Relatório do curso de TEST DRIVEN DEVELOPMMENT (TDD)
ITA (Instituto Tecnológico de Aeronaútica)

REFATORAÇÃO DO SAB

Aluno : **Tiago Ribeiro Santos**

Email : tiago.programador@hotmail.com

Linkedin: https://www.linkedin.com/in/tiagoribeirosantos/

Refatoração do SAB, consiste em identificar os maus cheiros(bad smells) no código antes de implementar a técnica de refatoração.

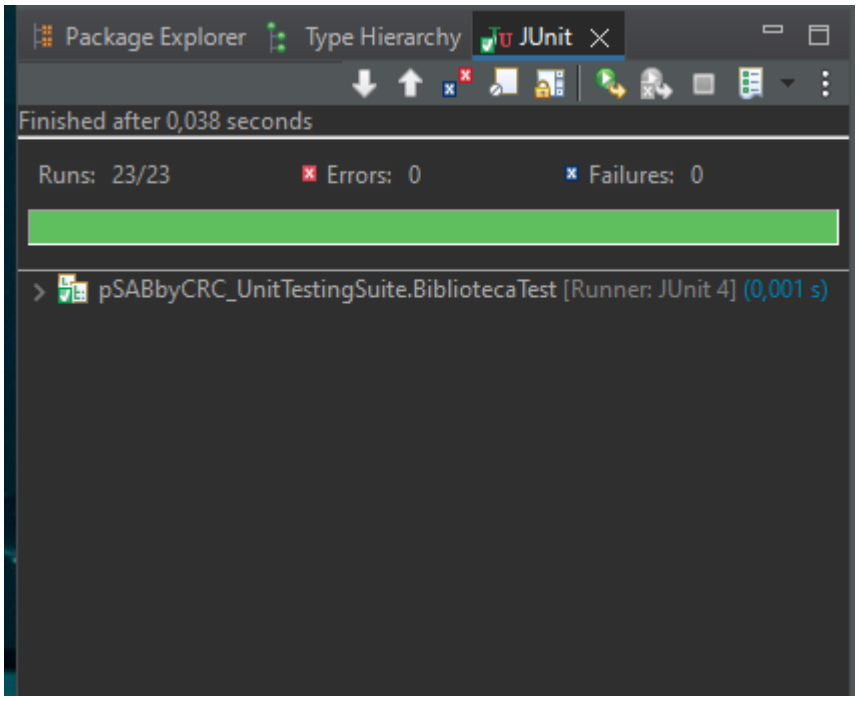
Examinação do método registraUsuario(String) descrita em passos antes de iniciar o ciclo de refatoração

Antes da refatoração

```
public void registraUsuario(String nome)
        throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
        UsuarioInexistenteException {
    if (nome != null) {
        if (!nome.isEmpty()) {
            Usuario usuario = new Usuario(nome);
            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""
                        + nome + "\"! Use outro nome!");
        } else
            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
    } else
        throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
}
```

Aqui logo de cara, podemos observar alguns exemplos de maus cheiros(bad smells) no código. Como por exemplo, a má formatação além da má identação e o ciclo de refatoração nos ajudará a resolver isso tornando o código mais “clean”(limpo).

Mas antes, vamos verificar os nossos cenários de testes para o nosso código de produção abaixo:



O teste passa, mas não significa que precisamos refatorar nosso código.

1º trecho de mau cheiro no código (bad smell)

```
public void registraUsuario(String nome)
        throws UsuarioJaRegistradoException, UsuarioComNomeVazioException,
        UsuarioInexistenteException {
    if (nome != null) {
        if (!nome.isEmpty()) {
            Usuario usuario = new Usuario(nome);
            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \""
                        + nome + "\"! Use outro nome!");
        } else
            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
    } else
        throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
}
```

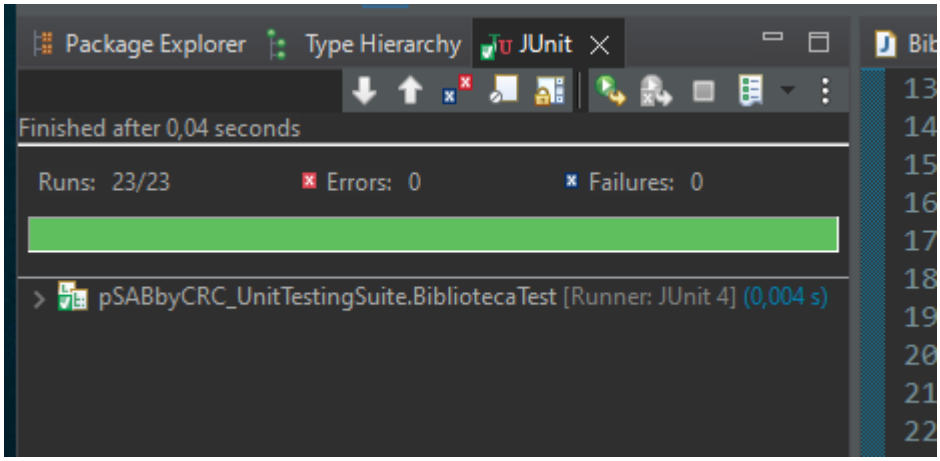
Analisando o código de acordo com as aulas sobre refatoração, é usado este **if negativo** que dificulta o entendimento do que esse trecho significa, até porque era bem mais simples apenas avaliar se o nome é nulo e se sim, disparar uma exceção informando ao **Usuário que não pode ser registrado um usuário inexistente.**

Refatorando o código

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

            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException(
                        "--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
        } else
            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
}
```

Testando após refatoração



2º trecho de mau cheiro no código (bad smell)

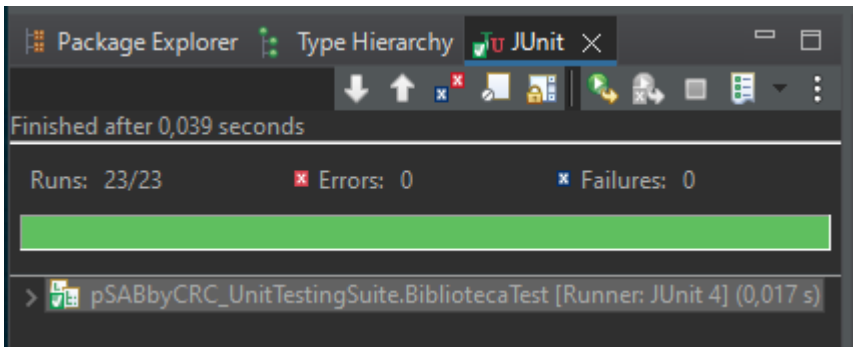
```
public void registraUsuario(String nome)
        throws UsuarioJaRegistradoException, UsuarioComNomeVazioException, UsuarioInexistenteException {
    if (nome == null) throw new UsuarioInexistenteException("--->Não pode registrar usuario inexistente!");
        if (!nome.isEmpty()) {
            Usuario usuario = new Usuario(nome);
            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException(
                        "--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
        } else
            throw new UsuarioComNomeVazioException("--->Não pode registrar usuario com nome vazio!");
}
```

Novamente aqui , podemos verificar que **um if negativo** está usado o que é mal costume , dificultando o entendimento.

Refatorando o código

```
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 usuario = new Usuario(nome);
            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
}
```

Testando após refatoração



3º trecho de mau cheiro no código (bad smell)

```
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 usuario = new Usuario(nome);
            if (!_usuarios.contains(usuario)) {
                _usuarios.add(usuario);
            } else
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
}
```

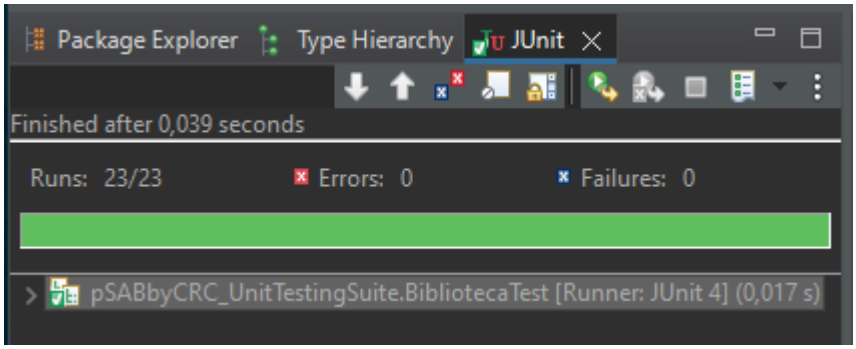
A linha **Usuario usuario = new Usuario(nome);** é meio duvidoso, porque de qual usuário se trata, especificamente? **Qual é realmente a intenção com essa linha? De criar um novo usuário se o nome não for vazio e criá-lo? Claramente, aqui não possui uma intenção direta o que confunde o programador.**

Além do mais, o próximo bad smell visto nesse código é a inserção de mais um **if negativo o que dificulta o entendimento da lógica.**

Refatorando o código

```
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 (_usuarios.contains(novoUsuario))
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
            else
                _usuarios.add(novoUsuario);
}
```

Testando após refatoração



4º trecho de mau cheiro no código (bad smell)

```
_usuarios = new HashSet<Usuario>();
```

```
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 (_usuarios.contains(novoUsuario))
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
            else
                _usuarios.add(novoUsuario);
}
```

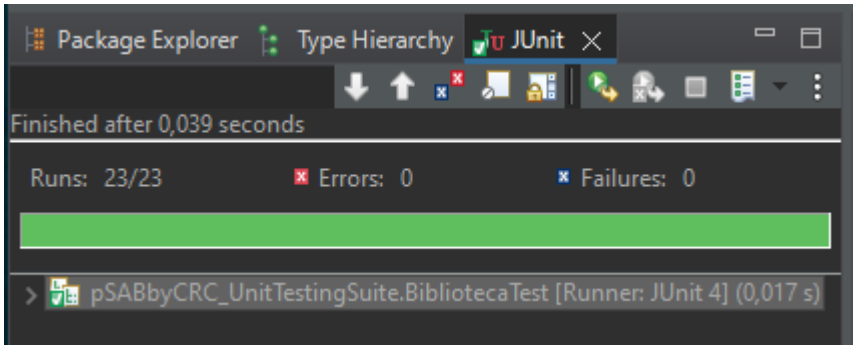
A variável que foi usada do tipo Hashset do tipo “Usuario” para mim foi criada inapropriadamente,**pois o underline antes do nome da variável indica uma variável de instância e não uma lista HashSet ou uma Lista(o método contains é usado por Listas ou HashMaps) não ficando claramente o nome apropriado para ser uma lista Hashmap.** Sendo assim, é interessante refatorar a variável também.

<u>Refatorando o código</u>

```
private HashSet<Usuario> listaUsuarios;
```

```
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 (listaUsuarios.contains(novoUsuario))
                throw new UsuarioJaRegistradoException("--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");
            else
                listaUsuarios.add(novoUsuario);
}
```

Testando após refatoração



Apresentação do código de produção 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 (listaUsuarios.contains(novoUsuario))
        throw new UsuarioJaRegistradoException(
                "--->Já existe usuário com o nome \"" + nome + "\"! Use outro nome!");

    listaUsuarios.add(novoUsuario);
}
```

Bateria de testes final após refatoração