Lista de Exercícios

MC302 – Programação Orientada a Objetos Instituto de Computação Universidade Estadual de Campinas		Design Patterns 2011 André Santanchè

Questão 1

Um laboratório de pesquisa está automatizando seu processo de testes biológicos. Cada experimento envolve uma sequência de ações automáticas a ser aplicadas em uma cultura de microrganismos, a fim de verificar a reação dos mesmos. Exemplos de ações são: elevação da temperatura ambiente, aplicação de um produto químico etc.

Cada ação é aplicada automaticamente por um robô controlado por um objeto de software que implementa a interface:

```
public interface RoboAction {
    public void execute();
}
```

Cada robô executa uma única ação e não é necessário informar parâmetros da ação.

Escreva uma classe denominada Experimento cujos objetos são capazes de aplicar um teste envolvendo uma sequência de ações, que podem ser realizadas por diferentes robôs. Cada objeto Experimento deve guardar uma lista com a sequência de ações a ser executada (cada ação é executada por um único objeto robô o qual ele guarda a referência) e deve implementar operações que permitam adicionar ações (sempre no final da sequência) e executar o teste (na sequência em que as ações foram cadastradas).

Depois de ter resolvido o exercício, leia sobre o design pattern Command (http://en.wikipedia.org/wiki/Command_pattern) e analise qual a relação com o código que você implementou.

Expansão:

Escreva um método extra denominado clone(), que fará parte da classe Experimento que é capaz de clonar um experimento completo.

Questão 2

O laboratório resolveu criar um processo automático para descobrir experimentos, que são mais eficientes na destruição de um dado tipo de microrganismo. Como as possibilidades de combinação são infinitas, decidiu-se usar um método baseado na programação genética.

Considere que existe uma classe ExperimentoMonitorado representando um experimento com funcionalidades extra de monitoramento. Esta classe é herdeira de Experimento e acrescenta o método contaMicro(), que não recebe parâmetros e retorna a contagem de microrganismos. Este método pode ser aplicado antes e depois da execução do experimento, de modo a verificar sua efetividade.

Implemente o processo em duas etapas:

Crie uma variante do método clone chamado cloneMutant que produz mutantes. Este método ao realizar a clonagem aplica uma única mutação na cópia que pode ser:

- um par de operações trocadas
- uma operação ausente
- uma operação duplicada

A posição da operação que sofre mutação também deve ser aleatória.

Crie um método estático (apenas o método sem uma classe) que receba dois parâmetros: um objeto da classe Experimento (base) e o número de ciclos que será usado para encontrar o melhor experimento. Este método deve atuar da seguinte maneira:

(1) cria uma lista de melhores experimentos com 5 posições (inicialmente vazia)

(2) adiciona o experimento base (recebido como parâmetro) na lista dos melhores

(3) percorre a lista dos melhores clonando (com mutação) cada um de seus componentes

(4) aplica todos os experimentos disponíveis (os da lista dos melhores e os clones)

(5) seleciona os 5 melhores resultados (se houver) e os coloca na lista dos melhores

(6) conta um ciclo e retorna ao passo (3) se não tiver completado o número de ciclos

Questão 3

Dada uma classe abstrata (já implementada) denominada `Noticiario` cuja função é enviar notícias para assinantes. Cada objeto desta classe fica monitorando uma fonte de notícias e quando surge uma nova notícia chama um método abstrato da própria classe com a seguinte assinatura:

```
public abstract void notificaNoticia(String textoNoticia, int dia, int mes, String topico);
```

Os parâmetros indicam o texto da notícia, seu dia e mês e o tópico a que se refere respectivamente. Tópico é uma string usada para classificar as notícias por tema. Exemplos de tópico: lançamento de livro, greve, inovação tecnológica etc.

Considerando que qualquer objeto consumidor de notícias implemente a interface:

```
public interface ConsomeNoticia {
    public void notificaNoticia(String textoNoticia, int dia, int mes, String topico);
}
```

Escreva uma classe herdeira de `Noticiario` chamada `NoticiarioAssina` que implemente o *pattern Observer* para divulgar suas notícias.

Questão 4

Implemente as classes de dois possíveis tipos de objeto que consomem esta notícia (implementam `ConsomeNoticia`): agregadores e publicadores.

Um objeto publicador publica cada notícia recebida (neste exercício, imprime no console).

Um objeto agregador concatena várias notícias recebidas e as envia para assinantes, implementando o mesmo *pattern* do `NoticiarioAssina`. Ele pode ser de dois tipos: agregador por tópico ou agregador por mês.

O agregador por tópico escolhe um tópico para agregar (informado no construtor) e concatena sempre dez notícias do mesmo tópico. Cada vez que consegue agregar dez notícias, ele envia a string concatenada para seus assinantes com o dia e o mês da última notícia.

O agregador por mês concatena todas as notícias do mesmo mês e, apenas quando chega uma notícia de um mês diferente, envia a string concatenada para seus assinantes com o dia zero e o tópico “mensal”.

Expansão:

Otimize o processo do *pattern Observer* para que objetos consumidores (implementam `ConsomeNoticia`) possam assinar apenas o tópico de notícia que estão interessados. Deste modo, não receberão a notificação dos outros tópicos.

Questão 5

Faça uma aplicação que concatene agregadores em sequência, ligados a um objeto `NoticiaAssina`, com um publicador no final. Uma concatenação interessante envolve o filtrar um tópico e o agregá-lo mensalmente.