PROGRAMAÇÃO ORIENTADA AOS OBJECTOS
(em JAVA6)

LEI - LCC
2º ANO/2º SEMESTRE – 2009/2010

Teste Teórico – 9H00 - 21 de Junho de 2010
Cotação: 20 valores - Duração: 1h 45 m

Nota 1: Responda a cada parte em folhas separadas.
Nota 2: Cada Parte e questão tem indicada a respectiva cotação. Cada resposta será avaliada numa escala de 0-3.
Nota 3: Pode sempre usar métodos anteriormente definidos (mas correctos).

PARTE I (7 valores)

Considere o problema de base do trabalho prático deste ano lectivo no qual uma empresa seguradora relaciona clientes e seguros tal como se mostra na classe Seguradora em anexo. Segundo este modelo, o **TreeMap<String, Cliente>** representa uma tabela na qual a cada código de cliente se associa a sua ficha de informação (classe Cliente), na qual um **TreeSet<String>** representa os códigos dos seguros de que tal cliente possui. O **TreeMap<String, Seguro>** associa a cada código de seguro a informação do respectivo seguro (classe Seguro), na qual consta o código do cliente que é titular do mesmo. Assim, se relacionam clientes e seguros.

A classe Seguro contém ainda as seguintes informações sobre cada seguro, sabendo-se que cada seguro é válido por períodos de um ano, renovados automaticamente após pagamento: código do seguro, data de início, número de anos, preço base, taxa de agravamento por cada activação do seguro, número de activações no ano corrente e taxa de bonificação em caso de não ter activações. A classe Seguro é abstracta pois apenas contém a informação comum aos vários tipos de seguros que podem existir, tais como Saúde, Veículos, Vida, etc.

Tendo esta informação e estas classes como referencial de base e assumindo que em tais classes estão pré-definidos os métodos get(), set(), construtores, toString(), clone() e equals(), programe em Java as seguintes operações da classe Seguradora:

1.- Considerando apenas o valor do preço base, pretende-se um método que calcule o valor total a receber pela seguradora (se todos os titulares pagarem a horas) no mês actual (cf. dataSistema). (2,0)

2.- Método que grave num ficheiro de texto toda a informação de todos os seguros que se encontram em atraso de pagamento e que possuem mais do que uma activação. (2,5)

3.- Método que dê como resultado uma tabela na qual a cada código de cliente se associam todas as informações referentes a todos os seus seguros de um tipo dado como parâmetro do método. (2,5)

PARTE II (7 valores)

4.- Escreva o código de um método que devolva sob a forma de um TreeSet<Seguro> a informação completa de todos os seguros que foram iniciados num determinado ano que é dado como parâmetro. (2,0)

5.- Considerando uma colecção TreeSet<Seguro> desenvolva todo o código necessário para que uma colecção deste tipo possa ser ordenada por data e exemplifique como codificaria um método que realize tal ordenação e como este poderia ser invocado. (3,0)

6.- Escreva um método que dado o código de um cliente determine se o cliente possui algum seguro em atraso de pagamento. (2,0)

PARTE III (6 valores)

Cada classe representativa de um dado tipo de seguro (cf. SeguroVeiculos em anexo) representa numa variável de classe todas as cláusulas que podem ser escolhidas para um seguro concreto e declara uma variável de instância que todas as subclasses herdam (cf. SeguroLigeiros) e que representa as cláusulas escolhidas pelo cliente de um seguro concreto que foi criado. Cada cláusula possui um código, um texto, e uma taxa de agravamento do preço base do seguro.

7.- Escreva o código de um método que dado o código de um seguro calcule o valor total a pagar, usando todas as informações conhecidas referentes a um seguro, tais como preço base, agravamentos, bonificações e preços das suas claúsulas. (3,0)

8.- Escreva o código que se deveria programar no método main() para que, considerando uma variável SeguroLigeiros segLig = new SeguroLigeiros(); esta possa ser gravada em ObjectStream e possa ser lida da stream onde foi gravada. (3,0)

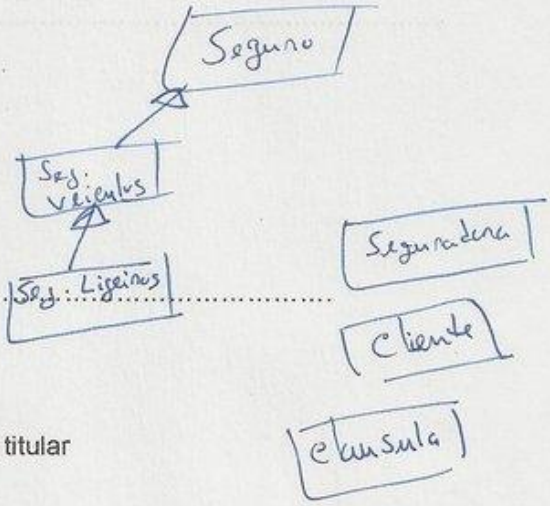
Prof. F. Mário Martins

ANEXO – CLASSES

```
public class Seguradora implements Serializable {
   // Decisão: Cada uma das tabelas indexa a outra, ou seja, cada cliente vai possuir
   // um conjunto de códigos de seguros de que é titular, e cada seguro possui o
   // código do cliente que é o seu titular
   private TreeMap<String, Cliente> clientes = new TreeMap<String, Cliente>();
   private TreeMap<String, Seguro> seguros = new TreeMap<String, Seguro>();
   private GregorianCalendar dataSistema;                  // data de referência da seguradora
   private TreeSet<String> nomesSeguros = new TreeSet<String>();     // tipos de seguros
```

```
public class Cliente implements Serializable {

   private String codCliente = "";
   private String nomeCliente = "";
   private String moradaCliente = "";
   private String nifCliente = "";
   private String biCliente = "";
   private TreeSet<String> codSeguros = new TreeSet<String>();
```




```
public abstract class Seguro implements Serializable {

   private String codSeguro = "";
   private String codTitular = "";        // é o código do Cliente que é o titular
   private GregorianCalendar dataInicio = null;
   private int numAnos;
   private double precoBase = 0.0;
   private double taxaActiv = 0.0;
   private int numActivEsteAno = 0;       // quando o titular paga anuidade este valor vai a 0 !
   private double taxaBonus = 0.0;
```

```
public class Clausula implements Serializable {

   private String codClausula;
   private String txtClausula;
   private double taxaClausula;
```

```
public abstract class SeguroVeiculos extends Seguro {

   // Variáveis e Métodos de Classe
   private static TreeMap<String, Clausula> tabClaus = new TreeMap<String, Clausula>();
   public static void addClausulaSeg(String cod, Clausula cl) {
      tabClaus.insereClausula(cl.clone());
   }
   public static void removeClausula(String cod) { tabClaus.removeClausula(cod); }

  // Variáveis de Instância
  private TreeMap<String, Clausula> tabClausulas = new TreeMap<String, Clausula>();
  // as clausulas desta tabela devem ser um subconjunto das clausulas de tabClaus
```

```
public class SeguroLigeiros extends SeguroVeiculos {

   // Variáveis e Métodos de Classe
   private static TreeMap<String, Clausula> tabClausVeic = new TreeMap<String, Clausula>();
   public static void addClausulaSeg(String cod, Clausula cl) {
      tabClausVeic.insereClausula(cl.clone());
   }
   public static void removeClausula(String cod) { tabClausVeic.removeClausula(cod); }

  // Variáveis de Instância
  private TreeMap<String, Clausula> tabClausulas = new TreeMap<String, Clausula>();
  // as clausulas desta tabela devem ser um subconjunto das clausulas de tabClausVeic
.....
```