Instituto de Computação
Bacharelado em Ciência da Computação
Disciplina: Técnicas de Programação Avançada
Segunda lista de exercícios - 2014.2

Padrões de projeto

1) Considere uma classe Cliente que precisa interagir com uma classe BaseDeDados para efetuar operações de cadastro, recuperação, atualização e remoção de instâncias de uma classe Modelo que possui vários Elementos. O Cliente precisa estabelecer uma conexão como a BaseDeDados receber uma instância de uma classe Conexão e a partir da Conexão efetuar as operações desejadas. Forneça uma solução utilizando padrões que simplifique a interação do Cliente com as demais classes deste subsistema.

   Dica: usar o padrão Façade

2) Uma classe Cliente precisa requisitar serviços de uma classe preexistente cuja interface não é a interface esperada pela classe Cliente. Estabeleça uma solução usando padrões que resolva o problema de discrepância das interfaces.

   Dica:usar o padrão Adapter

3) Um projetista precisa resolver o problema de projetar um subsistema de impressão e exibição de figuras em um sistema operacional. A impressão e exibição podem ser feitas em baixa e alta resolução através de *drivers*, para as duas resoluções, tanto para impressão quanto exibição.

Driver de	**Para máquinas de baixa capacidade**	**Para máquinas de alta capacidade**
Exibição	DEBR - driver de exibição de baixa resolução	DEAR - driver de exibição de alta resolução
Impressão	DIBR - driver de impressão de baixa resolução	DIAR - driver de impressão de baixa resolução

Inicialmente ele propôs a seguinte solução

```
class ControleAp{
        void desenhar(){
                switch(RESOLUÇÂO)
                        case BAIXA:
                                use DEBR:
                        case ALTA:
                                use DEAR
        }

        void imprimir(){
                switch(RESOLUÇÂO)
                        case BAIXA:
                                use DEBR:
                        case ALTA:
                                use DEAR
        }
}
```

Discuta as fragilidades desta abordagem. O que ocorreria se ele tivesse que tratar a escolha de *drivers* para uma configuração intermediária. Ajude o projetista a fornecer uma melhor solução para criação e uso das instâncias dos dois tipos de drivers levando em consideração a especificação e configuração do hardware no qual o sistema operacional irá executar.

Dica:usar o padrão Abstract Factory

4) Explique o padrão de projeto utilizado na hierarquia de classes para controle de fluxos (streams) de entrada e saída da linguagem java.

Dica: padrão Decorator

5) Diga dois mecanismos para adição de novos comportamentos a classes/objetos. Quais as limitações de cada um deles.

Dica: Comparar herança com delegação.

6) Faça uma pesquisa sobre o padrão Visitor. Compare-o com o padrão Iterator.

7) Considere uma classe que descreve uma série temporal. Desenvolva um pequeno aplicativo utilizando o padrão de projeto adequado que exiba a série temporal em três diferentes janelas: (a) uma janela que exibe a curva da série no tempo, uma

janela que exibe a tabela de valores no tempo, (c) uma janela que exiba a média, desvio padrão de toda a série até o momento e de subintervalos de k segundos especificados pelo usuário.

Dica:usar o padrão composto MVC

8) Classifique os padrões vistos em aula em comportamentais, construtivos e de criação.

9) No padrão Template Method diferencie um método abstrato convencional de um método gancho (hook). Cite dois possíveis usos de métodos hook.

Métodos Hook possuem uma implementação default normalmente vazia. Eles são tipicamente utilizados para estender/modificar o algoritmo codificado no template method sem ter que modificar o seu código. Um uso comum do método hook é capturar algum estado da execução dos passos do template method e utilizar essa informação para controlar o fluxo de execução do algoritmo codificado, uma vez que ele pode receber parâmetro associados a esses estados. Os métodos abstratos convencionais não são muito distintos dos hooks em sua construção. É a utilização do hook que o caracteriza como tal. Um método abstrato apenas fornece uma interface de um passo do algoritmo mas não é usado, por exemplo, para controlar o fluxo do algoritmo ou o ser um ponto de acesso a estados do mesmo, como o faz um método hook.

10) Considere uma classe Data que descreve algum dado. Data contém uma variável membro nome, e uma lista de atributos opcionais. Os atributos opcionais podem ser novos elementos de Data ou simplesmente instâncias de uma classe Attribute que contém um par de váriaveis membro nome e valor. Um programador especificou em um esboço de código um método as classes Data e Attribute conforme abaixo:

```
public class Attribute {
        private String name;
        private String value;

        Attribute(){...}

        String toString(){
                return "<" + nome + ">\n" +  "\t" +valor + "<" + nome +
                "/>\n";
        }
}

public class Data{
        private List attributes;
        ...
        public String toString(){
```

```
                String s = "<" + nome + ">\n"
                for (Data data:attributes)
                        if (this instanceof Attribute){
                                Attribute attr = (Attribute)data;
                                s += attr.toString()
                        }
                        else{
                             s+=data.toString();
                        }
                s += "<" + nome + "/>\n";
                return s;
        }

        addAttribute(int index, Data data){
                attributes.add(data);
        }

        Data getAttribute(int index){
                attributes.get(index);
        }
}
```

Refatore o código para que:

a) Não seja necessário utilizar o instanceof nas classes especificadas e em seus clientes.
b) Diferentes formatações da string de saída possam ser criadas

Dica:usar o padrão Composite

Frameworks

Dica: estudar os slides e se possível os capítulos 1, 15 e 16 do livro.

11) Forneça uma definição para um framework.

12) Explique o conceito de inversão de controle.

13) Diferencie frameworks, bibliotecas e APIs.

14) Diferencie *whitebox frameworks*, *blackbox frameworks* e *graybox frameworks*.

15) O que são *hot-spots* e *frozen-spots*. Cite exemplos.

16) Explique os padrões de construção de frameworks baseados na metodologia de desenvolvimento baseado em *hot-spots*

17) Relacione frameworks e padrões de projeto.

18) Explique como a estrutura de construção baseada em *template methods* e *hooks* pode ser utilizada para construção de frameworks.