PLANO DE TESTE PARA

BRACKET PAIR COLORIZED

Registro de Mudanças

Versão	Data de Mudança	Por	Descrição
1.1	19/06/2023	Marcos Roberto	Plano de Testes
1.2	19/06/2023	Amanda Marques	Adição de casos de teste da classe LineState
1.3	19/06/2023	Kathleen Freitas	
1.4	21/06/2023	Fábio Gil	Adição de caso de teste e atualização na documentação

1 Introdução

O projeto **Bracket Pair Colorized** é uma extensão do VSCode que melhora a legibilidade do código destacando parênteses, colchetes e chaves correspondentes com diferentes cores para melhor visualização e entendimento por parte do desenvolvedor. Sua última versão foi lançada em 2018 e tem cerca de 12 contribuidores até então.

O os casos de teste serão desenvolvidos para as seguintes classes:

1. GutterIconManager, responsável por gerenciar ícones exibidos na área vertical ao lado do código.
2. LineState, responsável por gerenciar as cores dos símbolos.

1.1 Escopo

1.1.1 No escopo

O objetivo deste projeto é testar as funcionalidades do sistema, baseando-se nos princípios de qualidade apresentados na disciplina de Qualidade e Teste.

1.1.1.1 GutterIconManager

O teste deve abranger todas as funcionalidades e métodos da classe GutterIconManager, como Dispose(), GetIconUri(), createIcon(), entre outros. Os testes devem verificar se os recursos estão implementados corretamente e se fornecem a funcionalidade esperada.

1.1.1.2 LineState

Serão testados os métodos getOpenBracketColor() e getCloseBracketColor(), além de saídas de estruturas condicionais da classe.

1.1.1.3 SingularIndex

Serão testados os métodos getCurrentColorIndex(), getScope(), getPreviousIndex(), entre outros e a integração da classe com outras funcionalidades do software.

1.1.2 Fora do escopo

Não serão testadas neste projeto as classes : IcolorIndex.ts, bracket.ts, bracketPair.ts, colorMode.ts, colors.ts (...) por não cumprirem os critérios de complexidades exigidos na disciplina.

1.2 Objetivos de Qualidade

1.2.1.1 GutterIconManager

Em resumo, o objetivo é fornecer confiança na qualidade e funcionalidade do GutterIconManager, sendo capaz de realizar as seguintes tarefas:

- Verificar a criação de ícone para bracket ou cor inválidos
- Liberação adequada de recursos ao chamar o método dispose
- Reutilização correta de ícones

Assim, os testes ajudariam a garantir confiabilidade no código e eficiência, estando em conformidade com as expectativas de uso e funcionalidade.

1.2.1.2 LineState

O objetivo dos testes aqui é o de garantir que as cores antecedentes e subsequentes de um símbolo específico mudem corretamente no próximo uso em uma classe, método ou estrutura condicional, de forma sequencial pré definida pelas configurações padrões do sistema ou do próprio usuário.

1.2.1.3 SingularIndex

Serão testados os métodos getCurrentColorIndex(), getScope(), getPreviousIndex(), entre outros e a integração da classe com outras funcionalidades do software.

1.2.1.4 TextLine

Será testado o método copyMultilineContext() que deve retornar uma cópia da linha teste com o estado do bracket conservado.

1.3 Papéis e Responsabilidades

Todos os membros participaram de discussões para a repartição da responsabilidade de cada classe do projeto. Cada integrante será responsável por garantir a qualidade e os testes de uma classe específica. São eles:

1. GutterIconManager, por Marcos Roberto
2. LineState, por Amanda Marques

3. SingularIndex, por Kathleen Freitas
4. TextLine, por Fábio Gil

2 Metodologia de Teste

2.1 Visão Geral

2.2 Fases de Teste

2.3 Triagem de Erros

2.4 Critérios de Suspensão e Requisitos de Retomada

2.5 Completude do Teste

3 Entregáveis de Teste

- Plano de teste
- Todas as classes Typescript citadas até então

4 Necessidades de Recursos e Ambiente

4.1 Ferramentas de Teste

Utilização do Jest, framework de teste em JavaScript projetado para garantir a correção de qualquer código JavaScript e Typescript.

4.2 Ambiente de Teste

As seguintes ferramentas serão necessárias para o cumprimento correto de todos os testes:

- Visual Studio Code
- GitHub