Utilizando Testes Unitários

Motivação

- Todos os programadores sabem que devem testar seu código
- Quanto mais curto o prazo menos testes são realizados
- Necessidade de testar cada método separadamente
- Centralizar o código de testes
- Regras já definidas para apresentação de erros
- Possibilidade de automatização

Porque testar?

- Como se pode ter certeza de que o que foi produzido funciona?
- Será que outras pessoas confiam no seu código tanto quanto você?
- Como passar confiança para as outras pessoas a respeito do código que você produziu?
- Você “coloca sua mão no fogo” pelo seu código?

Porque testar?

- O que o teste pode garantir?
- Porque vou perder tempo fazendo testes se já sei qual é a solução?
- Vou ter que parar para fazer testes!!! Isso vai reduzir minha produtividade pela metade?

Testes de Unidade

- Testam a aplicação em seus menores componentes, isoladamente
- Testam unidades lógicas
  - Métodos
  - Objetos
- Maior número de erros detectados
- Erros mais fáceis de corrigir
- Devem ser executados continuamente

Testes de Integração

- Testam como uma coleção de unidades interage entre si ou com o ambiente onde executam

- Executados continuamente (caso as unidades em desenvolvimento dependam de outras)

Testes Funcionais

- Ponto de vista do usuário
- Testam casos de uso
- Validam a interface com o usuário, as operações requisitadas, etc.
- São menos estáveis do que os outros tipos

O que se pode testar?

- 

-
  - E da educação de cada um

Junit

- O **JUnit** é um *framework open-source*, criado por Eric Gamma e Kent Beck, com suporte à criação de testes automatizados na linguagem de programação Java.

Escrevendo um teste

Assuma uma classe Conta ☺
Para testar a classe definimos uma classe ContaTest

```
import test.framework.TestCase;

public class ContaTest extends TestCase {
    public ContaTest(String testCase) {
        super(testCase);
    }
}
```

Escrevendo um teste

- Para testar o método getSaldo() definimos o método testGetSaldo() na classe ContaTest;

```
public void testGetSaldo() {
    Conta c = new Conta(1, 100);
    assertTrue(c.getSaldo() == 100);
}
```

- Para cada método da classe Conta é definido um método na classe ContaTest

Principais asserções

- `assertEquals(objEsperado, objRecebido);`
- `assertTrue(expBooleana);`
- `assertNull(obj);`
- `assertNotNull(obj)`
- `assertSame(obj1, obj2);`
- `fail(mensagem);`

Junit funcionando

- O TestRunner recebe uma subclasse de junit.framework.TestCase e executa o método run(Test)
  - Obtém dados de TestSuite - suite()
  - TestSuite usa Java Reflection para descobrir os métodos de teste

Test Suite

- Composição de testes

```
public static Test suite() {
        TestSuite suite = new TestSuite();
        suite.addTest(new ContaTest("testGetSaldo"));
        suite.addTest(new ContaTest("testCreditar"));
        return suite;
}
```

- Um TestSuite é usado pelo TestRunner para saber quais métodos devem ser executados como testes.

Test Suite

- Para executar mais de um teste de uma vez e/ou reusar testes de outras classes

```
public static Test suite() {
    TestSuite suite = new TestSuite();
    suite.addTest(new ContaTest("testSaldoConta"));
    suite.addTest(new ContaTest("testCredito"));

    suite.addTest(Poupanca.suite());
    return suite;
}
```

Como funcionam as asserções

- As asserções causam falhas quando os resultados não estiverem corretos
  - test.framework.AssertionFailedError
- Cada instância de TestCase será utilizada para executar um dos métodos de teste
  - As alterações que ele fizer ao estado do objeto não afetarão as demais classes

O que é TDD?

- Test Driven Development
  - Desenvolvimento orientado a testes
    - Mais uma invenção do louco do Kent Beck

- Elicitar requisitos
- Desenvolver os testes
- (Refinar os requisitos através dos testes)
- Implementar a solução mínima

Como se faz TDD?

- Um caso de uso é escolhido
- Desenvolve-se o teste para o mesmo
- O teste não vai sequer compilar
  - Se compilar, pare de programar e vá tomar um café
- Implemente os stubs
- Rode o teste
- O teste não vai passar
  - Se passar, pare de programar e vá tomar café
- Desenvolva a solução até que os testes passem
- Quando os testes passarem, pare de programar
  - E vá beber alguma coisa... talvez um café

Perca de produtividade

- Se vou parar para implementar os testes para testar alguma coisa que já sei o que é, então vou codificar dobrado... e pelo mesmo salário?!
- Ou melhor... minha produtividade vai cair, e vão diminuir meu salário

- Não!

- Gerar um bug é fácil
- Difícil é encontrá-lo e consertá-lo
  - E não errar mais na mesma coisa

A garantia dos testes

- Que você não vai cair no mesmo erro duas vezes

- Encontrei um bug... vou consertá-lo
- NÃO!!!

- Desenvolva um caso de teste para falhar exatamente sobre o bug
- Rode o teste
- Conserte o bug
- Rode o teste
- Salve o teste

Agora... Vamos brincar um pouco

- Preciso de um sistema que:
- O nome da classe pode ser Cadastro.java
  - Possa adicionar, remover e atualizar(a senha) um usuário
    - Métodos: adicionar(Usuario), remover(Usuario), atualizar (String, String)
  - Esse usuário deve ter login e uma senha
    - Duas Strings
  - A senha não deve ter menos que 4 caracteres
  - A senha não pode ser igual ao login
  - A senha não pode ter espaços em branco no início nem no final.
  - Caso a senha fornecida não seja válida, lançar um IllegalArgumentException
  - Não posso ter usuários repedidos (com mesmo login)
  - Deve poder me retornar a lista/estrutura de todos os usuários inseridos
    - Método getAll() que retorna o vector, array ou lista

Repostas às perguntas iniciais

- Sem testes não dá
- Com certeza não
- Garantindo que foram corretamente testados
- Nem se os testes passarem eu coloco. ;)

Respostas às perguntas inciais

- O que o teste pode garantir? (um dos principais benefícios)

  Que você não vai errar novamente no mesmo lugar

- Porque vou perder tempo fazendo testes se já sei qual é a solução?

  Porque você não pode garantir que vai codificar o que realmente quer

- Vou ter que parar para fazer testes!!! Isso vai reduzir minha produtividade pela metade?

Sua produtividade diminui quando tem que parar de produzir para procurar um bug

Fecomércio PR
Sesc Senac IFPD
Senac

A pergunta que não quer calar...

- Mas, teste é código, então pode estar errado. Quem testa o teste?!

- Alguém tem alguma outra dúvida (que não seja igual a citada acima)?