Procedimento Experimental

Para que os resultados sejam válidos e sob as mesmas condições para todos os participantes do experimento, é MUITO importante seguir os procedimentos do experimento de forma rígida.

1. Preparação

Para se preparar para o experimento, siga os seguintes passos:

a) Leia o conteúdo do arquivo “Documentação Experimento.pdf”. Esse arquivo possui os principais conceitos e as funcionalidades dos frameworks que serão utilizados no experimento. Ele poderá ser consultado durante a execução das tarefas, então não é preciso decorar suas informações, porém é importante ler seu conteúdo antes do experimento.

b) Renomeie o arquivo “ExperimentoEsfingeAOM.eclipse” para “ExperimentoEsfingeAOM.zip”. Descompacte o conteúdo do arquivo “ExperimentoEsfingeAOM.zip” em um diretório. Na ferramenta Eclipse, escolha “File” > “Import” > “Existing projects into workspace”, e em seguida selecione o diretório onde o arquivo foi descompactado. O projeto “ExperimentoEsfingeAOM” aparecerá na lista. Escolha-o e confirme.

c) O projeto já é para estar com todas as bibliotecas configuradas e estão incluídas no próprio diretório. Caso apareça algum problema de configuração de máquina virtual ou de JDK, acerte as configurações do projeto clicando com o botão direito no projeto e escolhendo “Properties”. Em “Java Compiler” veja se você possui um compilador compatível com o que está configurado e, caso contrário, altere para um presente an sua máquina. Em “Java Build Path”, verifique em “Libraries” se existe algum problema com o “JRE System Library”, e, em caso positivo, exclua essa biblioteca e em “Add Library...” adicione a “JRE System Library” da JDK presente na sua máquina. É importante ressaltar que esse procedimento só é necessário caso o projeto importado apresente algum problema.

d) Execute os testes de unidade do projeto importado com o JUnit. É para serem executadas as classes de teste “TesteValidacaoPessoaTarefa1”, na qual somente o teste “validaPessoaBase()” deve passar, e “TesteValidacaoCobrancaTarefa2”, na qual somente o teste “validaCobrancaBase()” deve passar. Caso os testes tenham executado dessa forma sem erros de compilação, o ambiente está pronto para a execução do experimento.

e) Termine de ler esse documento, e tenha conhecimento de todo o procedimento do experimento antes de começar.

2. Execução das tarefas

O experimento consiste na execução de duas tarefas de mesma natureza, porém utilizando abordagens diferentes. Os participantes foram divididos em 2 grupos. Antes de prosseguir com o experimento verifique a que grupo você pertence.

Grupo 1: Tarefa 1 usando Abordagem 1; em seguida, Tarefa 2 usando Abordagem 2
Grupo 2: Tarefa 2 usando Abordagem 1; em seguida, Tarefa 1 usando Abordagem 2

O procedimento que deve ser feito para cada tarefa é o seguinte:

1. Reserve um tempo entre 1 e 2 horas em que possa executar a tarefa sem interrupções e busque um ambiente tranquilo onde consiga focar na tarefa.

2. Leia a especificação da tarefa e da abordagem que vai utilizar.
3. Abra a documentação fornecida (ou imprima) e deixe-a de forma de fácil consulta.
4. Abra o IDE Eclipse com o projeto e rode novamente a classe de testes da tarefa que irá executar. Abra as classes relacionadas a tarefa e se familiarize com elas (ainda sem altera-las).
5. Utilize um marcador de tempo e comece a marcar o tempo da tarefa nesse momento. A precisão da marcação do tempo pode ser em minutos.
6. Faça a tarefa pedida. Ela é considerada concluída quando todos os testes relacionados aquela tarefa estiverem passando.
7. Pare e registre o tempo no momento que você executar todos os testes com sucesso.

Observação A: caso existam interrupções inevitáveis durante a tarefa, pare o tempo e retome quando voltar. Registre quanto tempo você precisou interromper a sua tarefa (isso será perguntado depois no questionário).

Observação B: não é preciso fazer as duas tarefas de forma contínua. É possível fazer uma e em seguida a outra. O registro de tempo de cada uma é separado.

Observação C: é permitido qualquer tipo de consulta durante o experimento, como a livros e a sites da internet.

Após a execução das tarefas, observe que ainda existe uma parte final, onde o participante deve responder um questionário e enviar os arquivos de código.

Seguem as próximas sub-seções apresentam as tarefas que precisam ser feitas, assim como as abordagens.

2.1. Descrição Geral das Tarefas

A tarefa que deve ser realizada é a de criar a implementação da validação de dados para um tipo de entidade criado utilizando o framework Esfinge AOM Role Mapper. Em outras palavras, o que será implementado deve verificar se os dados de uma entidade daquele tipo estão de acordo com certas restrições. As classes a serem utilizadas em cada tarefa estão respectivamente nos diretórios “tarefa1” e “tarefa2”.

O código-fonte fornecido possui o código que cria a entidade com as propriedades especificadas em cada tarefa. O código também apresenta uma bateria de testes que verifica se a validação está correta para casos em que um dos valores não obedece alguma das restrições especificadas. A função de validação presente na versão atual do código retorna verdadeiro sempre, de forma que o único teste da bateria que passa é o que apresenta uma entidade com todos os seus dados corretos.

Quando todos os testes estiverem passando, a tarefa é considerada finalizada.

2.2. Tarefa 1

Deve ser validada a entidade “Pessoa” com as seguintes restrições:

- Propriedade “nome”: precisa ter o tamanho maior do que 5 e menor do que 50 caracteres;
- Propriedade “altura”: precisa ser maior que 0 e menor ou igual que 215 (recebe valor inteiro em centímetros);
- Propriedade “cpf”: precisa ter o formato “999.999.999-99”, ou seja, precisa seguir a

expressão regular “[0-9]{3}\\.[0-9]{3}\\.[0-9]{3}\\-[0-9]{2}” - não é necessário (mas não é proibido) validar o dígito verificador do CPF, somente sua máscara;
- Propriedade “email”: precisa possuir obrigatoriamente um caractere “@”;
- Propriedade “profissao”: não pode ter o valor nulo nem ser uma String vazia;
- Propriedade “nascimento”: precisa ser uma data no passado;
- Propriedade “login”: precisa ter mais que 7 caracteres.

As seguintes classes relacionadas com essa tarefa fazem parte do projeto:

- A classe “FabricaTiposTarefa1” possui um método que cria o tipo de entidade;
- A classe “ValidadorTarefa1” possui o método de validação que precisa ser implementado;
- A classe “TesteValidacaoPessoaTarefa1” possui os testes de validação das restrições em uma entidade.

2.3. Tarefa 2

Deve ser validada a entidade “Cobranca” com as seguintes restrições:

- Propriedade “empresa”: não pode ser uma String vazia ou um valor nulo
- Propriedade “cnpj”: precisa ter o formato “99.999.999/9999-99”, ou seja, precisa seguir a expressão regular “[0-9]{2}\\.[0-9]{3}\\.[0-9]{3}\\/[0-9]{4}\\-[0-9]{2}” - não é necessário (mas não é proibido) validar dígitos verificadores do CNPJ, somente sua máscara.
- Propriedade “destinatario”: precisa ser um e-mail com obrigatoriamente um caractere “@”
- Propriedade “valor”: propriedade de ponto flutuante que precisa ser maior que 0 e não pode ser maior que 5000
- Propriedade “previsao”: precisa ser uma data no futuro
- Propriedade “faturamento”: precisa ser uma data no passado
- Propriedade “quantidadeItens”: deve um valor igual a 1 ou maior

As seguintes classes relacionadas com essa tarefa fazem parte do projeto:

- A classe “FabricaTiposTarefa2” possui um método que cria o tipo de entidade;
- A classe “ValidadorTarefa2” possui o método de validação que precisa ser implementado;
- A classe “TesteValidacaoCobrancaTarefa2” possui os testes de validação das restrições em uma entidade.

2.4. Abordagem 1

Essa abordagem faz a validação das propriedades programaticamente. Siga os seguintes passos para implementa-la:

1. **No método de validação**, recupere o valor de uma das propriedades.
2. Verifique se valor atende as restrições especificadas.
3. Caso a restrição não seja atendida, o método deve retornar falso.
4. Repita o procedimento para todas as propriedades.
5. Caso todas as restrições sejam atendidas, o método de validação deve retornar verdadeiro.

Nessa abordagem você deve trabalhar no código apenas da classe que possui o método de validação. A classe de testes e a classe que cria o tipo de entidade NÃO devem ser alteradas.

2.5. Abordagem 2

Essa abordagem irá utilizar a funcionalidade do framework Esfinge AOM Role Mapper de criar um adapter Java Bean para entidade, e utilizar o framework Hibernate Validator para executar a lógica de validação no Java Bean gerado. Siga os seguintes passos para implementa-la:

1. **No método de validação**, obtenha o adapter Java Bean da entidade passada como parâmetro.
2. Crie um validador do Hibernate Validator e passe o adapter Java Bean como parâmetro para o método de validação
3. Retorne falso se for retornada alguma violação e retorne verdadeiro se nenhuma violação for encontrada.
4. **No método de criação do tipo da entidade**, adicione metadados que vão configurar as restrições de cada propriedade.

As anotações do Hibernate Validator que estão na documentação fornecida estão todos mapeados para metadados no arquivo "AnnotationMapping.json" existente no projeto. O nomes e propriedades das anotações foram mapeados sempre para metadados com o mesmo nome, mas com todas as letras minúsculas, como, por exemplo, "@NotEmpty" foi mapeado para "notempty".

Nessa abordagem você deve trabalhar no código da classe que possui o método de validação e na classe que cria o tipo de entidade, para adição dos metadados. A classe de testes NÃO deve ser alterada.

3. Questionário e Envio dos Resultados

Após realizar as duas tarefas, responda o questionário presente nesse link:

http://goo.gl/forms/JfwvEWaKXNptGeFv2

Em seguida, crie um arquivo .zip contendo apenas o conteúdo do diretório "src" da pasta do projeto do Eclipse. Nomeie esse arquivo da seguinte forma: "nome.sobrenome.grupox.zip". Um exemplo seria "eduardo.guerra.grupo2.zip". Envie esse arquivo em anexo para o e-mail guerraem@gmail.com com a tag [Experimento AOM] no assunto.

Muito obrigado pela sua participação!