Curso | Instituto Tecnológico de Aeronáutica

TDD – Desenvolvimento de software guiado por testes

Semana 2 – Tarefa avaliada por colega: Refatoração do SAB

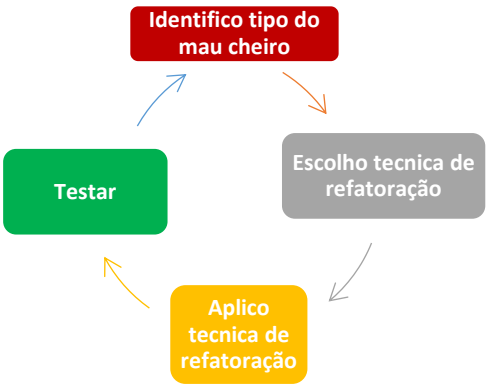
Aluno: Vinicius Oruam Rodrigues

Examine o método abaixo registraUsuario (String) e faça o seguinte:
a) Identifique uma lista de maus cheiros que você encontra no código, relacionando cada um deles com o correspondente tipo de mau cheiro exercitado nesta parte do curso: [mau cheiro no código (trecho do código) /tipo de mau cheiro (de acordo com Fowler, pode estar em português)].
b) Realize o Ciclo de Refatoração apresentado, eliminando cada um dos maus cheiros encontrado no código do método, considerando apenas os tipos de mau cheiro exercitados nesta Semana 2 do curso.
c) Entregue um documento em que você apresenta o seguinte:
A) Código anterior do método registraUsuario (String), antes de iniciar o Ciclo de Refatoração.
B) Imagem: Imagem da execução bem-sucedida (verde) no Eclipse ou outro ambiente Java, comprovando que código atual do SAB, incluindo o método registraUsuario (String) está funcionando direito (pelo menos de acordo com a bateria de testes atual).
C) Ciclo de Refatoração até a Lista de Maus Cheiros ficar vazia, apresentando 5 coisas para cada refatoração realizada no Ciclo de Refatoração:
Antes: O código Antes da refatoração, com o trecho a ser refatorado com as letras coloridas ou com fundo amarelo
Tipo Mau Cheiro/Técnica de Refatoração: Indique o tipo do mau cheiro identificado no código em 1) e a técnica de refatoração empregada, ambos de acordo com Fowler e podendo estar em português!
Depois: O código Depois da refatoração, com o trecho refatorado com as letras coloridas ou com fundo verde
Imagem: Imagem da execução bem-sucedida (verde) no Eclipse ou outro ambiente Java, comprovando que a refatoração foi feita a contento!
Lista de maus cheiros: Atualize a lista, eliminando o mau cheiro que deu origem à refatoração deste ciclo
D) Código Depois final do método abaixo registraUsuario (String), sem letras ou fundo coloridos!
Procure apresentar o código sempre bem apresentado, de acordo com boas práticas de apresentação/formatação de código Java. Pode usar, por exemplo, o Source/Format do Eclipse ou equivalente do seu ambiente Java.
A bateria de testes atual não pode ser modificada de forma alguma durante o Ciclo de Refatoração!

1. **Tipos mais comuns de mau cheiro:**

• **Nome inadequado**	• **Comandos If e Swith**
• **Código duplicado**	• **Inveja de Característica**
• **Método Grande**	• **Intimidade Imprópria**
• **Classe Grande**	• **Comentários**

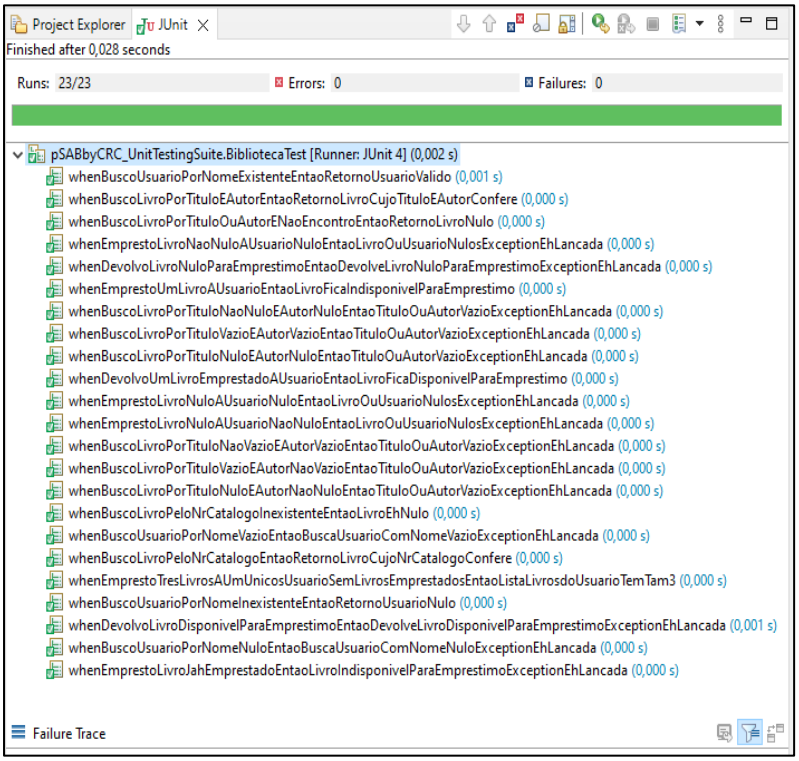
2. **Ciclo de Refatoração:**



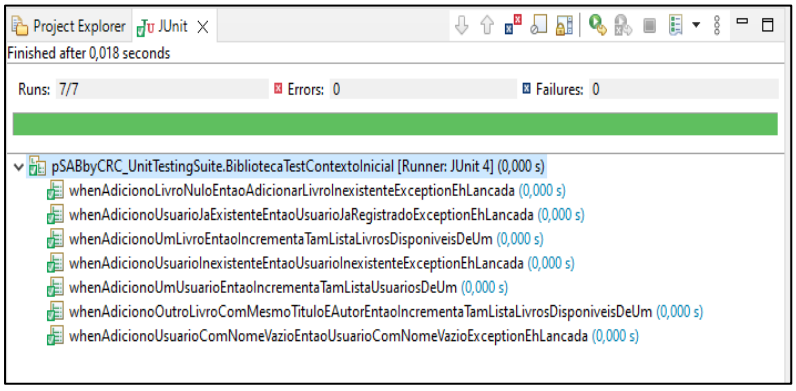
Código anterior do método registraUsuario(String), antes de iniciar o Ciclo de Refatoração.

```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome != null) {
29            if (!nome.isEmpty()) {
30                Usuario usuario = new Usuario(nome);
31                if (!_usuarios.contains(usuario)) {
32                    _usuarios.add(usuario);
33                } else
34                    throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
35                            + nome + "\"! Use outro nome!");
36            } else
37                throw new UsuarioComNomeVazioException("--->N‹o pode registrar usuario com nome vazio!");
38        } else
39            throw new UsuarioInexistenteException("--->N‹o pode registrar usuario inexistente!");
40    }
```

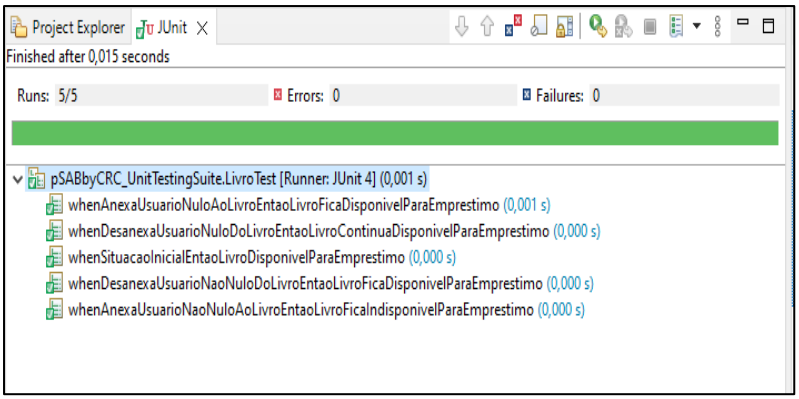
Execução bem-sucedida (verde), do funcionamento do código de produção e seu ambiente de teste de acordo com a bateria de testes atual.



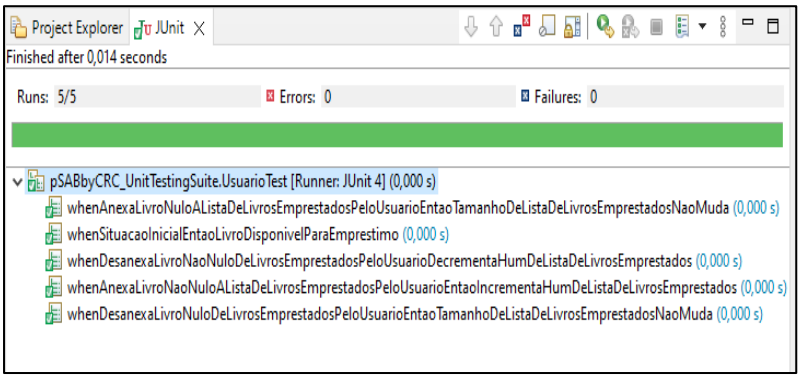
BibliotecaTest.java



BibliotecaTestContextoInicial.java



LivroTest.java



UsuarioTest.java

1° Trecho de código: **linha 28** e **linha 39**

Tipo de mau cheiro: Código escrito de forma negativa, isso é mais difícil do cérebro humano entender.

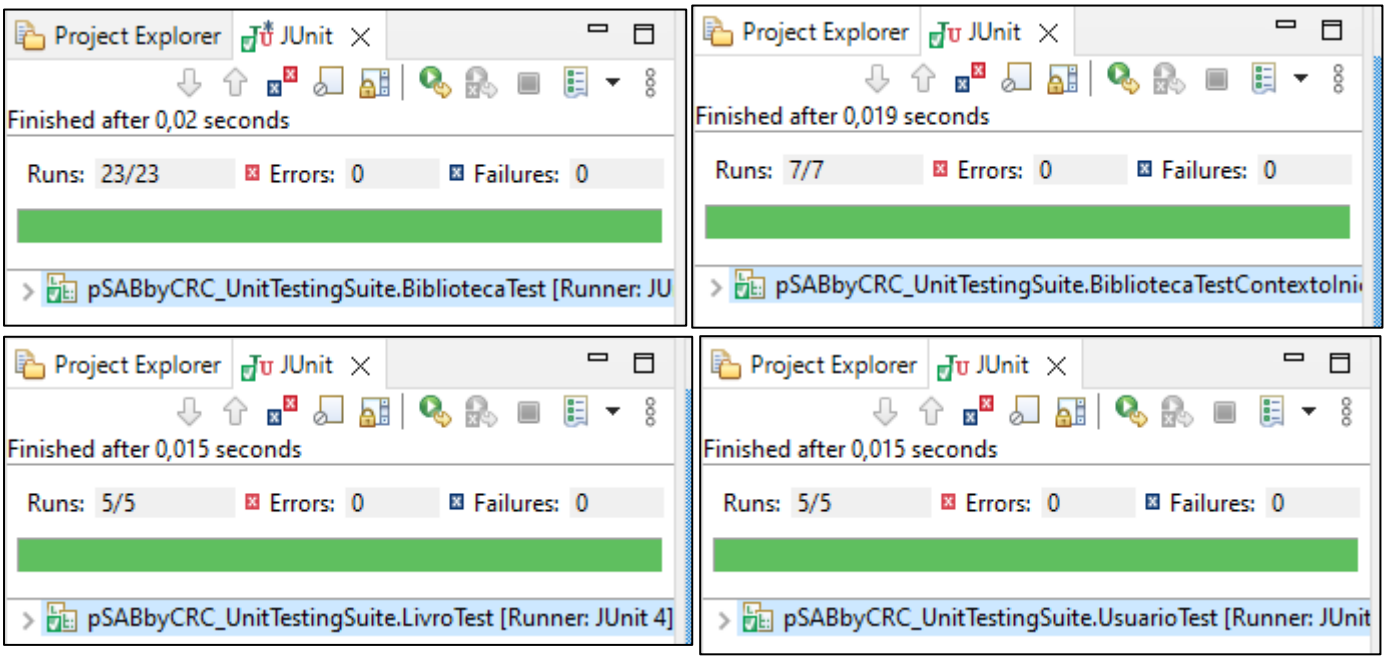
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
                     Exception {
28          if (nome != null) {
                if (!nome.isEmpty()) {
30                  Usuario usuario = new Usuario(nome);
31                  if (!_usuarios.contains(usuario)) {
32                      _usuarios.add(usuario);
33                  } else
34                      throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
35                              + nome + "\"! Use outro nome!");
36              } else
37                  throw new UsuarioComNomeVazioException("--->N‹o pode registrar usuario com nome vazio!");
39              throw new UsuarioInexistenteException("--->N‹o pode registrar usuario inexistente!");
```

Refatoração

```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
28          if (nome == null) throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30              Usuario usuario = new Usuario(nome);
31              if (!_usuarios.contains(usuario)) {
32                  _usuarios.add(usuario);
33              } else
34                  throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
35                          + nome + "\"! Use outro nome!");
36          } else
37              throw new UsuarioComNomeVazioException("--->N‹o pode registrar usuario com nome vazio!");
38    }
```

O código começa a ficar mais fácil de entender e também fica mais limpo.

Aplicação do teste



2° Trecho de código: **linha 29** e **linha 37**

Tipo de mau cheiro: Código escrito de forma negativa, isso é mais difícil do cérebro humano entender.

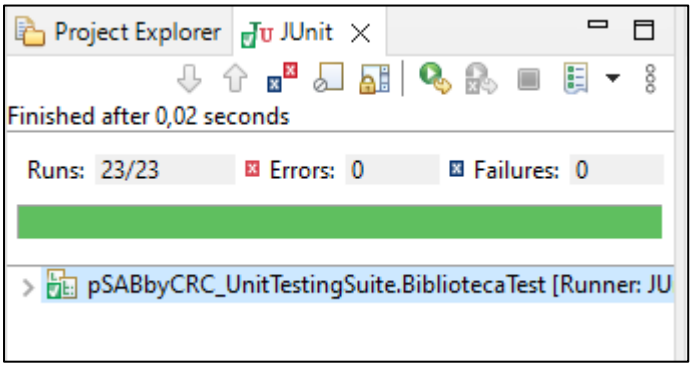
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
29            if (!nome.isEmpty()) {
31                if (!_usuarios.contains(usuario)) {
32                    _usuarios.add(usuario);
33                } else
34                    throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
35                            + nome + "\"! Use outro nome!");
37                throw new UsuarioComNomeVazioException("--->N‹o pode registrar usuario com nome vazio!");
```

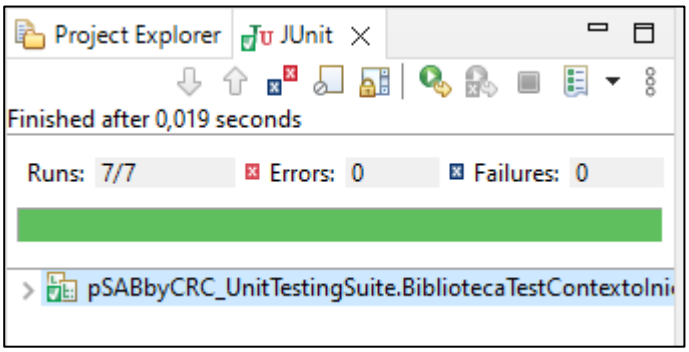
Refatoração

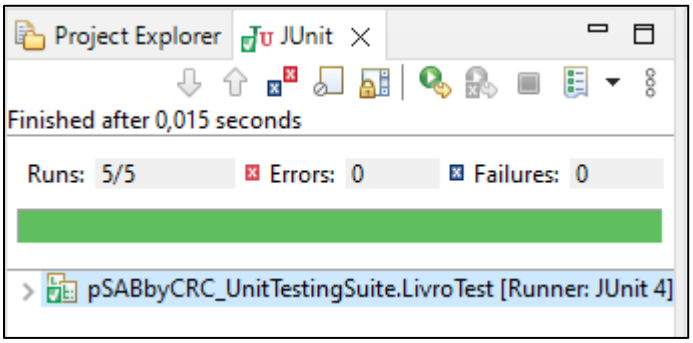
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
30        if (!nome.isEmpty())
31            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
33                if (!_usuarios.contains(usuario)) {
34                    _usuarios.add(usuario);
35                } else
36                    throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
37                            + nome + "\"! Use outro nome!");
38    }
```

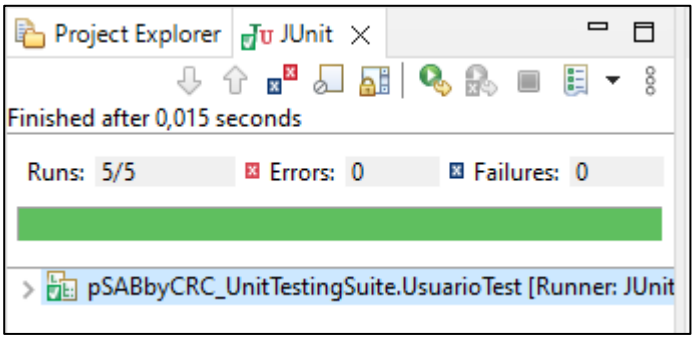
O código já começa apresentar com maior facilidade um interpretação melhor e mais bem-sucedida das responsabilidades da classe registraUsuario.

Aplicação do teste









3° Trecho de código: **linha 33** e **linhas 36, 37**

Tipo de mau cheiro: Código escrito de forma negativa, isso é mais difícil do cérebro humano entender.

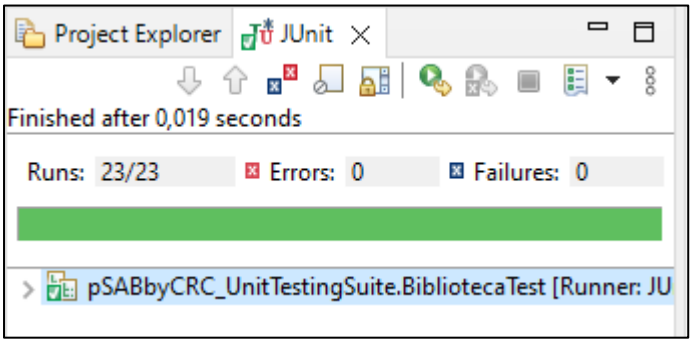
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (!nome.isEmpty())
31            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
33                if (!_usuarios.contains(usuario)) {
34                    _usuarios.add(usuario);
36                    throw new UsuarioJaRegistradoException("--->J‡ existe usu‡rio com o nome \""
37                            + nome + "\"! Use outro nome!");
```

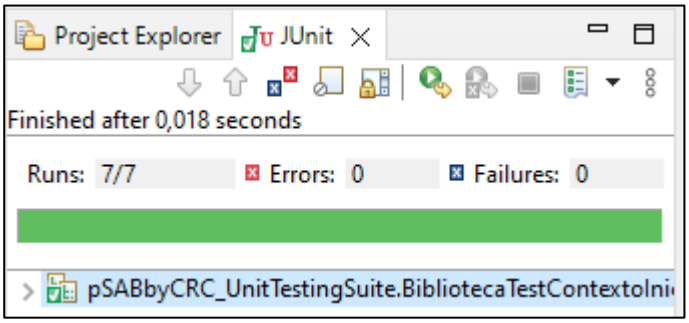
Refatoração

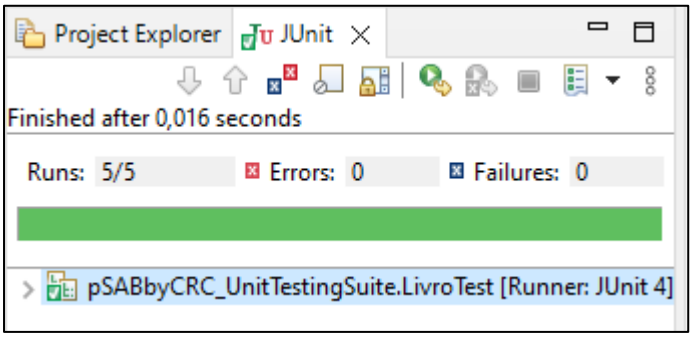
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (nome.isEmpty())
31            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
33        if (_usuarios.contains(usuario))
34            throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""+ nome + "\"! Use outro nome!");
36        else
37            _usuarios.add(usuario);
38    }
```

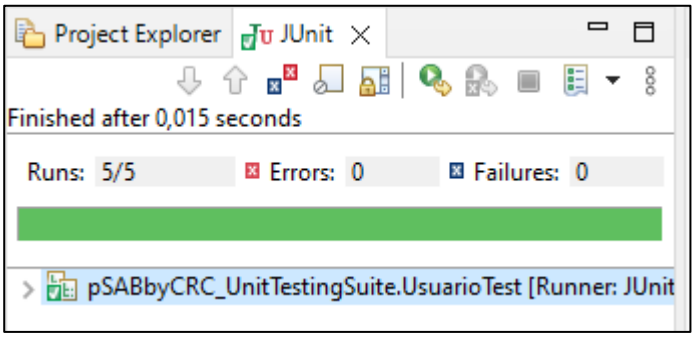
Após essa refatoração já fica nítido o entendimento do código, em referencia a essa refatoração nas linhas 33 e 34 facilmente percebe-se que o código está informando que já existe um usuário com este nome que ele deve tentar outro nome.

Aplicação do teste









4° Trecho de código: **linhas 32, 33 e 37**
Tipo de mau cheiro: Nome inadequado para a variável usuario*, não apresentando qual a intenção*

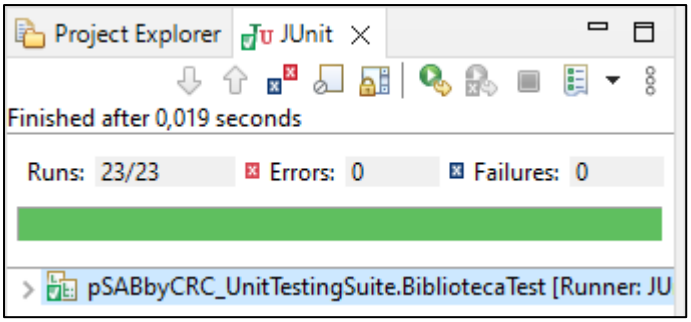
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (nome.isEmpty())
32                Usuario usuario = new Usuario(nome);
33        if (_usuarios.contains(usuario))
34            throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""+ nome + "\"! Use outro nome!");
35
36        else
37            _usuarios.add(usuario);
```

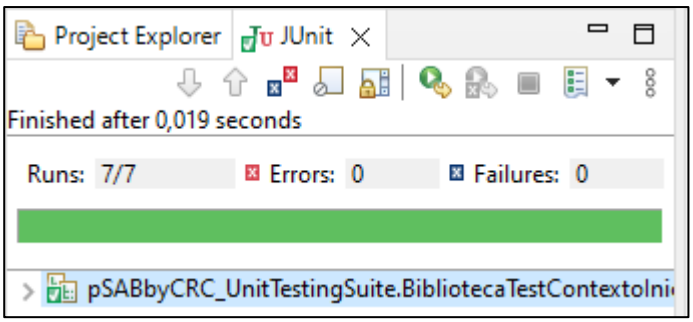
Refatoração

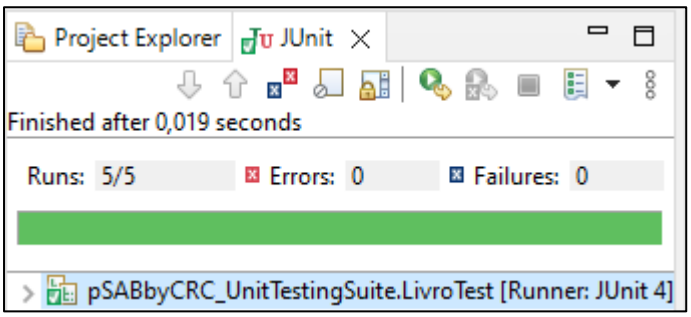
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (nome.isEmpty())
32                Usuario novoUsuario = new Usuario(nome);
33        if (_usuarios.contains(novoUsuario))
34            throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""+ nome + "\"! Use outro nome!");
35
36        else
37            _usuarios.add(novoUsuario);
```

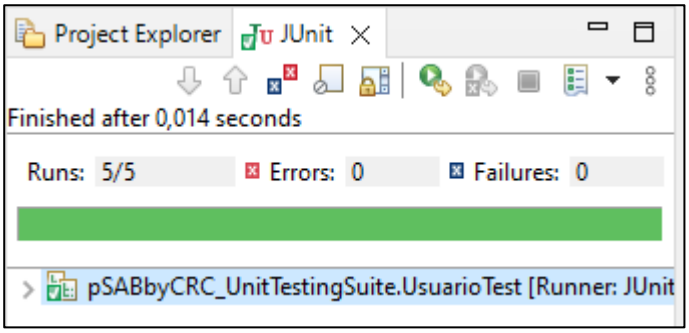
Com ajuda do Eclipse, utilizo o comando "refactor" na variável usuario*, seleciono o "rename" e altero seu nome para* **novoUsuario** *de forma que as alterações "são transpassadas a todos os locais" onde a variável está sendo utilizada.*

Aplicação do teste









5° Trecho de código: **linhas 33 e 37**

Tipo de mau cheiro: Nome inadequado, o nome _usuarios *não deixa definido explicitamente que se trata de uma lista, foi necessário visualizar a declaração para a confirmação de que se trata de uma lista, claro que o método* contains *me transmiti a intenção, porem foi identificado o mau cheiro para ser refatorado.*

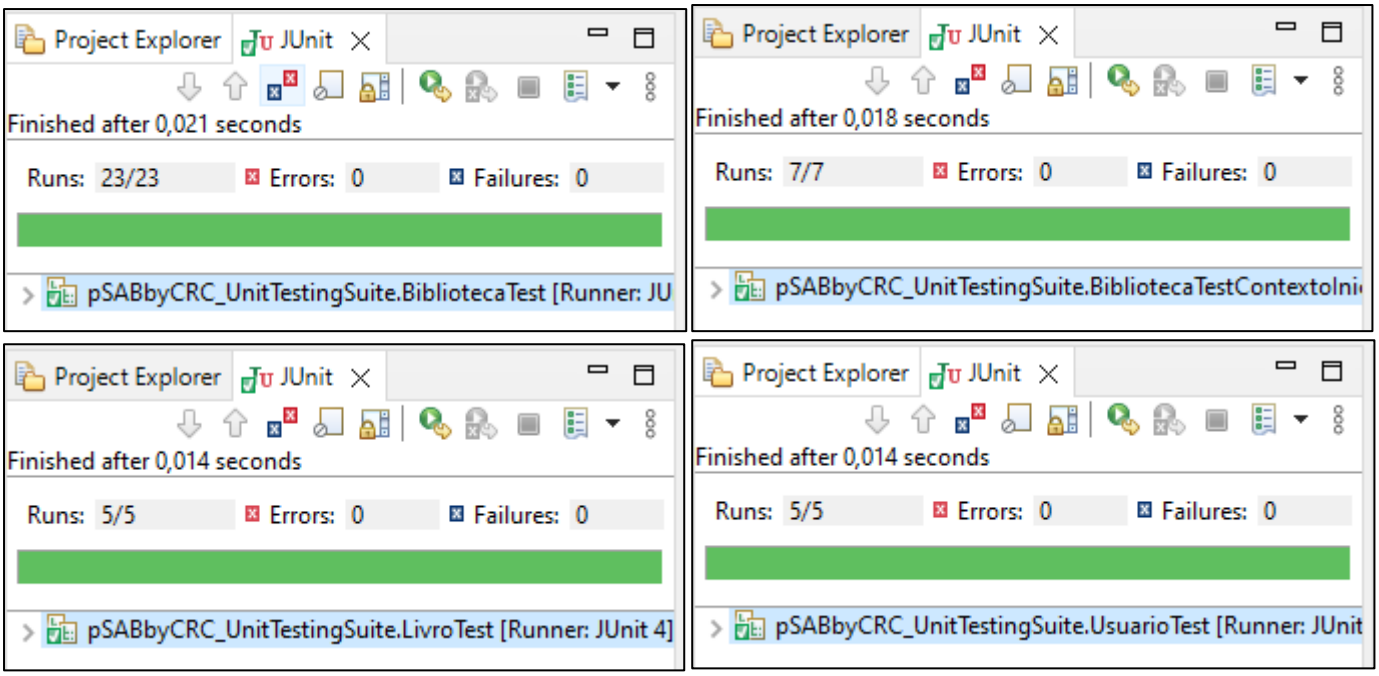
```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (nome.isEmpty())
31            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
33        if (_usuarios.contains(novoUsuario))
34            throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""+ nome + "\"! Use outro nome!")
35
36        else
37            _usuarios.add(novoUsuario);
```

Refatoração

```
25    public void registraUsuario(String nome)
26            throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
27            UsuarioInexistenteException {
28        if (nome == null)
29            throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
30        if (nome.isEmpty())
31            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
33        if (listaDeUsuarios.contains(novoUsuario))
34            throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""+ nome + "\"! Use outro nome!");
35
36        else
37            listaDeUsuarios.add(novoUsuario);
```

Com ajuda do Eclipse, utilizo o comando "refactor" na variável de instância _usuarios*, seleciono o "rename" e altero seu nome para* **listaDeUsuarios** *de forma que as alterações "são transpassadas a todos os locais" onde a variável está sendo utilizada, deixado ela por último para não afetar todos os testes.*

Aplicação do teste



Apresentação do código final.

```
public void registraUsuario(String nome)
                  throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
UsuarioInexistenteException {
            if (nome == null)
                  throw new UsuarioInexistenteException("--->Não pode registrar usuario
inexistente!");
            if (nome.isEmpty())
                  throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com
nome vazio!");

            Usuario novoUsuario = new Usuario(nome);

            if (listaDeUsuarios.contains(novoUsuario))
                  throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome
\""+ nome + "\"! Use outro nome!");

                  listaDeUsuarios.add(novoUsuario);
      }
```

Apresentação da bateria de testes final.

Bateria de testes não modificada em nenhum momento do ciclo de refatoração.