

## Relatório TDD

Curso: TDD - Desenvolvimento Guiado por Testes

Projeto: Refatoração do SAB (Sistema de Automação de Biblioteca)

Autor: Thalys Henrique

LinkedIn: <https://www.linkedin.com/in/thalyshenrique7/>

Passos realizados:

De acordo com **Martin Fowler**, devemos identificar os **maus cheiros** e **refatorar** o código para deixá-lo mais limpo, fácil de ler e realizar manutenções futuras.

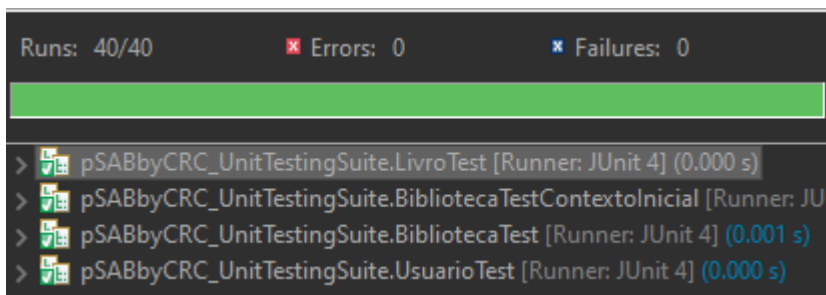
### Classe Biblioteca

Método: **registraUsuario(String nome){}**, 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 usuário com nome vazio!");
    } else
        throw new UsuarioInexistenteException("--->Não pode registrar usuário inexistente!");
}
```

### Classe Testes

Testes da aplicação antes da refatoração:



1º Mau cheiro encontrado:

```

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 usuário com nome vazio!");
    } else
        throw new UsuarioInexistenteException("---->Não pode registrar usuário inexistente!");
}

```

**Expressões booleanas negativas** são ruins para o nosso código, pois fica mais difícil para o nosso cérebro compreender. **Precisamos deixá-lo positivo!**

**Código após refatoração:**

```

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

```

Além de reduzirmos o tamanho do código, deixamos mais fácil de ler.

Testes após a refatoração do código:

```

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0

```

---

```

> pSABbyCRC_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.000 s)
> pSABbyCRC_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JU
> pSABbyCRC_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.001 s)
> pSABbyCRC_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)

```

Ainda no mesmo método, temos mais uma linha de código com mau cheiro:

```

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

```


Vamos removê-lo!

Antes de iniciar a refatoração, mostrarei a bateria de testes:

```

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0

```



```

> [✓] pSABbyCRC_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.000 s)
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JU
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.001 s)
> [✓] pSABbyCRC_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)

```

Código após a refatoração do 2º mau cheiro:

```

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


```

Testes realizados após a refatoração:

```

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0

```



```

> [✓] pSABbyCRC_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.000 s)
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JU
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.001 s)
> [✓] pSABbyCRC_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)

```

Ainda possuímos mais um trecho de código com mau cheiro, segue a imagem:

```

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

```

Vamos para a refatoração!

**Código após a refatoração:**

```

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

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


```

Testes realizados após a refatoração:

```

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0

```



```

> [✓] pSABbyCRC_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.001 s)
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JUnit 4] (0.000 s)
> [✓] pSABbyCRC_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.000 s)
> [✓] pSABbyCRC_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)

```

Por que o código deu certo dessa forma?

R: O **If** sem chaves **{ }** faz com que a próxima linha de código seja avaliada, ou seja, quando temos apenas **um comando** a ser executado.

Dessa vez irei refatorar as variáveis de instâncias, segue a imagem:

```

public class Biblioteca {

    private String _nome;
    private int _nrUnico = 0; // _nrUnico > zero!
    private TreeSet<Livro> repositorioLivros;
    private HashSet<Usuario> _usuarios;
}

```

Aqui temos uma lista de usuários, o ideal seria chamá-la de **listaDeUsuarios**.

```

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

```

Vamos refatorar essas linhas de código!

**Código após a refatoração:**

```

private String _nome;
private int _nrUnico = 0; // _nrUnico > zero!
private TreeSet<Livro> repositorioLivros;
private HashSet<Usuario> listaDeUsuarios;

public Biblioteca(String nome) {
    _nome = nome;
    repositorioLivros = new TreeSet<Livro>();
    listaDeUsuarios = new HashSet<Usuario>();
}

```

```

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

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

```


Vejamos que o código fica mais bonito de se ver, mais limpo, fácil de ser compreendido, facilita até uma possível manutenção futura para os desenvolvedores que forem trabalhar com o código.

**Testes após a refatoração:**

```

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0

```



```

> [p] pSABbyCRC_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.001 s)
> [p] pSABbyCRC_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JUnit 4] (0.000 s)
> [p] pSABbyCRC_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.000 s)
> [p] pSABbyCRC_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)


```





Como vimos na imagem anterior, temos mais variáveis necessitando de refatoração, segue a imagem com todas as refatorações feitas:

```
public class Biblioteca {  
    private String nome;  
    private int nrUnico = 0; // nrUnico > zero!  
    private TreeSet<Livro> repositorioDeLivros;  
    private HashSet<Usuario> listaDeUsuarios;  
  
    public Biblioteca(String nome) {  
        this.nome = nome;  
        repositorioDeLivros = new TreeSet<Livro>();  
        listaDeUsuarios = new HashSet<Usuario>();  
    }  
}
```

Testes realizados:

Runs: 40/40      ✖ Errors: 0      ✖ Failures: 0



>  pSABbyCRC\_UnitTestingSuite.LivroTest [Runner: JUnit 4] (0.001 s)  
>  pSABbyCRC\_UnitTestingSuite.BibliotecaTestContextoInicial [Runner: JUnit 4] (0.000 s)  
>  pSABbyCRC\_UnitTestingSuite.BibliotecaTest [Runner: JUnit 4] (0.000 s)  
>  pSABbyCRC\_UnitTestingSuite.UsuarioTest [Runner: JUnit 4] (0.000 s)