— Exame —
Desenvolvimento de Sistemas de Informação

LESI/LMCC
Chamada 3 - 2001/02

02/09/2002

Duração máxima: 2h00
Leia as questões com atenção.

Grupo I

Considere os seguintes extractos de código Java:

```
public class ContasDB {

    // Variáveis de instância
    private SetConta contas;

    // Fazer um Depósito
    // Testar: existeConta(numConta)
    public void fazDeposito(String numConta, double quantia) {
        Conta c = contas.getConta(numConta);

        contas.remove(c);
        c.deposito(quantia);
        contas.add(c);
    }
}

public class SetConta {

    // Variáveis de instância
    private HashSet setConta;   // Um HashSet de Conta

    // adicionar conta
    public void add(Conta c) {
        setConta.add(c);
    }

    // remover conta
    public void remove (Conta c) {
        setConta.remove(c);
    }

    // consultar conta
    // Testar: contains(num)
```

```
    public Conta getConta (String num) {
        Iterator i = setConta.iterator();
        Conta c = (Conta) i.next();
        boolean b = c.is(num);

        while (!b) {
            c = (Conta) i.next();
            b = c.is(num);
        }
        Conta nc = (Conta) c.clone();
        return nc;
    }
}

public class Conta implements Cloneable {

    // Variáveis de Instância
    private String nome, numero;
    private double saldo;

    // Verificar número
    public String is(String num) {
        return this.numero==num;
    }

    // Depósito
    // testar: Conta.quantiaValida(qt)
    public void deposito(double qt) {
        this.saldo += qt;
    }

    public Object clone() {
        Conta cr = new Conta(this.nome, this.numero, this.saldo);
        return cr;
    }
}
```

1. Descreva o comportamento do método `fazDepósito` utilizando um **Diagrama de Sequência**.
2. Sabendo que existem três sub-classes de Conta (Ordem, SuperConta, Ordenado), descreva a estrutura de classes do código utilizando um **Diagrama de Classes** (ao indicar os atributos/métodos de cada classe é suficiente indicar o nome).

Grupo II

Considere a seguinte interface Java:

```
public interface Encomenda {
    public void disponível(Vector linhas);
    public void disponível();
    public void envia();
    public void cancela();
    public void adiciona(Vector linhas);
}
```

Uma classe que implemente a interface Encomenda deverá ter o seguinte comportamento:

> Após ser criada uma encomenda fica pendente a aguardar que todas as suas linhas sejam satisfeitas.
>
> Existem duas formas de registar que uma (ou mais) linha(s) da encomenda já está/estão satisfeita(s). O método `disponível(Vector linhas)` assinala a satisfação das linhas passadas como parãmetro. O método `disponível()` assinala a satisfação de todas as linhas ainda por satisfazer. Após estar totalmente satisfeita a encomenda passa a poder ser enviada.
>
> O envio da encomenda é registado utilizando o método `envia()`. Após ser enviada a encomenda fica registada como expedida.
>
> Em qualquer momento antes do envio é possível cancelar a encomenda (método `cancela()`). Nesse caso ela fica registada como cancelada.
>
> O método `adiciona(Vector linhas)` pode ser utilizado em qualquer encomenda que não cancelada ou expedida para lhe adicionar novas linhas.

Escreva um **Diagrama de Estado** que represente o comportamento descrito.

Grupo III

Na estação de serviço auto FORADEHORAS são efectuados três tipos de serviço: mudanças de óleo, lavagem, e montagem de pneus. Para precaver a possibilidade de ampliação da gama de serviços prestados, cada tipo de serviço é identificado por um código. A cada tipo de serviço está ainda associado um preço base. Ao tipo de serviço mudança de óleo está também associado o preço por litro do óleo em utilização, à lavagem está associado o preço por hora da mão de obra. Para cada tipo de serviço existem um ou mais pontos de atendimento. Em cada ponto de atendimento trabalham um ajudante e um encarregado. Cada encarregado pode apenas efectuar um tipo definido de serviço, os ajudantes podem trabalhar em qualquer serviço.

A alocação de funcionários a pontos de atendimentos é efectuada numa base semanal pela gerência da estação de serviço. A gerência pretende informatizar esse processo. Para além da gestão normal de funcionários e postos de atendimento (registo, consulta e alteração) o sistema a desenvolver deverá suportar as seguintes tarefas:

- listagem dos funcionários não alocados a um ponto de atendimento;
- listagem dos pontos de atendimento a que faltem funcionários;
- alocação de um funcionário a um ponto de atendimento.

Tendo em atenção o exposto, responda às seguintes questões (note que os diagramas pedidos são interdependentes):

1. Identifique os Actores e Use Cases contidos na descrição dada e escreva o **Diagrama de *Use Case*** correspondente.
2. Proponha um **Diagrama de Classe** para o sistema (para cada classe, inclua os atributos que considerar mais relevantes).
3. Periódicamente é verificada a integridade da alocação de funcionários. Essa verificação é efectuada testando se o tipo de serviço do encarregado de cada ponto de atendimento é igual ao serviço prestado nesse ponto. Esta verificação é desencadeada invocando o método

   ```
   public boolean check()
   ```

   da classe principal do sistema. Proponha um **Diagrama de Sequência** para esse método.