Projeto e Construção de Sistemas

Módulo 6 – Testes de Unidade e Controle de Versões

Lista de Exercícios

Testes de Unidade

Refaça a classe de Teste referente aos programas Comprimento_01 e Comprimento_02 da lista de exercícios do Módulo 3 (Criação de Classes), de forma que ela passe a utilizar o framework JUnit.

Para isso, algumas modificações adicionais são solicitadas:

a) Alteração nas classes Comprimento:

No exercício da lista 3, eram pedidas duas implementações diferentes para uma classe Comprimento: uma que armazenava a informação de comprimento em três atributos (metros, centímetros e milímetros) e outra que armazenava a informação de comprimento em apenas um atributo (valor total em milímetros).

Neste exercício, você vai definir uma interface IComprimento com todas as operações presentes nas duas implementações. As duas classes de implementação Comprimento_Impl01 e Comprimento_Impl02 implementarão esta mesma interface (Comprimento).
Portanto teremos:
class Comprimento_Impl01 implements Comprimento (implementação com 3 atributos).
class Comprimento_Impl02 implements Comprimento (implementação com 1 atributo).

Defina estas duas classes mais a interface Comprimento no mesmo pacote.

b) Criação das classes de Teste usando o JUnit

Defina a classe ComprimentoTest a partir do JUnit, de forma que ela seja capaz de testar as duas implementações com uma única implementação. Para isso, o código de teste deve utilizar apenas a interface IComprimento. Esta classe ComprimentoTest será abstrata, pois deverá ter operações abstratas que retornem a instância de IComprimento a ser testada. Essas operações serão implementadas em duas subclasses de teste adicionais:
Comprimento_Impl01Test e Comprimento_Impl02Test.

Essas duas classes herdarão de ComprimentoTest e a única implementação presente em ambas é a instanciação do objeto IComprimento a ser testado. Como temos três formas de criar um objeto comprimento (três construtores – itens a, b, c da lista 3), deveremos ter três operações abstratas em ComprimentoTest:

```
public IComprimento getInstanciaComprimento(int metros, int cm, int mm)
public IComprimento getInstanciaComprimento(int mm)
public IComprimento getInstanciaComprimento(int cm, int mm)
```

Na classe Comprimento_Impl01Test, estas operações serão implementadas da seguinte forma:

```
public IComprimento getInstanciaComprimento(int metros, int cm, int mm) {
        return new Comprimento_Impl01(metros, cm, mm);
}
public IComprimento getInstanciaComprimento(int mm) {
        return  new Comprimento_Impl01(mm);
}
public IComprimento getInstanciaComprimento(int cm, int mm)  {
        return  new Comprimento_Impl01(cm, mm);
}
```

De forma análoga, a classe Comprimento_Impl02Test terá essas mesmas operações implementadas.

Desta forma, todo o código de teste poderá ser expresso na superclasse Comprimento_Test, sendo que toda vez que um método de teste precisar de uma instância de comprimento, deverá chamar a operação getInstanciaComprimento com os parâmetros esperados.

c) Criação do suíte de testes

Agora que as duas classes de teste foram criadas (Comprimento_Impl01Test e Comprimento_Impl02Test), crie uma classe para o suite de testes que permita a execução em lote dos testes definidos nas duas classes. Crie, por exemplo, uma classe AllTestsComprimento que implemente um suíte de testes.

Lembre-se: ela deve implementar uma única operação:
`public static Test suite() { ... }`

Dica: usar o método addTestSuite(<nome da classe>.class) para adicionar uma classe de teste a um suite de testes.

d) Criação do repositório

Crie um repositório subversion no assembla.com, conforme mostrado em aula e nos slides, e coloque toda a implementação (incluindo as classes de teste) nesse repositório. Não se esqueça de me adicionar como usuário desse repositório. A entrega do trabalho deverá ser feita através desse repositório (basta me adicionar como usuário, que eu vou pegar a versão publicada até a data/hora limite de entrega).