Gerçek Zamanlı Sözdizimi Vurgulayıcı Uygulama Dokümantasyonu

1. Genel Bakış

Bu proje, Java Swing kütüphanesi kullanılarak geliştirilmiş bir gerçek zamanlı sözdizimi vurgulayıcı (syntax highlighter) uygulamasıdır. Kullanıcıların kod yazabileceği bir metin editörü sağlar ve yazılan kodun anahtar kelimeleri, operatörleri, sayıları, yorumları ve tanımlayıcılarını farklı renklerle vurgulayarak görsel olarak ayırt edilmesini sağlar. Ayrıca, basit bir top-down parser ile temel sözdizimi kontrolü yapar ve satır numaralarını dinamik olarak gösterir.

2. Proje Yapısı

Proje, üç ana bileşenden oluşur:

- **Token Sınıfı**: Sözdizimi birimlerini (token) temsil eden veri yapısı.
- **Lexer Sınıfı**: Kodun leksikal analizini yaparak token'lara ayırır.
- **Parser Sınıfı**: Token'ları analiz ederek basit sözdizimi kontrolü yapar.
- **SyntaxHighlighterGUI Sınıfı**: Grafiksel kullanıcı arayüzünü (GUI) oluşturur ve gerçek zamanlı vurgulama ile satır numaralarını yönetir.

3. Sınıf ve Bileşenler

3.1. Token Sınıfı

- **Amaç**: Kodun temel yapı taşlarını (token) temsil eder.
- **Özellikler**:
    - type: Token'ın türü (örneğin, KEYWORD, OPERATOR, NUMBER, COMMENT, IDENTIFIER).
    - value: Token'ın değeri (örneğin, "if", "+", "42").
    - start: Token'ın kod içindeki başlangıç konumu.
    - end: Token'ın kod içindeki bitiş konumu.
- **Kullanım**: Lexer tarafından oluşturulan token'lar, Parser ve GUI tarafından işlenir.

3.2. Lexer Sınıfı

- **Amaç**: Girilen kodu token'lara ayırarak leksikal analiz yapar.
- **Özellikler**:
    - keywords: Anahtar kelimeler kümesi (if, while, else, for, return).
    - operators: Operatörler kümesi (+, -, *, /, =, ==).

    - tokenize metodu: Kod string'ini tarar ve token listesi üretir.
- **Desteklenen Token Türleri**:
    - **COMMENT**: // ile başlayan tek satırlık yorumlar.
    - **OPERATOR**: Tanımlı operatörler.
    - **NUMBER**: Tam veya ondalıklı sayılar.
    - **KEYWORD**: Tanımlı anahtar kelimeler.
    - **IDENTIFIER**: Değişken adları veya diğer tanımlayıcılar.
- **Çalışma Mantığı**:
    - Kodu karakter karakter tarar.
    - Boşlukları atlar.
    - Yorumları, operatörleri, sayıları ve tanımlayıcıları/anahtar kelimeleri algılar.
    - Bilinmeyen karakterleri atlar.

3.3. Parser Sınıfı

- **Amaç**: Token'ları analiz ederek basit bir sözdizimi kontrolü yapar.
- **Özellikler**:
    - tokens: Lexer'dan gelen token listesi.
    - pos: Mevcut token pozisyonu.
    - parse metodu: Token'ları tarar ve if anahtar kelimesinden sonra bir tanımlayıcı olup olmadığını kontrol eder.
- **Kısıtlamalar**:
    - Şu an sadece if anahtar kelimesinden sonra tanımlayıcı kontrolü yapar.
    - Hata durumunda bir istisna fırlatır.

3.4. SyntaxHighlighterGUI Sınıfı

- **Amaç**: Kullanıcı arayüzünü oluşturur ve gerçek zamanlı sözdizimi vurgulaması sağlar.
- **Bileşenler**:
    - **JFrame**: Ana pencere.
    - **JTextPane**: Kod yazım alanı.

    - **StyledDocument**: Metin stillerini yönetir.
    - **JTextArea (lineNumbers)**: Satır numaralarını gösterir.
    - **JScrollPane**: Metin alanı ve satır numaralarını birleştirir.
    - **JPanel (legendPanel)**: Renk göstergesi paneli.
- **Özellikler**:
    - **Metin Alanı Özellikleri**:
        - Arka plan: Siyah-yakın gri (#1E1E1E).
        - Metin ve imleç rengi: Beyaz.
        - Yazı tipi: Monospaced, 12 punto.
    - **Renk Şeması**:
        - Anahtar Kelimeler: Açık mavi (#6495ED).
        - Operatörler: Açık kırmızı (#FF4040).
        - Sayılar: Mor (#8C5A8C).
        - Yorumlar: Yeşil.
        - Tanımlayıcılar: Beyaz.
    - **Satır Numaraları**: Metin alanındaki satır sayısına göre dinamik olarak güncellenir.
    - **Renk Göstergesi**: Sağ altta, her token türünün rengini gösteren bir panel.
- **Olay Yönetimi**:
    - KeyListener: Her tuş vuruşunda highlight ve updateLineNumbers metodlarını çağırır.
    - highlight: Kodu lexer ile token'lara ayırır, token'lara göre vurgulamayı günceller ve parser ile sözdizimi kontrolü yapar.
    - updateLineNumbers: Metindeki satır sayısına göre satır numaralarını günceller.

4. Kullanım

1. Uygulama başlatıldığında, bir JFrame penceresi açılır.
2. Kullanıcı, metin alanına kod yazabilir.
3. Yazılan kod, lexer tarafından token'lara ayrılır ve her token türüne uygun renk uygulanır.

4. Satır numaraları otomatik olarak güncellenir.
5. Sağ altta yer alan renk göstergesi, hangi renklerin hangi token türlerini temsil ettiğini belirtir.
6. if anahtar kelimesinden sonra tanımlayıcı olmaması durumunda konsolda hata mesajı görünür.

5. Kurulum ve Çalıştırma

1. Kodu bir Java IDE'sine (örneğin, IntelliJ IDEA, Eclipse) veya metin editörüne kopyalayın.
2. SyntaxHighlighterApp.java dosyasını derleyin ve çalıştırın:
3. javac SyntaxHighlighterApp.java
4. java SyntaxHighlighterApp
5. Uygulama, 600x400 piksel boyutlarında bir pencere açar.

6. Örnek Kullanım

Aşağıdaki kod metin alanına yazıldığında:

if variable == 42 // Bu bir yorum

while true

- if ve while açık mavi renkte vurgulanır.
- == açık kırmızı renkte vurgulanır.
- 42 mor renkte vurgulanır.
- // Bu bir yorum yeşil renkte vurgulanır.
- variable ve true beyaz renkte vurgulanır.

7. Sonuç

Bu uygulama, temel bir sözdizimi vurgulayıcı ve editör olarak kullanılabilir. Eğitim amaçlı projeler veya basit kod düzenleme ihtiyaçları için uygundur. Esnek yapısı sayesinde, yeni token türleri veya sözdizimi kuralları eklenerek genişletilebilir.