Gerçek Zamanlı Sözdizimi Vurgulayıcı

(Real-Time Grammar-Based Syntax Highlighter with GUI)

Final Raporu

1. Giriş

Bu projede, yazdığımız kodun sözcüksel ve sözdizimsel olarak analiz edilip gerçek zamanlı şekilde renklendirilmesini sağlayan bir uygulama geliştirdim. Program, yazılan kodu anlık olarak tanıyıp hangi kelimenin ne tür bir öğe olduğunu belirliyor ve ona göre renklendiriyor. Böylece kullanıcı, yazdığı kodu daha rahat okuyabiliyor ve yazım hatalarını daha kolay fark edebiliyor.

2. Teknoloji ve Araçlar

Uygulamayı **Java** programlama diliyle geliştirdim. Arayüz için ***JavaSwing*** kütüphanesinden faydalandım. Java'da *JTextPane*, *StyledDocument* gibi sınıflar sayesinde metin üzerinde farklı renklerle işlem yapabildim.

3. Sözcüksel Analiz (Lexical Analysis)

Yazılan kodun ilk aşamada parçalara (tokenlara) ayrılması gerekmektedir. Bu işlem için kendi yazdığım ***LexicalAnalyzer*** sınıfını kullandım. Bu sınıf, girilen kodu satır satır okuyarak aşağıdaki türlerde token'lara ayırır:

- **Keyword (Anahtar kelimeler)** → (int, if vb.)
- **Number (Sayılar)** → (örneğin: 5, 123)
- **Operator(Operatörler)** → (+, -, =)
- **Comment(Yorum satırları)** → (// veya /* */)
- **Identifier (Değişken adları)**
- **String Literal (Metin ifadeleri)** →(tırnak içindeki string'ler)

Her token belirli kurallarla tanımlanmıştır ve renklendirme işlemi buna göre yapılmaktadır.

4. Sözdizimsel Analiz (Parsing)

Daha sonra, kodun sadece kelime olarak değil, anlamlı bir yapı olup olmadığını kontrol etmek için sözdizim analizini yaptım.

Örneğin:

- int x = 7; → bu yapı doğru bir tanımlamadır. Ancak
- int = x 7; → bu yapı hatalı söz dizimidir.

Bunu anlayabilmek için kurallar belirledim ve ***SimpleParser*** adında bir sınıf yazdım. ***SimpleParser*** sınıfı, yukarıdan aşağıya (top-down) çalışan özyinelemeli bir parser'dır. Burada özellikle if gibi yapılar, atama ifadeleri ve değişken tanımlamaları denetlenmektedir.

Parser, kod yapısında hata olduğunda kullanıcıya anlaşılır bir hata mesajı vermektedir. Böylece hem öğrenmeye katkı sağlanır hem de yazım hataları kolayca fark edilir.

5. Renklendirme (Highlighting)

Her token türü için özel bir renk belirlenmiştir. Böylece kullanıcı yazarken bu renkler otomatik olarak uygulanır:

TOKEN TÜRÜ	RENK KODU	AÇIKLAMA
KEYWORD (Anahtar Kelimeler)	#5568AF	Örnek: int, if
NUMBER (Sayılar)	#F37826	Rakamlar
OPERATOR(Operatörler)	#EC1763	+, -, = gibi işaretler
COMMENT(Yorum satırları)	#CDD629	//, /* */
IDENTİFİER (Değişken Adları)	#F8C9DD	Tanımlanan isimler
STRİNG LİTERAL (Metin İfadeleri)	#CEEAEE	Tırnak içindeki yazılar

Renklendirme işlemi, ***SyntaxHighlighterGUI*** adlı sınıf tarafından yapılır. Kullanıcı kod yazarken *DocumentListener* ile yazılan metin izlenir ve analiz tetiklenir.

6. Arayüz (GUI) Tasarımı

Arayüz basit ve kullanımı kolay olacak şekilde tasarladım. *JTextPane* üzerinde yazılan her şey anında analiz edilmekte ve hatalı yazımlar varsa kullanıcıya hata mesajı gösterilmektedir.

Arayüzde şunlar yer alıyor:

- Kod yazma alanı
- Gerçek zamanlı renklendirme
- Hata mesajı gösterimi

7. Yaşanan Zorluklar

Bu projeyi yaparken Java'da *StyledDocument* ile nasıl renklendirme yapılacağını ilk kez öğrendim. Ayrıca, kendi yazdığım ***LexicalAnalyzer*** ve ***SimpleParser*** sınıflarını birbirine entegre etmek başlangıçta zor oldu ama öğrendikçe daha iyi hale getirdim.

Ayrıca ***JavaSwing*** ile renklendirme yapmak için uygun yapıyı kurmak başlangıçta biraz karışıktı. Ama zamanla öğrenerek çözdüm.

8. Sonuç

Proje boyunca hem Java bilgimi geliştirdim hem de sözcüksel ve sözdizimsel analiz konularını daha iyi anladım. Gerçek zamanlı renklendirme sistemi çalışır durumda ve sade bir arayüzle kullanıcıya kolaylık sağlıyor.

SELEN YAKIN - 23360859034