Regex Tabanlı Java Sözdizimi Çözümleyici: Gerçek Zamanlı Vurgulama Uygulaması

1. Programming Language and Grammar Choice

Kullanılan Dil: Java
Tanımlanan Gramer: Java dilinin temel yapıları dikkate alınarak;
- Anahtar kelimeler (public, class, if, return, vs.)
- Operatörler (+, ==, &&, vs.)
- Sayılar (tam ve ondalıklı)
- String ifadeler (“merhaba”)
- Yorum satırları (// ve /* ... */ biçiminde)

Tam bir gramer tanımı yoktur; bağlamdan bağımsız, yüzeysel token tanıma hedeflenmiştir.

2. Syntax Analysis Process

Genel Yapı:
Kod çözümleme süreci şu sırayla işler:
1. Lexer (Regex Tabanlı Tokenizer): Metin düzenli ifadelere göre parçalanır.
2. Parser: Kullanılmamaktadır. Derin yapı (AST) üretilmez.
3. Highlighting: Token’lar renklendirilerek GUI üzerinde gösterilir.

Syntax Analizi:
Gerçek bir parser içermediği için tam anlamıyla syntax analizi yapılmamaktadır. Yüzeysel olarak regex ile tanımlanan token'lar belirlenmekte ve sınıflandırılmaktadır.

3. Lexical Analysis Details

Kullanılan Yöntem:
- Regex Tabanlı Lexer
- State Diagram veya Tablo Tabanlı Yapı: Kullanılmamıştır. Regex deseni doğrudan uygulanır.

Token Türleri ve Regex’leri:

Token Türü	Regex Deseni	Örnekler
Anahtar Kelime	\b(public|class|if|...)\b	public, return
Operatör	[+\-*/=<>!&|%^]+	+, ==, &&
Sayı	\b\d+(\.\d+)?\b	42, 3.14
Yorum	//.*?$|(?s)/*.*?*/	// yorum, /*...*/
String	"(?:\\.|[^"\\])*"	"merhaba"

Avantaj: Basit, hızlı ve düşük karmaşıklıklı tokenizasyon sağlar.
Dezavantaj: Bağlam duyarlılığı yoktur, örneğin "if" gibi string içinde geçen yapılar da vurgulanır.

4. Parsing Methodology

Parser Türü: Kullanılmamıştır.
Regex yaklaşımıyla yalnızca lexer katmanı uygulanmıştır.

Seçim Nedeni:
- Basit prototipleme
- Düşük kaynak tüketimi
- Gerçek zamanlılık için yeterli performans

Parser Yapısı: Yoktur. Kod çözümlemesi tamamen pattern matching'e dayanır.

5. Highlighting Scheme

Tanımlı Token Türleri ve Vurgulama:
Aşağıdaki token türleri renklendirilmiştir:
1. Anahtar Kelime (keyword) – Örn: public → mavi
2. Tanımlayıcı (identifier) – Tanımsız, tüm tanımlanmamış kelimeler → varsayılan
3. Sayı (number) – Örn: 42, 3.14 → turuncu
4. String – Örn: "merhaba" → yeşil
5. Operatör – Örn: ==, +, && → mor
6. Yorum – Tek ve çok satırlı yorumlar → gri/italic

Vurgulama işlemi StyledDocument üzerinden yapılır.

6. GUI Implementation

Kullanılan Framework: Java Swing

Temel Bileşenler:
- JTextPane: Metin giriş ve gösterim
- StyledDocument: Renkli biçimlendirme
- ScheduledExecutorService: debounce mekanizması ile performans optimizasyonu sağlar.

Gerçek Zamanlılık:
- Kullanıcı girişinden sonra belirli bir gecikmeyle (debounce) highlight() fonksiyonu tetiklenir.
- Bu sayede her tuş vuruşunda değil, duraklamadan sonra çözümleme yapılır.

Arayüz Yapısı:
- Tek pencere yapısı
- Üst panelde metin giriş alanı
- Arka planda çalışan lexer ile çözümleme ve biçimlendirme