Prática de Orientação a Objetos (extra 1)

fabio@engenhariadesoftware.inf.br

A classe `TestaAnimais` possui o método `main` e, neste, um laço (`for`) através do qual todos os sons produzidos por um *array* de `Animal` são enviados para a saída padrão. Abaixo segue um conjunto de passos que irá alterar, de forma significativa, a forma como este mesmo efeito é obtido. Ao final dos passos, contudo, espera-se que a abordagem empregada possa ser utilizada em outros contextos com o propósito de fragmentar uma operação sem que suas partes sejam dependentes entre si.

---

1. Crie um método na classe `TestaAnimais` para realizar esta operação, por exemplo, `void ExibeSonsAnimais(Animal[])`. Nesta nova forma, observe que o método `main` está mais "limpo" e faz uso de funcionalidade que provavelmente será reutilizada em outras partes de uma aplicação, o que não era possível com a versão anterior.
2. O método criado no item anterior serve a um propósito específico, contudo, realiza uma operação que pode ser fatorada, pois provavelmente é muito empregada: percorrer uma seqüência de instâncias de `Animal` e realizar alguma operação com cada um deles. No caso acima, um *array* é percorrido e, para cada instância, a mensagem `som()` é enviada e o retorno exibido na saída padrão. Podemos fragmentar a funcionalidade deste método em duas: percorrer o *array* e executar alguma operação com cada instância. Neste exercício a funcionalidade oferecida por `ExibeSonsAnimais` deverá ser substituída por método que realiza uma operação fornecida como argumento sobre todas as instâncias contidas no *array* também fornecido como argumento. A assinatura deste método deve ser: `void executaOperacao(Animal[] as, Operacao op)`. Nesta assinatura se observa a interface `Operacao`. Esta interface deve declarar uma única operação: `void executa(Object obj)`. O argumento é uma forma de passar para qualquer implementação desta classe o objeto sobre o qual a operação deverá ser executada. O que o método `executaOperacao` deve realizar é percorrer o *array* e, para cada um deles, enviar a mensagem `executa`. Ou seja, algo como

```
for (Animal a : animais) {
 op.executa(a);
}
```

   O resultado da execução do código acima deve produzir o mesmo efeito que já era obtido com as versões anteriores e, para tal, é necessário uma implementação da interface Operacao. Esta implementação é trivial, conforme ilustrado abaixo.

```
public class ExibeSomAnimal implements Operacao {
    public void executa(Object obj) {
        System.out.println(((Animal)obj).som());
    }
}
```

3. O código produzido acima é mais provável de ser reutilizado do que aquele das versões anteriores, embora ofereça a mesma funcionalidade. Para comprovar, crie o método `int idadeMedia()` na classe `Animal`. Este método deve retornar a idade média de vida do animal em questão. (Não me pergunte, não tenho a menor idéia de quantos anos vive em média um macaco, porco e outros animais de estimação.) Naturalmente você terá que acrescentar métodos `set/get` e, provavelmente, alterar o construtor da classe `Animal` ou adicionar um outro construtor. A disponibilidade deste método com a proposta de solução fornecida acima, permite, por exemplo, a identificação da soma de todas as idades médias de um conjunto de animais sem que uma nova sentença `for` seja criada. Crie código necessário para comprovar este fato.