— Exame —
Desenvolvimento de Sistemas de Informação

LESI/LMCC
2ª Chamada - 2005/06

12/07/2006

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

Grupo I

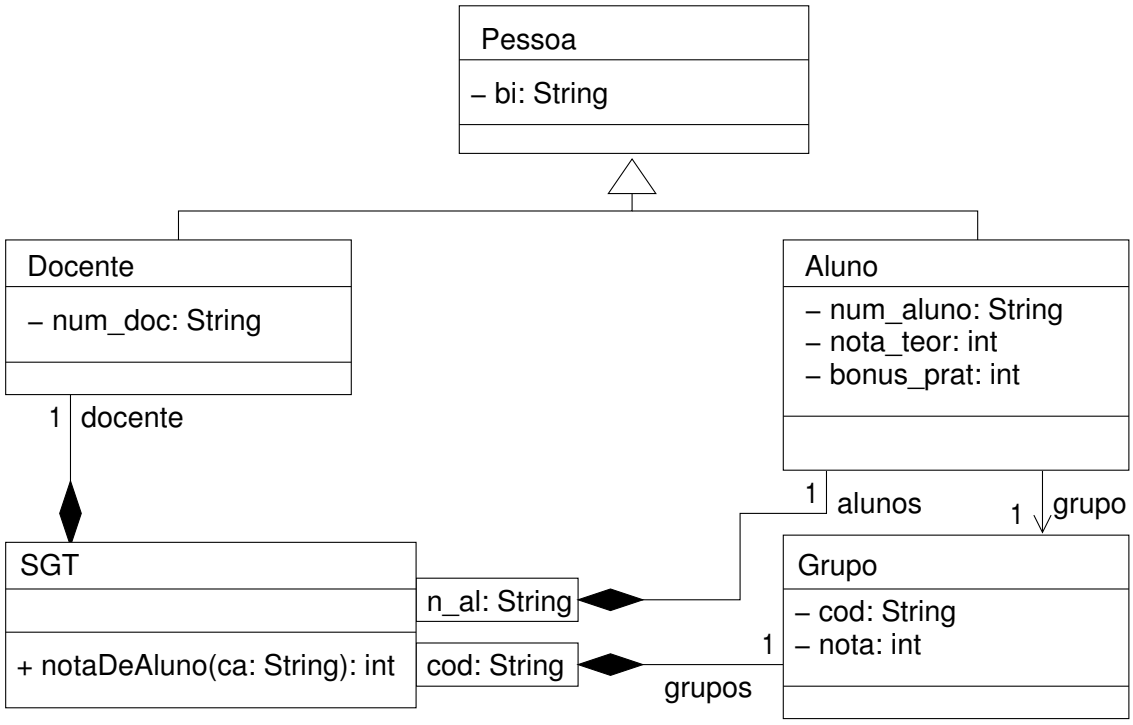
Considere um sistema de inscrições em exames.

1. Desenhe um diagrama UML, e descreva os *Use Case* associados, em que mostre que o sistema deverá obedecer aos seguintes requisitos funcionais:

   - Os possíveis utilizadores do sistema são os alunos, os SAUM e os professores, sendo obrigatório que estejam identifiquem perante o sistema para o poderem utilizar.
   - Existem dois tipo de autenticação possíveis: através de um par nome de utilizador/palavra passe, ou através de um processo de reconhecimento de impressões digitais.
   - O registo dos utilizadores está a cargo dos SAUM. Para cada utilizador devem ser indicados o número, o nome, e uma conta de email. O sistema avisa caso se tente registar um utilizador com número, ou conta de email, iguais às de um utilizador já existente.

Grupo II

Considere a proposta de arquitectura apresentada na figura:



1. Sabendo que o método `int notaDeAluno(String ca)` já está implementado, desenhe, com base na arquitectura proposta (e sabendo também que as tabelas são implementadas com Map e as listas com List), um **Diagrama de Sequência** para o método `List<Aluno> grupo(String cod)` (da classe SGT) que calcula a lista de todos os alunos que pertencem ao grupo com código `cod`.

2. Sem utilizar herança múltipla, redesenhe o diagrama de classes acima de modo a acomodar as seguintes alterações aos requisitos:

   (a) A arquitectura deverá incluir uma tabela de utilizadores (os utilizadores do sistema são os docentes e os grupos).

   (b) Deverá ser permitido que o grupo de cada aluno possa variar ao longo do semestre (dada uma data, deverá ainda ser possível saber o grupo de cada aluno nessa data).

Grupo III

Considere o seguinte excerto de código Java:

```
public class Compras extends Observable implements Serializable {
  private Map<String,Comprador> baseDados;

  public String eComprador(String bilhete) throws SGCException {
    boolean encontrado = false;
    Set s = this.baseDados.keySet();
    Iterator i = s.iterator();
    Object chave = null;
    Comprador comprador;
    try {
      while(!encontrado) {
        chave = i.next();
        encontrado = testa(chave, bilhete);
      }
    }
    catch (NoSuchElementException e) {
      throw new SGCException("Bilhete nao vendido!");
    }
    return (String)chave;
  }

  public boolean testa(Object chave, String bilhete) {
    Comprador comprador = this.baseDados.get(chave);
    boolean b = comprador.comprou(bilhete);
    return b;
  }
}

public class Comprador implements Cloneable {
  private List<Bilhete> bilhetes;
  private boolean isActive;
  ...
  public boolean comprou(String b) {

    for (Bilhete bi : bilhetes)
      if (bi.temNome(b))
        return true;
    return false;
  }
  ...
}
```

1. Desenhe um **Diagrama de Classes** que represente as classes bem como toda a informação existente sobre elas.

2. Escreva um **Diagrama de Comunicação** que descreva o comportamento do método `String eComprador(String bilhete)`, considerando apenas a situação em que não ocorrem erros.