

Desenvolvimento de Sistemas Software

Licenciatura em Engenharia Informática

2014/2015

Prática Laboratorial UML #05

António Nestor Ribeiro

anr@di.uminho.pt

José Creissac Campos

jose.campos@di.uminho.pt

Conteúdo

1	Objectivos	3
2	Exercícios	3
2.1	Compras online	3
2.2	Sistema de Avaliação de Trabalhos	4

1 Objectivos

1. Praticar a utilização de Diagramas de Sequência e de Diagramas de Classe;
2. Relacionar estes diagramas com a implementação que eles representam.

2 Exercícios

Para os exercícios abaixo propostos analise os enunciados e responda às questões criando os respectivos diagramas.

2.1 Compras online

Considere o seguinte extracto de código Java (o método `comprados` calcula um `ArrayList` com todos os bilhetes comprados por um dado comprador). Note que o `ArrayList res` é passado por referência no método `addBilhetes`.

```
public class Compras {

    private HashMap<String,Comprador> baseDados; // idComprador -> Comprador
    ...

    public ArrayList<String> comprados(String bi) {
        ArrayList<String> res;
        boolean existe = this.baseDados.containsKey(bi);
        if (existe)
            res = this.calcula(bi);
        return res;
    }

    public ArrayList<String> calcula(String bi) {
        Comprador c = this.baseDados.get(bi);
        ArrayList<String> res = new ArrayList<String>();
        c.addBilhetes(res);
        return res;
    }
    ...
}

public class Comprador {
```

```
private ArrayList<String> bilhetes;
...

public void addBilhetes(ArrayList<String> res) {
    String o;
    int i=0;
    int tam = this.bilhetes.size();

    while(i < tam) {
        o = this.bilhetes.get(i);
        res.add(o);
        i++;
    }
}
...
}
```

1. Escreva um Diagrama de Sequência que descreva o comportamento do método **comprados**.
2. Considere agora que no método **addBilhetes** o ciclo **while** é substituído por:

```
res.addAll(this.bilhetes);
```

Refaça o Diagrama de Sequência da pergunta anterior, agora com a nova versão do método.

2.2 Sistema de Avaliação de Trabalhos

Considere o excerto de código Java que a seguir se apresenta:

```
interface Identificavel {
    void getID();
}

abstract class Pessoa {
    private String nome;
    abstract void setNome(String n);
}

class Aluno extends Pessoa implements Identificavel {
    private Grupo m_g;
```

```
    private int numAluno;
    private int notaTeo;
    private int bounsPrat;
    void regista(Grupo g) {...};
}

class Grupo {
    private int cod;
    private int nota;
    private ArrayList<Entrega> entregas;
    void addEntrega(Entrega e) {...}
}

class Entrega implements Identificavel {
    private Date data;
    private int nota_docente;
    private Aluno avaliador;
    private int nota_avaliador;
    private String comentarios;
}

class Docente extends Pessoa implements Identificavel{
    private int cod;
}

class SGT {
    private Docente responsavel;
    private ArrayList <Docente> docentes_praticas;
    private TreeMap <Integer,Aluno> alunos;
    private ArrayList <Grupo> grupos;
    int getNotaAluno(int codAluno) {...}
    boolean validaAvaliadores() {...}
}
```

Relativamente ao código apresentado:

1. Analise o código e apresente o correspondente Diagrama de Classes, procurando ser o mais exaustivo possível na identificação dos relacionamentos entre as classes.
2. Desenhe o Diagrama de Sequência para o método `boolean validaAvaliadores()`. O método deverá verificar que nenhum aluno seja avaliador do seu próprio grupo.