Exame de Recurso de Programação Orientada aos Objectos

2013.07.02
Duração: **2h**

Leia o teste com muita atenção antes de começar.
Responda a cada Parte em folhas separadas.

Parte I - 6 valores

1. Relembre o exercício do sistema de gestão de email (retirado das fichas práticas). As seguintes classes foram desenvolvidas para representar mensagens:

```
public class Comunicacao {
  private String remetente;
  private String destinatario;
  private GregorianCalendar data;
  ...
}

public class Mensagem extends Comunicacao {
  private String assunto;
  private String texto;
  ...
}
```

Para implementar um servidor de mensagens desenvolveu-se depois a classe `MsgMap`, que associa a cada destinatário todas as mensagens por ele recebidas (ordenadas por ordem de chegada):

```
public class MsgMap {
  private TreeMap<String, ArrayList<Mensagem>> msgBox;
   ...
}
```

Escreva os seguintes métodos para a classe MsgMap:

(a) O construtor de cópia

(b) Um método para determinar o número de mensagens no sistema: `public int tamanho()`

(c) Um método para determinar quantos emails tem por origem um dado remetente: `public int quantosDe(String remetente)`

(d) Um método para, dada uma palavra, elimina todos os emails que tenham essa palavra no assunto: `public void antiSpam(String palavra)`

(e) Um método para devolver um Map, associando a cada remetente as mensagens por ele enviadas: `public Map<String, List<Mensagem>> msgDeRemetente()`

Parte II - 5 valores

2. Ainda em relação ao exercício anterior, considere que foram também desenvolvidas as seguintes subclasses:

```
public class SMS extends Mensagem {
   private static int MAX_SIZE = 160;

   private long numberId;
   private int totalParts;
       // um SMS com mais caracteres que MAX_SIZE é partido
       // em vários de até esse tamanho e custa o valor dos vários
       // SMS de tamanho normal
   private int number;
   ...
}

public class EMail extends Mensagem {
   private String format;
   ...
}

public class Telefonema extends Comunicacao {
    private int duracao;
    ...
}
```

É-lhe agora pedido que desenvolva um sistema de gestão de custos para um operador móvel. Numa primeira versão, a estrutura de dados a utilizar será um Map que associe um número de cliente à relação ordenada das comunicações efectuadas:

```
public class ComunicacoesMoveis {
   private HashMap<String,ArrayList<Comunicacao>> comms;
   ...
}
```

(a) escreva o método da classe `ComunicacoesMoveis` para adicionar uma comunicação:
`public void adComunicacao(Comunicacao com) throws ComunicacaoExiste`

(b) o operador móvel tem o seguinte modelo de facturação: as comunicações verbais são taxadas por duração da chamada, os emails pela quantidade de caracteres e os sms custam 7 cêntimos por mensagem (note no entanto que um sms grande é partido em sms mais pequenos). Dessa forma, codifique o que for necessário (justificando as decisões tomadas) nas classes anteriormente identificadas, bem assim como o método `public int factura(String numeroOriginador, GregorianCalendar inicio, GregorianCalendar fim)` , da classe `ComunicacoesMoveis`, que determina o valor a pagar ao operador móvel pelo cliente parâmetro.

Parte III - 5 valores

3. Recorde o projecto prático de POO. Um dos grupos idealizou a seguinte estrutura para o registo dos tempos obtidos durante uma corrida:

```
public class Corrida {
  ...
  private ArrayList<TreeMap<String,Tempo>> temposPorVolta;
  ...
}

public class Tempo {
   private int minutos;
   private int segundos;
}
```

O arraylist `temposPorVolta` apresenta os tempos da volta 1 na posição 0. A chave do `TreeMap<String,Tempo>` corresponde à identificação do carro (exemplo: o número). Sempre que um carro deixa de ter tempo numa volta significa que desistiu nessa volta.

Tendo em conta esta estruturação responda às seguintes perguntas:

(a) desenvolva o método `public List<String> classificacaoNaVolta(int volta) throws ...`, sendo que os carros que desistem ficam ordenados pela volta em que deixam de ter tempo registado, isto é, um carro que desiste na volta 3 fica à frente de um carro que desista na volta 2.

(b) `public Tempo tempoRecorde()`, que determina a volta mais rápida que foi efectuada na corrida.

Parte IV - 4 valores

4. Considere que é necessário guardar em parque fechado os carros que disputam as 24 horas de Le Mans. No entanto o parque fechado tem uma lotação que não permite que todos os carros possam lá caber (e ter assistência técnica, etc.), pelo que é necessário gerir o funcionamento deste parque. Considere que foi desenvolvida a classe `ParqueFechado` que implementa a interface `AcessoParque`:

```
public interface AcessoParque {
  public void entra(Carro v) throws ParqueCheioException;
  public void sai(String numeroCarro) throws CarroNaoExisteException;
}
```

Responda às seguintes questões:

(a) desenvolva a classe `ParqueFechado`, declarando as suas variáveis de instância e os métodos de `AcessoParque`.

(b) Tendo em conta que a classe `ParqueFechado` já existe, desenvolva a classe `ParqueComFilaDeEspera` que, para além de implementar `AcessoParque` disponibiliza uma fila de espera para os carros que não têm lugar de momento no parque. Para saber os carros que estão na fila de espera, deve ser disponibilizado o método `Collection<String> getElementosEmFila()`, que devolve uma colecção com o número dos carros em espera.