— Exame —
Desenvolvimento de Sistemas de Informação

LESI/LMCC
1ª Chamada - 2005/06

16/06/2006

Duração máxima: 2h00
Leia o exame com atenção e responda utilizando UML 2.0.

Grupo I

1. Considere um sistema de inscrições em exames. O sistema deverá ser utilizado por alunos para se inscreverem, pelos SAUM para criarem o calendário de exames, e pelos professores para consultarem os alunos inscritos.

   Desenhe um diagrama UML em que identifique quais são, para si, os cinco principais requisitos funcionais de tal sistema. O diagrama deverá identificar três situações de erro tratadas pelo sistema.

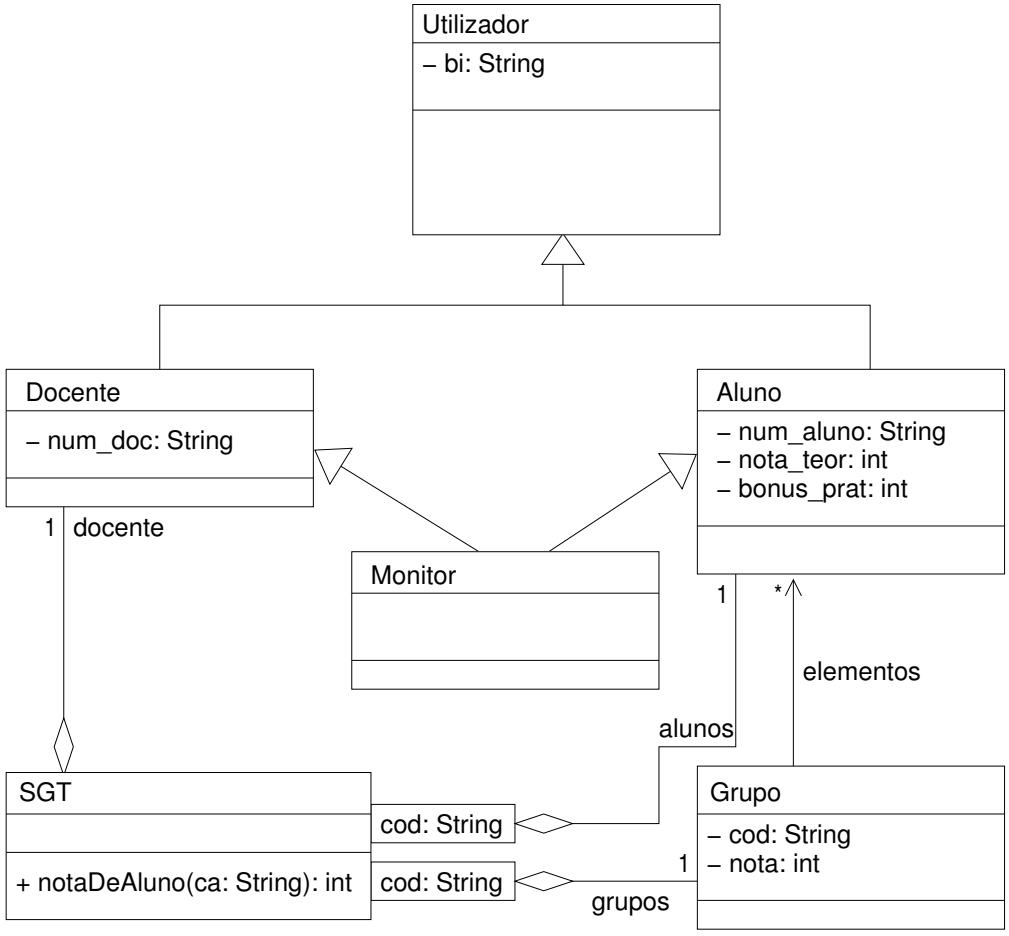
2. Considere agora que se pretende que o sistema possibilite a inscrição em exame, mesmo que já fora de prazo. Nesse caso deverá ser implementado um sistema de cotas. O funcionamento do sistema deverá ser o seguinte:

   > O aluno terá que efectuar o pedido para se inscrever numa dada chamada, caso a cota para essa chamada ainda não tenha sido excedida o aluno é informado que ficou inscrito, ao mesmo tempo que é enviada uma mensagem ao docente a informá-lo do novo número de inscritos em cada chamada. Caso a cota já esteja excedida, o pedido é enviado ao docente para aprovação. Caso o docente aprove o pedido o processo é semelhante ao anteriormente descrito. Caso o docente não aprove o pedido, é oferecida ao aluno a possibilidade de se inscrever na outra chamada. Se o aluno aceita o docente é notificado do novo número de inscritos em cada chamada, caso contrário nada mais acontece.

   Desenhe um **Diagrama de Actividades** para o comportamento descrito, identificando quais as actividades realizadas por cada actor envolvido no processo.

Grupo II

Considere a proposta de arquitectura apresentada na figura:



1. Com base na arquitectura proposta (e sabendo que as tabelas são implementadas com Map e as listas com List), desenhe um **Diagrama de Sequência** para o método `int notaDeAluno(String ca)` (da classe SGT) que calcula a nota de um aluno.

   (considere que a nota é calculada pela fórmula: .5*Nota_Teórica+.5*Bonus_Prática*Nota_Grupo)

2. Sabendo que esta arquitectura vai ser implementada em Java:

   (a) Redesenhe o diagrama de modo a evitar a utilização de herança multipla.
   (pode acrescentar ao modelo os elementos arquitecturais que lhe parecerem necessários)

   (b) Escreva o código para as classes SGT e Grupo.

Grupo III

Considere o seguinte excerto de código Java:

```
public class Turma extends Observable {
    private Map<String,Aluno> turma;
    private AlunoFactory fabrica;

    public Turma() {
        this.fabrica = new AlunoFactory();
        this.turma = new TabAl();
    }

    public Aluno getAluno(String num) throws TurmaException {
        Aluno res = this.turma.get(num);
        if (res == null) { res = this.fabrica.mkAluno("","",0,0); }
        return res;
    }
}

public class TabAl implements Map<String,Aluno> {
    public Connection conn;

    public boolean containsKey(Object key) throws NullPointerException {
        try { Statement stm = conn.createStatement();
              String sql = "SELECT nome FROM TAlunos WHERE numero='"+(String)key+"'";
              ResultSet rs = stm.executeQuery(sql);
              return rs.next(); }
        catch (Exception e) {throw new NullPointerException(e.getMessage());}
    }

    public Aluno get(Object key) {
        Aluno al = null;
        Statement stm = conn.createStatement();
        String sql = "SELECT * FROM TAlunos WHERE numero='"+(String)key+"'";
        ResultSet rs = stm.executeQuery(sql);
        if (rs.next())
          al = new Aluno(rs.getString(2),rs.getString(1),rs.getInt(3),rs.getInt(4));
        return al; }
    }
}
```

1. Desenhe um **Diagrama de Interacção** à sua escolha para o método `Aluno getAluno(String num)` da classe `Turma`

   (não necessita modelar o tratamento de excepções, nem a concatenação de Strings).