PROGRAMAÇÃO ORIENTADA AOS OBJECTOS
(em JAVA5/6)

LEI - LCC
2º ANO/2º SEMESTRE – 2007/2008

Exame de Recurso – 10 de Julho de 2008
Cotação: 20 valores Duração: 120 m

Nota 1: Responda a cada parte em folhas separadas.
Nota 2: Cada questão tem a respectiva cotação indicada no fim da mesma.
Nota 3: Pode sempre usar métodos anteriormente definidos (mas correctos).

Numa dada exploração agrícola, um sistema de informação regista todas as medições de temperatura e humidade realizadas nas suas várias estufas, cada uma delas devidamente codificada.
Apresentam-se em seguida as definições das principais classes que constituem tal sistema de informação.
A classe **FichaReg** (*ficha de registo*) representa o tipo de informação que é obtida num dado momento a partir do sensor de uma estufa, designadamente, *o código da estufa*, *a temperatura* e *a humidade*.

```
public class FichaReg implements Serializable {
    private String estufa;
    private double temp;
    private double humidade;
    // ......
    public FichaReg(String cod, double tmp, double humid) {
          estufa = cod; temp = tmp; humidade = humid;
    }
    ..............
    // os métodos getX(), setX(..), toString(), equals() e clone() estão disponíveis
}
```

A classe **Hora** servirá para registar os momentos em que cada leitura foi efectuada.

```
public class Hora implements Serializable, Comparator<Hora> {
    private int hora;
    private int minuto
    private int seg;
    ...........
    // os métodos getX(), setX(..), toString(), equals() e clone() estão disponíveis
}
```

PARTE I (7.5 valores)

1) Escreva o código do método que garante que a classe **Hora** implementa de facto, e de forma standard, a interface **Comparator<Hora>** (1.5);
2) Apresente uma implementação da classe **MinhaHora** (que serviria para registar igualmente a hora, os minutos e os segundos de um registo), mas em que **MinhaHora** é subclasse de **GregorianCalendar**, apresentando as suas variáveis de instância, o construtor de cópia, o método `int getMinuto()` e o método `String toString()` (2.5);
3) Considerando agora, de novo, a classe **Hora** apresentada acima, escreva o código do método `long getTimeInSeconds()` que converta uma dada *hora* no seu equivalente em segundos (1);

4) Desenvolva uma classe, designada **OrdenacaoDeHoras** que permita guardar da melhor forma possível, várias implementações de **Comparator<Hora>**, cada uma delas acessível por uma dada designação textual (exº "CRESCENTE", "DECRESCENTE", etc.), e apresente os métodos que permitem inserir, seleccionar e remover de tal classe os diferentes comparadores nela registados. (2.5);

PARTE II (6 valores)

Chegamos assim à classe principal que se pretende implementar, a classe **RegistoEstufas**. A classe **RegistoEstufas** é representada por um **TreeMap** que associa uma **hora** a cada **ficha de registo**, não existindo nunca, portanto, dois registos no mesmo momento (ou seja, *hora*).

```
public class RegistoEstufas implements Serializable {
    private TreeMap<Hora, FichaReg> registos;
    // Trata-se um TreeMap que a cada hora associa uma ficha de registo
    .....
    // Construtores
    .....
    // Métodos de instância
    .....
}
```

São os seguintes os métodos de **RegistoEstufas** que devem ser implementados:

5) Escreva o código do construtor,

```
public RegistoEstufas(TreeMap<Hora, FichaReg> regs, Comparator<Hora> ch)
```

que inicializa a variável registos usando os objectos do **TreeMap** parâmetro, mas usando também o comparador dado como parâmetro (2.0);

6) Método que devolve uma lista de todos os registos efectuados numa dada hora e minuto, independentemente dos segundos (exº 13h15m*s) (1.5);
7) Método que dê como resultado a temperatura média das estufas observadas (ou seja, de todas as registadas) (1.0);
8) Método que dê como resultado o conjunto de horas em que se verificou uma dada temperatura na estufa dada como parâmetro (1.5);

PARTE III (6.5 valores)

9) Método que grava numa ObjectStream de nome dado todos os registos realizados entre duas Horas (h1 e h2, sendo h1 < h2) dadas como parâmetro (1.5).
10) Método que devolva a média das humidades registadas numa dada estufa (1.5);
11) Método que construa e dê como resultado uma tabela, na qual, a cada estufa (código) seja associada uma outra tabela onde a cada instância de **Hora** se associe a respectiva temperatura registada (1.5);
12) Método que leia de um ficheiro de texto linhas da forma "estf1/12/55/45/21/75/" e devolva o respectivo **TreeMap<Hora, FichaReg>** (2.0);

Prof. F. Mário Martins