ENGENHARIA DE SOFTWARE II
Atividade 02 – Testes Unitários com JUnit, SOLID e TDD

Data de entrega e apresentação: 18-10-2022

Cenário: Uma aplicação responsável por fazer um controle de empréstimo de livros.

Para tanto, teremos que criar uma classe **`Livro`**.

A classe **`Livro`** **(**veja o fragmento abaixo**)** **terá** os atributos **`autor`**, **`titulo`**, **`emprestado`** e **`reservado`**. Estes dois últimos são do tipo boolean. Também possui um atributo chamado **`historico`** que é uma lista de Empréstimos.

```java
public class Livro {
  private String autor;
  private String titulo;
  private boolean emprestado;
  private boolean reservado;
  private List<Emprestimo> historico;

  public boolean emprestar(){
    if(!this.isEmprestado()){
      this.setEmprestado(true);
      System.out.println("Livro emprestado com sucesso.");
    }
    return this.isEmprestado();
  }

  public List<Emprestimo> consultarEmprestimosPorUsuario(Usuario usuario){
    List<Emprestimo> emprestimosPorUsuario = new ArrayList<Emprestimo>();
    for (Emprestimo emprestimoUsuario : emprestimosPorUsuario) {
      if(emprestimoUsuario.getUsuario().equals(usuario)){
        emprestimosPorUsuario.add(emprestimoUsuario);
      }
    }
    return emprestimosPorUsuario;
  }
}
```

Classe **`Emprestimo`**: Cada empréstimo da lista contém informações como:

1. quem foi o *Usuário* que pegou o livro emprestado,

```java
public class Usuario {
  private String nome;
   private String matricula;
  private boolean emDebito;
```

2. qual foi a data do empréstimo (`dataEmprestimo`),
3. data prevista para devolução (`dataPrevista`). A data prevista é definida pelo sistema, e sempre será 7 dias após a data do empréstimo.
4. qual foi a data da devolução (`dataDevolucao`).
5. Cada empréstimo estará associado a uma Lista de Livros)
6. Valor do empréstimo. Fica definido o valor fixo de R$ 5,00 para cada livro emprestado. Caso a devolução ocorra com atraso será aplicada seguinte regra: R$0,40 por dia de atraso para cada livro, limitada a 60% do valor do aluguel.

OBS: Um usuário poderá simultaneamente fazer a locação de até 3 livros.

Esta aplicação tem o objetivo apenas de criar elementos com as estruturas necessárias para exemplificar o uso de testes unitários.

Nota 01: Você deverá realizar decisões de projeto na organização das classes, nomenclatura, visibilidade e definição de métodos, criação de classes de serviço e de teste.

Nota 02: Você pode sugerir mudanças e refatoração neste cenário

Nota 03: Programe com os princípios SOLID em mente.

Nota 04: Tente desenvolver o código com a prática do TDD.

A bateria de testes deve possuir no mínimo os seguintes cenários de teste:

- **Realizar empréstimo em livro que não esteja reservado.**
- **Tentativa de empréstimo em livro que já possui uma reserva.**
- **Garantir que a data prevista esteja correta, após a locação de um livro.**
- **Testa um usuário sem empréstimo.**
- **Usuário com 1 (um) empréstimo.**
- **Usuário com 3 (três) empréstimos**
- **Tentativa de realizar um 4° empréstimo para o mesmo usuário.**

Na classe Livro refatore o método *consultarEmprestimosPorUsuario()* usando função lambda, em seguida execute a bateria de testes.

Para os próximos testes deverão ser checados:

- **uma devolução antes da data prevista**
- **uma devolução na data prevista**
- **uma devolução 1(um) dia após a data prevista**
- **uma devolução 30 dias após a data prevista**

Links úteis:

https://howtodoinjava.com/junit-5-tutorial/

https://junit.org/junit5/docs/current/user-guide/