clientes) {
		this.clientes = clientes;
	}
}
```

O tipo enum lista os planos permitidos no sistema. Para exemplificar o seu uso, referirmo-nos ao plano da ASSIM escrevendo "PLANO.ASSIM". A classe Contato representa o paciente no sistema. A classe Clientes representa os pacientes do sistema. Observe que a coleção declara nesta classe é um mapa (java.util.Map), uma vez que queremos armazenar o contato associado com a hora em que este foi agendado. Logo abaixo a declaração desta coleção temos 2 construtores, os quais inicializam a coleção com um mapa vazio (primeiro construtor), ou com um mapa passado por parâmetro (segundo construtor).

a) Insira campos na classe Contato que permitam armazenar o telefone e e-mail do paciente.
b) Declare um construtor que inicialize os campos da classe.
c) Suponha que os clientes possam ser pessoas jurídicas (empresas). Crie uma classe ContatoEmpresa, a qual terá todos os campos de Contato mais o nome da empresa e seu CNPJ. Declare o construtor desta classe para inicialização de seus campos. Os objetos desta classe também poderão ser adicionados ao mesmo mapa dos clientes comuns.
d) Usando o mapa definido na classe Clientes, defina o método *agendaCliente,* o qual recebe um contato (Contato), uma data (GregorianCalendar) e adiciona essa associação no mapa. É importante observar que só é possível um agendamento por vez. Ou seja, se o cliente já existir no mapa, ele não é adicionado novamente. Caso a data existente no mapa seja anterior a data atual (agendamento passado), a data é atualizada com a nova, passada como parâmetro. Caso o agendamento ainda seja atual (data existente posterior a data atual), a data passada por parâmetro é descartada.
e) Crie no método main 2 objetos para ilustrar a manipulação das classes definidas. Os valores dos objetos são:

Tipo	Nome	Telefone	Email	Plano	Empresa	CNPJ
Física	Fulano	22222222	fulano@dominio.com	Particular		
Jurídica	Beltrano	33333333	beltrano@dominio.com	Golden Cross	UFF	11111

Adicione estes 2 objetos na coleção declarada.

f) Redefina na classe Clientes o método Object.toString, o qual retorna uma string do objeto da classe Clientes. A string retornada deve exibir os pacientes no formato abaixo:

```
Beltrano 1/10
Fulano 30/9
```

Ou seja, deve exibir o nome e a data (dia/mês) de agendamento de todos os pacientes no sistema.

Dica: Caso tenha dificuldades na iteração de objetos num mapa, observe essa discussão no site StackOverflow: http://stackoverflow.com/questions/46898/iterate-over-each-entry-in-a-map

g) No método main, chame o método toString() para imprimir o conteúdo da coleção.

RESPOSTA:

```
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;

enum PLANO {
	AMIL, UNIMED, AMS, ASSIM, GOLDEN, OUTRO, PARTICULAR
}

class Contato {
	String nome;
	String telefone;
	String email;
	PLANO plano;

	public Contato(String nome, String telefone, String email, PLANO p) {
		this.nome = nome;
		this.telefone = telefone;
		this.email = email;
		this.plano = p;
	}
}

class ContatoEmpresa extends Contato {
	String empresa;
	String cnpj;

	public ContatoEmpresa(String nome, String telefone, String email, PLANO
p, String empresa, String cnpj) {
		super(nome, telefone, email, p);
		this.empresa = empresa;
		this.cnpj = cnpj;
	}
}

class Clientes {
	Map<Contato, GregorianCalendar> clientes;

	public Clientes() {
		this.clientes = new HashMap<Contato, GregorianCalendar>();
	}

	public Clientes(Map<Contato, GregorianCalendar> clientes) {
		this.clientes = clientes;
	}

	public GregorianCalendar agendaCliente (Contato c, GregorianCalendar
data) {
		if (clientes.containsKey(c)) {
			GregorianCalendar ultimaData = clientes.get(c);
```

```
                    if (ultimaData.after(new GregorianCalendar()))
                          return ultimaData;
             }
             clientes.put(c, data);
             return data;
      }

      public String toString() {
             String saida = "";
             // http://stackoverflow.com/questions/46898/iterate-over-each-
entry-in-a-map
             for (Entry<Contato, GregorianCalendar> entrada :
clientes.entrySet()) {
                    GregorianCalendar data = entrada.getValue();
                    saida = saida + entrada.getKey().nome + " " +
data.get(Calendar.DAY_OF_MONTH) + "/" + data.get(Calendar.MONTH) + "\n";
             }
             return saida;
      }
}

public class AD2_2015_2 {
      public static void main(String[] args) {
             Contato pessoa1 = new Contato("Fulano", "22222222",
"fulano@dominio.com", PLANO.PARTICULAR);
             ContatoEmpresa empresa1 = new ContatoEmpresa("Beltrano",
"33333333", "beltrano@dominio.com", PLANO.GOLDEN, "UFF", "11111");
             Clientes agenda = new Clientes();
             agenda.agendaCliente(pessoa1, new GregorianCalendar(2015, 9, 30,
9, 0, 0));
             agenda.agendaCliente(empresa1, new GregorianCalendar(2015, 10, 1,
9, 0, 0));
             System.out.println(agenda);
      }
}
```