Gmail **Nascimento Luiz Wagner <nascimentolwtn@gmail.com>**

[CAP-389-3 Projeto Ágil de Software 2016] Video-aulas e próxima aula presencial (16/11)

1 message

Eduardo Guerra <guerraem@gmail.com> Wed, Nov 2, 2016 at 9:58 PM
To: Jo Choma <jh.choma@hotmail.com>, Phyllipe Lima <phyllipe_slf@yahoo.com.br>, raian@dpi.inpe.br, Antonio Dias <antoniodiasabc@gmail.com>, "Alexandre A. D'Avila de Oliveira" <alexandre.oliveira@cptec.inpe.br>, André Ivo <andre.ivo@gmail.com>, Cícero Alves junior <cicerocasj@gmail.com>, Fernando Miguel <fernando@origamiweb.com.br>, Leonardo Cherubini <cherubini18@gmail.com>, Nascimento Luiz Wagner <nascimentolwtn@gmail.com>, Max Gomes <maxgomes92@live.com>, Rafael Luckez <rluckez@outlook.com>, Ricardo Luiz dos Reis Silva <silva.ricardoluiz@gmail.com>, RODRIGO LUCAS TEIXEIRA BARBOSA <rodrigo.barbosa10@fatec.sp.gov.br>, "Rodrigo M. Mariano" <rodrigo.mariano@inpe.br>, Wendel Silvério <wendelsilverio@gmail.com>, giovanna botelho <giovannacbotelho@gmail.com>, Marco Nardes <marconardes@gmail.com>
Cc: Tiago Silva da Silva <silvadasilva@gmail.com>

Bom dia!

Essa semana teve na quarta-feira o feriado de finados e na próxima semana, tanto eu quanto o Tiago estaremos fora participando do Agile Brasil. A próxima aula presencial será somente no dia 16/11

Encaminho as video-aulas para essas próximas duas semanas:

Semana 02/11 - 09/11

Mock Frameworks (apresentando JMock)
Hands-on Mock Objects com JMock
TDD Step Patterns

Semana 09/11 - 16/11

Refactoring to Patterns
Hands-on Padrões TDD - parte 1
Hands-on Padrões TDD - parte 2

Exercício

Encaminho abaixo um exercício referente ao conteúdo da aula, que é para ser entregue até dia 16/11 a meia-noite. Deve ser encaminhado o que foi solicitado nos emails: guerraem@gmail.com e silvadasilva@gmail.com com o seguinte cabeçalho "[CAP-389-3 Projeto Ágil de Software 2016] Exercício 2 - $seunome". Entregas fora desse padrão não serão consideradas!

Criar, utilizando TDD, uma classe chamada CaixaEletronico, juntamente com a classe ContaCorrente, que possuem os requisitos abaixo:

- A classe CaixaEletronico possui os métodos logar(), sacar(), depositar() e saldo() e todas retornam uma String com a mensagem que será exibida na tela do caixa eletrônico.
- Existe uma classe chamada ContaCorrente que possui as informações da conta necessárias para executar as funcionalidades do CaixaEletronico. Essa classe faz parte da implementação e deve ser definida durante a sessão de TDD.
- As informações da classe ContaCorrente podem ser obtidas utilizando os métodos de uma interface chamada ServicoRemoto. Essa interface possui o método recuperarConta() que recupera uma conta baseada no seu número e o método persistirConta() que grava alterações, como uma mudança no saldo devido a um saque ou depósito. Não tem nenhuma implementação disponível da interface ServicoRemoto e deve ser utilizado um Mock Object para ela durante os testes.
- O método persistirConta() da interface ServicoRemoto deve ser chamado **apenas** no caso de ser feito algum saque ou depósito **com sucesso**.

- Ao executar o método saldo(), a mensagem retornada deve ser "O saldo é R$xx,xx" com o valor do saldo.
- Ao executar o método sacar(), e a execução for com sucesso, deve retornar a mensagem "Retire seu dinheiro". Se o valor sacado for maior que o saldo da conta, a classe CaixaEletronico deve retornar uma

String dizendo "Saldo insuficiente".
- Ao executar o método depositar(), e a execução for com sucesso, deve retornar a mensagem "Depósito recebido com sucesso"
- Ao executar o método login(), e a execução for com sucesso, deve retornar a mensagem "Usuário Autenticado". Caso falhe, deve retornar "Não foi possível autenticar o usuário"

- Existe uma interface chamada Hardware que possui os métodos pegarNumeroDaContaCartao() para ler o número da conta do cartão para o login (retorna uma String com o número da conta), entregarDinheiro() que entrega o dinheiro no caso do saque (retorna void) e lerEnvelope() que recebe o envelope com dinheiro na operação de depósito (retorna void). Não tem nenhuma implementação disponível da interface Hardware e deve ser utilizado um Mock Object para ela durante os testes.
- Todos os metodos da interface Hardware podem lançar uma exceção dizendo que houve uma falha de funcionamento do hardware.

Deve-se criar testes também para os casos de falha, principalmente na classe Hardware que pode falhar a qualquer momento devido a um mau funcionamento.

Lembre-se de usar o TDD e ir incrementando as funcionalidades aos poucos.

Você deve entregar o código final, incluindo os testes e os mock objects criados. Coloque todo código relativo a teste em uma pasta separada.

Saudações!

º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø
Eduardo Guerra - Associate Researcher
Instituto Nacional de Pesquisas Espaciais - INPE
Laboratório Associado de Computação e Matemática Aplicada - LAC
Redes sociais: @emguerra e YouTube
Instrutor Coursera: Programa de cursos integrados Programação Java e Desenvolvimento Ágil
Autor: Design Patterns com Java: Projeto orientado a objetos guiado por padrões
Componentes Reutilizáveis em Java com Reflexão e Anotações
º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø,¸¸,Ø¤º°`°º¤Ø