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) Descompacte o conteúdo do arquivo "ExperimentoEsfingeMetadata.zip" em um diretório. Na ferramenta Eclipse, escolha "File" > "Import" > "Existing projects into workspace", e em seguida selecione o diretório em que se encontra o arquivo. O projeto "ExperimentoEsfingeMetadata" aparecerá na lista. Escolha-o e confirme.

b) 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.

c) Execute os testes de unidade do projeto importado com o JUnit. É para serem executadas as classes de teste "TesteValidacaoTarefa1" e "TesteValidacaoTarefa2", respectivamente, dos pacotes tarefa1 e tarefa2. Em ambas as classes de teste, é esperado que apenas um único caso de teste passe e todos os outros falhem. Caso os testes tenham executado sem erros de compilação, o ambiente está pronto para a execução do experimento.

d) 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 dois grupos, conforme a Figura 1. 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.

IMPORTANTE: O arquivo "Documentação Experimento.pdf" só deve ser lido antes do início das tarefas que façam o uso da Abordagem 2 - Esfinge$_{METADATA}$. Esse arquivo possui os principais conceitos e as funcionalidades do meta-framework que será utilizado no experimento. Ele poderá ser consultado durante a execução da tarefa, então não é preciso decorar suas informações, porém é importante ler seu conteúdo antes do experimento.

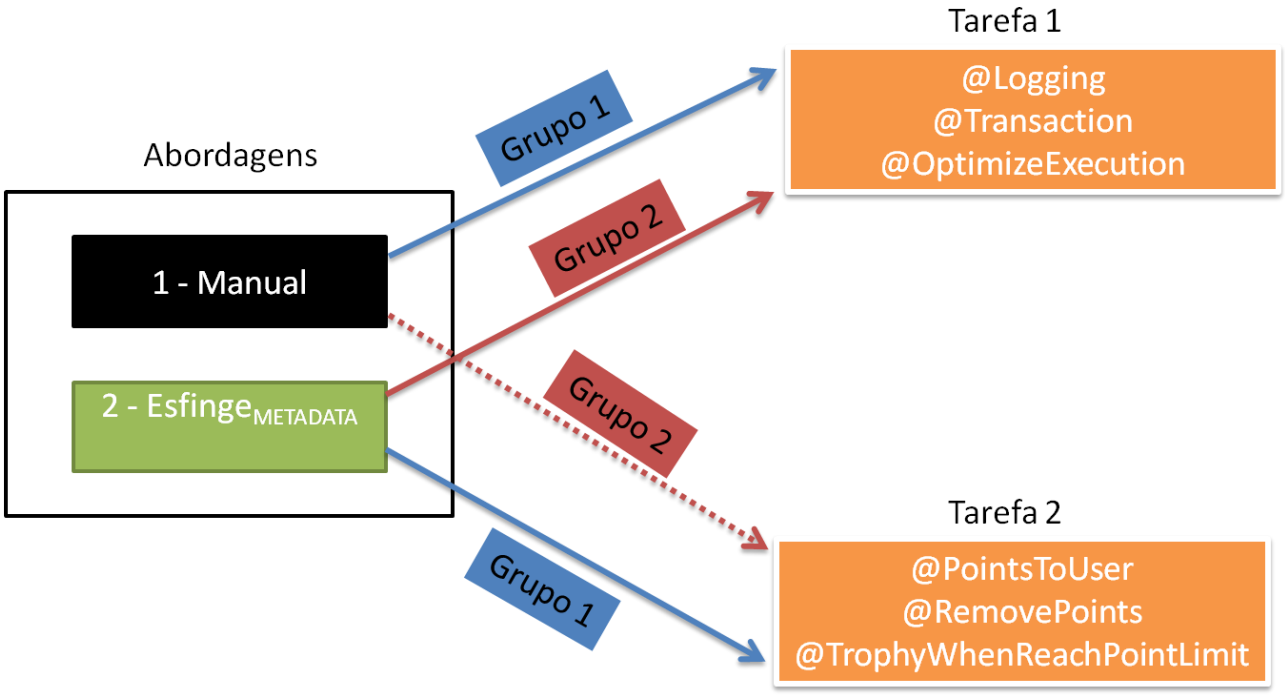

Figura 1 – Ilustração do Experimento.

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 tranqüilo 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 alterá-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 anotações utilizando duas abordagens distintas. Em outras palavras, o que será implementado deve verificar se a configuração dos metadados (anotações) estão de acordo com certas restrições. As classes a serem utilizadas em cada tarefa estão respectivamente nos pacotes "tarefa1" e "tarefa2".

O código-fonte fornecido possui as classes que devem ser validadas com os metadados já configurados e 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 (não implementada) presente na versão atual do código retorna verdadeiro ou falso, de forma que o único teste da bateria que passa é o que apresenta uma classe com todos os seus metadados corretos.

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

2.2. Tarefa 1

Devem ser validadas as anotações @Logging, @Transaction e @OptimizeExecution com as seguintes restrições:

- **@Logging:** Define um metadado ligado a um método que significa que sua execução deve ser registrada pelo mecanismo de *logging*. Se a anotação @Logging for adicionada em uma classe, todos os seus métodos são considerados anotados. Se ela for definida no pacote, todas as classes e os métodos das classes são considerados anotados. Restrição: para um método ser anotado com @Logging ele requer a anotação @Transaction ligada ao mesmo elemento.
- **@Transaction:** Define um metadado ligado a um método que significa que sua execução deve ser feita no contexto de uma transação. Se a anotação @Transaction for adicionada em uma classe, todos os seus métodos são considerados anotados. Se ela for definida no pacote, todas as classes e os métodos das classes são considerados anotados. Restrição: um método com a anotação @Transaction ligada a ele, não pode possuir a anotação @OptimizeExecution para o mesmo elemento.
- **@OptimizeExecution:** Define um metadado ligado a um método que significa que deve ser feita uma otimização durante sua execução. Se a anotação @OptimizeExecution for adicionada em uma classe, todos os seus métodos são considerados anotados. Se ela for definida no pacote, todas as classes e os métodos das classes são considerados anotados. Restrição: um método com a anotação @OptimizeExecution ligada a ele, não pode possuir a anotação @Transaction ligada a ele.

Os Exemplos 1 - 3 ilustram cenários válidos e inválidos de configuração das anotações supramencionadas.
No Exemplo 1, apresenta-se um cenário válido de configuração da anotação @Logging, definida no método doSomething(), e a anotação requerida (@Transaction) por ela, está configurada na classe Example – como se todos os seus métodos fossem anotados.

```
1	@Transaction
2	public class Example {
3
4		@Logging
5		public void doSomething() {
6
7		}
8	}
```

Exemplo 1: Configuração Válida de @Logging

No Exemplo 2, apresenta-se um cenário inválido de configuração da anotação @Logging, definida no método doSomething(), sem a presença da anotação @Transaction, requerida em sua configuração para o mesmo elemento do código.

```
1
2	public class Example {
3
4		@Logging
5		public void doSomething () {
6
7		}
8	}
```

Exemplo 2: Configuração Inválida de @Logging

No Exemplo 3, apresenta-se um cenário inválido de configuração da anotação @Logging, definida no método doSomething() junto com a anotação @OptimizeExecution e a anotação requerida (@Transaction) por ela na classe Example. Este é um cenário inválido pelo fato que @Logging não pode ser definida para o mesmo elemento do código com @OptimizeExecution.

```
1	@Transaction
2	public class Example {
3
4		@Logging @Optimize Execution
5		public void doSomething () {
6
7		}
8	}
```

Exemplo 3: Configuração Inválida de @Logging

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

- A classe “ValidadorTarefa1” possui o método de validação que precisa ser implementado;
- A classe “TesteValidacaoPessoaTarefa1” possui as classes com as anotações configuradas e os testes de validação das restrições das anotações. Esta classe NÃO deve ser alterada.

2.3. Tarefa 2

Devem ser validadas as anotações @TrophyWhenReachPointLimit, @PointsToUser e @RemovePoint com as seguintes restrições:

- **@TrophyWhenReachPointLimit:** Define um metadado ligado a um método que significa que sua execução deve atribuir um troféu ao usuário por ter atingido determinada pontuação.

Essa anotação pode ser definida dentro de outras anotações. Restrição: um método com a anotação @ TrophyWhenReachPointLimit liga a ele, requer a anotação @PointsToUser na configuração do mesmo elemento.

- **@PointsToUser:** Define um metadado ligado a um método que significa que sua execução deve atribuir pontos ao usuário. Essa anotação pode ser definida dentro de outras anotações. Restrição: um método anotado com @PointsToUser, não poderá ter a definição da anotação @RemovePoint para o mesmo elemento.

- **@RemovePoint:** Define um metadado ligado a um método que significa que sua execução deve retirar pontos do usuário. Essa anotação pode ser definida dentro de outras anotações. Restrição: um método anotado com @RemovePoint, não poderá ter a definição da anotação @PointsToUser para o mesmo elemento.

OBSERVAÇÃO: Apesar de não fazerem parte da especificação, na classe de testes [TesteValidacaoTarefa2()], serão utilizadas as anotações @ExtraPoints e @LosePoint, respectivamente, para o caso em que as anotações @PointsToUser e @RemovePoint estão definidas dentro de uma outra anotação.

Os Exemplos 4 - 6 ilustram cenários válidos e inválidos de configuração das anotações supramencionadas.

No Exemplo 4, apresenta-se um cenário válido de configuração da anotação @TrophyWhenReachPointLimit, definida no método doSomething(), junto com a anotação requerida (@PointsToUser) por ela, configurada dentro da anotação @ExtraPoints para o mesmo elemento do código.

```
1   @Retention(RetentionPolicy.RUNTIME)
2   @PointsToUser(value=1)
3   public @interface ExtraPoints {
4
5   }
```

```
1   public class Example {
2       @Extra Points
3       @ TrophyWhenReachPointLimit
4       public void doSomething() {
5
6       }
7   }
```

Exemplo 4: Configuração Válida de @TrophyWhenReachPointLimit

No Exemplo 5, apresenta-se um cenário inválido de configuração da anotação @TrophyWhenReachPointLimit, definida no método doSomething(), sem a presença da anotação @PointsToUser, requerida em sua configuração para o mesmo elemento do código.

```
1   public class Example {
2
3       @ TrophyWhenReachPointLimit
4       public void doSomething() {
5
6       }
```

```
7       }
```

Exemplo 5: Configuração Inválida de @TrophyWhenReachPointLimit

No Exemplo 6, apresenta-se um cenário inválido de configuração da anotação @PointsToUser, definida no método doSomething(), junto com a anotação @RemovePoint, proibida em sua configuração para o mesmo elemento do código.

```
1       public class Example {
2
3               @ PointsToUser  @ RemovePoint
4               public void doSomething() {
5
6               }
7       }
```

Exemplo 6: Configuração Inválida de @PointsToUser

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

- A classe “ValidadorTarefa2” possui o método de validação que precisa ser implementado;
- A classe “TesteValidacaoPessoaTarefa2” possui as classes com as anotações configuradas e os testes de validação das restrições das anotações. Esta classe NÃO deve ser alterada.

2.3. Abordagem 1

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

1. **No método de validação** chamado validarAnotacoesNaClasse(), recupere as anotações da classe passada como parâmetro.
2. Verifique se as anotações utilizadas na classe atendem as restrições de uso.
3. Caso a restrição não seja atendida, o método deve retornar falso.
4. Repita o procedimento para todas as anotações encontradas.
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$_{METADATA}$ para executar a lógica de validação dos metadados utilizados. Siga os seguintes passos para implementá-la:

1. Adicione as meta-anotações do Esfinge$_{METADATA}$ nas anotações a serem validadas, de acordo com suas restrições de uso.
2. Faça a importação das bibliotecas requeridas pelas meta-anotações.
3. **No método de validação** chamado validarAnotacoesNaClasse(), invoque o método de validação do Esfinge$_{METADATA}$, passando como parâmetro a classe à ser validada.
4. O meta-framework retornará uma exceção caso alguma restrição de uso das anotações da classe seja violada.

5. Deve ser retornado falso caso alguma violação na configuração das anotações seja encontrada, e verdadeiro caso contrário.

Nessa abordagem você deve trabalhar no código da classe que possui o método de validação e nas anotações que precisam ser validadas, adicionando metadados (meta-anotações) que serão utilizados pelo $\text{Esfinge}_{\text{METADATA}}$. 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:

https://goo.gl/forms/8ErW4enXZBSfEqbV2

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 EsfingeMetadata] no assunto.

Muito obrigado pela sua participação!