GİRİŞ

Programlama dillerinde bulunan **Syntax** yani **söz dizimi**, çeşitli sembol kombinasyonlarının ne anlama geldiğini tanımlar. Kullanılan programlama dilinin kurallarını belirler. Anahtar kelimeler ve simgeler Syntax sayesinde tanımlanır. Böylece Syntax highlighting (kodları gerçek zamanlı renklendirme) ile hata yapma olasılığı azalır ve kodun yapısının daha iyi anlaşılması sağlanır.

Bu projede, kütüphane veya hazır araçlar kullanılmadan, tamamen gramer tabanlı gerçek zamanlı syntax highlighter geliştirilmiştir. Proje kapsamında, öncelikle bir **lexical analyzer** (sözcüksel analizör) ile girilen kodun anlamlı tokenlara ayrıştırılması sağlanmış, ardından bir **parser** (sözdizim analizörü) ile tokenlar dilin kurallarına göre işlenmiştir. Sonuç olarak, kullanıcı arayüzünde yazılan kod, en az beş farklı token türü gerçek zamanlı olarak renklendirilmiştir.

Language and Grammar Choice (Dil ve Gramer Seçimi)

Bu projede, sözdizim vurgulayıcısı için **Java programlama dili** tercih edilmiştir. Java'nın tam sözdizimi oldukça geniş ve karmaşık olduğu için yalnızca temel yapılar dikkate alınmıştır. Bu yapılar **değişken tanımlamaları, veri tipleri, kontrol yapıları, operatörler ve yorum satırlarıdır.** Proje kapsamında Java dilinin gramer yapıları sadeleştirilmiş ve belirli token türleri ile sınırlandırılmıştır. Örneğin; anahtar kelimeler, tanımlayıcılar, sayılar, operatörler, semboller, yorumlar.

Syntax Analysis Process (Sözdizimi Analiz Süreci)

Bu projede söz dizimi analizi, gramer kurallarına uygun olarak gerçekleştirilmiştir. Analizin amacı, tokenize edilen girdinin dilin kurallarına uygun olup olmadığını kontrol etmektir.

Proje kapsamında Top-Down Parsing (Yukarıdan Aşağıya Çözümleme) yöntemi tercih edilmiştir. Projede tanımlanan basit kurallara göre tokenlar sırayla kontrol edilir. Yani gramerin başlangıcından başlayıp sırayla alt elemanlara bakılır.

Lexical Analysis Details (Sözcüksel Analiz Detayları)

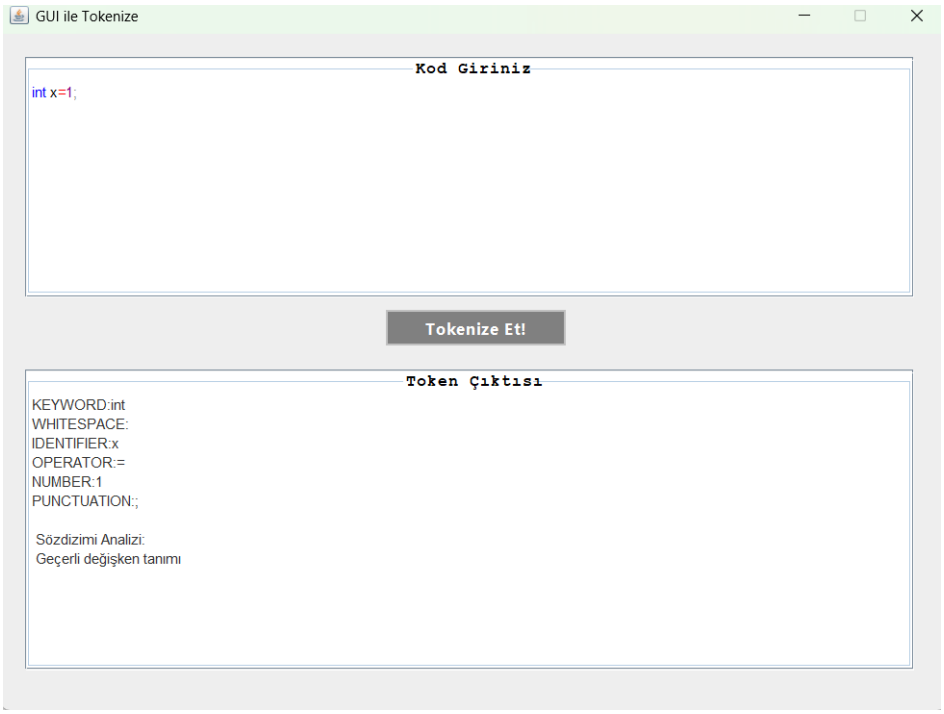
Bu projede sözcüksel analiz işlemi, **State Diagram & Program Implementation** yaklaşımı temel alınarak gerçekleştirilmiştir. Uygulama,kullanıcıdan alınan kod metnini satır ve karakter bazında tarayarak sözcüksel analiz (lexical analysis) işlemi gerçekleştirir. Her satır, düzenli ifadeler aracılığıyla tanımlanmış tokenlara göre parçalanır. Analiz sonucunda her bir parça Token sınıfı ile temsil edilir. Tokenlar kendi türleriyle birlikte ayrı bir listede saklanır. Projede, Java dilindeki temel token türleri belirlenmiş ve her biri için ayrı kontrol mekanizmaları geliştirilmiştir. Lexer karakter dizisini tek tek okur ve token türlerini tanımlar. Bu token türleri: anahtar kelimeler, tanımlayıcılar, sayılar, operatörler, semboller ve yorum satırlarıdır. Lexer, boşluk, tab ve satır sonu karakterlerini atlayarak, kodu en doğru şekilde anlamlı tokenlara dönüştürür. Ayrıca, dilin temel yapıları dışındaki karakterler ve hatalı diziler analiz sırasında tespit edip kullanıcıya bildirir.

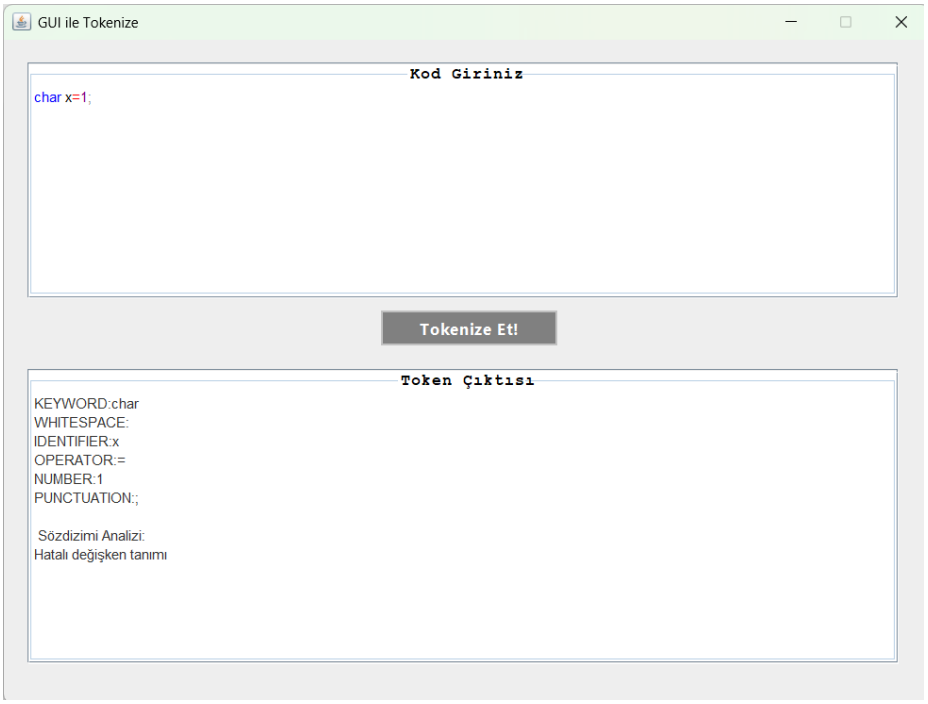
```java
public enum TokenTipi{
    KEYWORD,IDENTIFIER,NUMBER,OPERATOR,PUNCTUATION,WHITESPACE,UNKNOWN,COMMENT,STRING, CHAR_DEGER
}


public static class Token{
    public final TokenTipi tip;
    public final String metin;

    public Token(TokenTipi tip,String metin)
    {
        this.tip=tip;
        this.metin=metin;


    }
    public String toString()
    {
        return tip+":"+ metin;

     }


    public static List<Token> tokenize(String girdi){
        String[] KEYWORDS= {"abstract","boolean", "break","case", "catch", "const", "continue","int", "float", "double", "char", "bo
        "if",  "interface", "implements", "import", "instanceof","else", "while", "for", "return", "void", "class", "public", "priva
        char[] OPERATORS= {'+', '-', '*', '/', '=','<','>'};
        char[] PUNCTUATIONS= {'(', ')', ';', '{', '}','.'};
        char[] WHITESPACE= {};
```

Parsing Methodology (Sözdizimsel Çözümleme Yöntemi)

Tokenlara ayrılmış kod parçaları üzerinde temel sözdizimi kontrolü yapılmaktadır. Bu kontrol, “SözDizimiAnalizi” adlı sınıf tarafından gerçekleştirilir. Amaç, kullanıcıya yalnızca tokenların değil aynı zamanda token dizilimlerinin de doğru olup olmadığını göstermektir.

Örneğin, int x=1; şeklinde bir ifade sırasıyla veri tipi, tanımlayıcı, değer ve noktalı virgül tokenlarından oluştuğu için doğru kabul edilir. Ancak char x=1; gibi yanlış veri tipi kullanılmış token dizileri hatalı olarak işaretlenir.





Highlighting Scheme (Gerçek Zamanlı Renklendirme)

Projede, kullanıcı arayüzüne entegre edilen gerçek zamanlı sözdizimi renklendirme mekanizması ile yazılan her karakter anlık olarak analiz edilmekte ve uygun sözcük türüne (token) göre renklendirilmektedir. Gerçek zamanlı renklendirme işlemi, “javax.swing.Timer” sınıfı ile düzenli aralıklarla (200 ms) tetiklenen bir işlem döngüsü üzerinden gerçekleşmektedir. Kullanıcının girdiği metin, bu zamanlayıcı tarafından sürekli takip edilip değişiklik algılandığında metin, önce

lexical analyzer (sözcüksel analizör) aracılığıyla tokenlara ayrılır. Daha sonra her token türü için önceden tanımlanmış bir stil uygulanarak metin üzerinde vurgulama yapılır.

Kullanıcının girdiği metin gerçek zamanlı olarak renklendirilmekte ve sözdizimi öğeleri ayırt edilebilmektedir. Bu işlem, StyledDocument yapısı aracılığıyla JTextPane bileşeni üzerinde uygulanmaktadır.

-Anahtar Kelimeler (Keywords): Mavi

-Tanımlayıcılar (Identifiers): Siyah

-Sayılar (Numbers): Mor

-Operatörler: Kırmızı

-Yorum Satırları: Gri

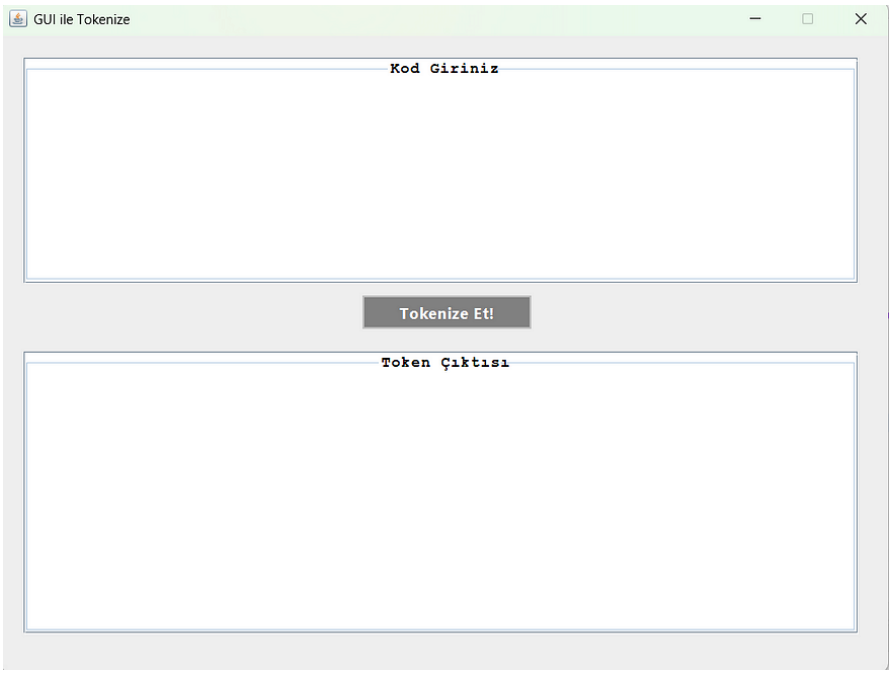
GUI Implementation (Arayüz Uygulaması)

Arayüz, Java Swing kütüphanesi kullanılarak geliştirilmiştir. Kullanıcı etkileşimleri JTextPane, JTextArea, JScrollPane, JButton gibi bileşenler ile sağlanmıştır.

JTextPane, kullanıcıdan kod girişi almakta ve sözcüksel analiz sonrası renklendirme burada uygulanmaktadır. Alt kısımda yer alan JTextArea, token listesini ve sözdizimi analiz çıktısını metin olarak göstermektedir.

Arka planda, DocumentListener kullanılarak JTextPane üzerindeki her değişiklik algılanmakta ve kısa bir süre sonra (Timer aracılığıyla) analiz ve renklendirme işlemi yeniden yapılmaktadır.

Uygulama, çok büyük metinlerde dahi hızlı çalışması için arayüz güncellemelerini SwingUtilities.invokeLater() içinde gerçekleştirerek kullanıcı deneyimini iyileştirmektedir. Ayrıca kullanıcı ekranını daha da zenginleştirmek için buton kullanılmıştır. Kullanıcı arayüzünde yer alan **“Tokenize Et”** butonu, kullanıcı tarafından yazılan kod üzerinde **sözcüksel (lexical)** ve **sözdizimsel (syntax + parse)** analiz işlemlerini başlatmak için kullanılacak şekilde tasarlanmıştır.



Sonuç ve Değerlendirme

Bu projede, Java programlama diline yönelik temel bir sözdizimi vurgulayıcı (syntax highlighter) geliştirilmiştir. Kullanıcıdan alınan kod girişleri, gerçek zamanlı olarak sözcüksel analizden geçirilmiş ve uygun renklendirme şemasıyla görsel olarak sunulmuştur. Aynı zamanda, temel düzeyde sözdizimsel analiz yapılarak kullanıcıya yapısal hatalara dair geri bildirim sağlanmıştır.