— Exame —
Desenvolvimento de Sistemas Software

LEI 2011/12

08/02/2012

Duração máxima: 2h00

Leia o exame com atenção e responda a cada grupo em folhas separadas!

Grupo I – Modelação de requisitos

Na cadeia de lojas de café 1XPRS é possível comprar, quer cápsulas de café, quer máquinas de café. A cada venda é associado um código de transacção único. Seja qual for a venda efectuada (cápsulas, máquinas), ficam registados o comprador, a loja em que a venda foi efectuada e a data da compra. Às vendas de cápsulas fica também associada a máquina em que o café será utilizado. Às vendas de máquinas poderá ficar associada uma primeira compra de cápsulas, caso ele tenha sido efectuada em simultâneo. Para cada tipo de venda existem um ou mais pontos de atendimento em cada loja. Em cada ponto de atendimento trabalham um ajudante e um encarregado. Cada encarregado pode apenas efectuar um tipo de venda (cápsulas, máquinas), os ajudantes podem trabalhar em qualquer posto.

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

- listagem dos funcionários de uma loja não alocados a um ponto de venda (acessível quer ao gerente da loja, quer à gerência da cadeia de lojas);
- listagem dos pontos de atendimento a que faltem funcionários (acessível quer ao gerente da loja, quer à gerência da cadeia de lojas);
- alocação de um funcionário a um ponto de atendimento (acessível apenas ao gerente da loja).

Tendo em atenção o exposto, responda às seguintes questões:

1. Proponha um **Modelo de Domínio** para o problema descrito.

2. Identifique os Actores e Use Cases contidos na descrição dada e desenhe o **Diagrama de *Use Case*** correspondente.

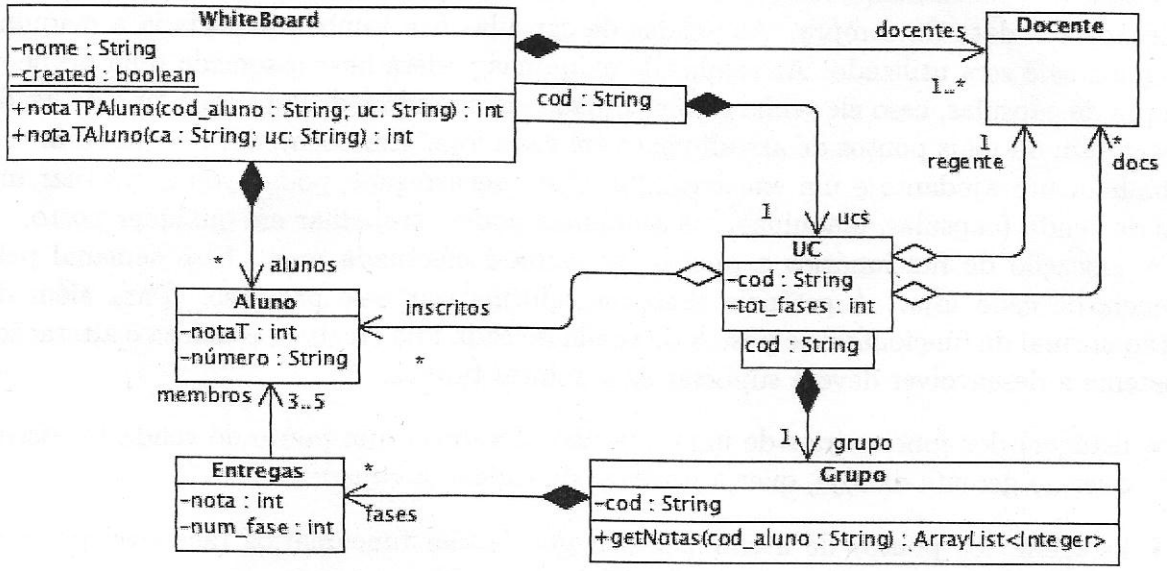
3. Desenhe o **Diagrama de Sequência de Sistema** do *Use Case* correspondente à alocação de um funcionário a um ponto de venda sabendo que:

   - o *Use Case* tem sucesso quando aloca um funcionário a um posto;
   - o *Use Case* falha quando não é possível alocar um funcionário a um posto;
   - um posto não pode ter mais que um ajudante e um encarregado (por exemplo, não deve ser possível alocar um segundo ajudante a um posto);
   - o sistema tem a capacidade de calcular a lista de todos os postos de atendimento com falta de pessoal (que pode ser vazia);
   - em cada momento, um funcionário só pode estar alocado a um posto, devendo ser possível transferi-lo de um posto para outro no momento da alocação.

(Não se esqueça de responder a cada grupo em folhas separadas!)

Grupo II – Modelação de Arquitectura / Comportamento

Considere a proposta de arquitectura apresentada na figura:



4. Numa folha diferente da utilizada para o Grupo I, desenhe um **Diagrama de Sequência** para o método
   List<Aluno> getAlunos(cod_uc: String, cod_grupo: String, nota: int)
   (da classe WhiteBoard) que devolve uma lista com os alunos que fizeram entregas com uma dada nota, num dado grupo de uma dada UC.

5. Considere a seguinte interface Java:

```
public interface Receita {
    public boolean dispensa(ArrayList<LinhaReceita> linhas);
    public boolean dispensa();
    public void paga();
    public void anula();
    public void adiciona(ArrayList<LinhaReceita> linhas);
}
```

Sabendo que uma classe que implemente a interface Receita deverá ter o seguinte comportamento:

- Após ser criada uma receita fica pendente, a aguardar que todas as suas linhas sejam satisfeitas.
- Existem duas formas de registar que uma (ou mais) linha(s) da receita já está/estão satisfeita(s). O método `dispensa(ArrayList<LinhaReceita> linhas)` assinala a satisfação das linhas passadas como parâmetro. O método `dispensa()` assinala a satisfação de todas as linhas ainda por satisfazer.
- Quando todas a linhas estiverem satisfeitas, a receita passa de pendente a dispensada. Após estar dispensada a receita passa a poder ser paga.
- O pagamento da receita é registado utilizando o método `paga()`. Após ser paga a receita fica registada como aviada.
- Em qualquer momento antes do pagamento é possível anular a receita (método `anula()`). Nesse caso ela fica registada como anulada.
- O método `adiciona(ArrayList<LinhaReceita> linhas)` pode ser utilizado em qualquer receita que não anulada ou aviada para lhe adicionar novas linhas.

Numa folha diferente da utilizada para o Grupo I, desenhe um **Diagrama de Máquina de Estado** que represente o comportamento descrito.