Sözcük Çözümleyici Uygulama Yazılımı Gereksinim Spesifikasyonu Belgesi

Ders Adı : Yazılım Kalite ve Test Süreci
Yürütücü İsmi : Prof. Dr. Oya Kalıpsız
Öğrenci Numara, İsim ve Soy İsimleri :
15011069 – Ömer Muhammed Demir
16011044 – İbrahim Furkan Erçelebi
16011105 – Buğra Karaca
16011702 – Mustafa Aydın
16011706 – Duygu Erduran

İçindekiler

1. Giriş

Bu belge, sözcük çözümlemek isteyen yazılım geliştiricilerinin, sözdizimlerini kontrol altında tutması, tüm dış davranışlarını, gereksinimlerini tanımlamak için hazırlanmıştır. Amaç, sözdizimi takibini yönetmek ve yazılımları daha iyi inceleyebilmektir.

2. Proje Tanımı

Üst düzey kodu, yani programlama dillerini, bilgisayarın anlayabileceği bir biçime - ikili koda - çevirme görevi, bir derleyicinin ana işidir. Basit bir şekilde konuşursak, derleyici 3 bölüme ayrılabilir:

- Lexical Analyzer – LA (Sözcük Çözümleyicisi)

- Syntax Analyzer – SA (Sözdizimi Çözümleyicisi)

- Semantic Analyzer – SMA (Semantik Çözümleyici)

Sözcük Analizcisi, kaynak kodu, kodu oluşturan sözcükler olan lexeme'lere ayırmaktan sorumludur. Tüm lexeme'leri ayırdıktan sonra, LA onları Token sınıflandırmasını kullanarak sınıflandırır. **Keyword'ler**, **Özel Semboller**, **Identifier'lar** ve **Operatörler** token örnekleridir. Derlenmiş kodun beyaz boşluklarını ve yorumlarını kaldırmak da Lexical Analyzer tarafından oynanan bir roldür. Bu işlemin çıktısı, lexeme'leri ve token sınıflandırmalarını içeren bir tablodur. Lexeme birimlerinin geçersiz yapıları olarak sözcük hataları, *ör. '12variableName'*, *'na;;me'*, ayrıca LA tarafından yakalanır.

Bu proje, Java'da yapılmış **basit** bir Sözcük Analizcisi'nin bir uygulamasıdır. Kullanıcının kodu yazabileceği ve belirteçlerini alabileceği bir GUI sağlar. Kodu bir dosyadan yüklemek ve analiz yapmak da mümkündür.

3. Gereksinimler

3.1. Fonksiyonel Gereksinimler

- Ondalık sayı değerlerini tanıma
- Karakter katarı değerlerini tanıma
- Tam sayı değerlerini tanıma
- Değişkenleri tanıma
- Tanımsızları tanıma
- Kod satırı içerisinde doğru sözcük ayrımı yapabilme
- Bir kod bloğu içerisindeki sözcük tiplerini doğru ayırıp gerçek sayı adedini verebilme
- Kod satırlarının, kod bloğu içerisindeki bütünlülüğünü koruyabilme

3.2. Performans Gereksinimleri

- Ardarda sık periyotlarla gelen kod satırlarıyla dolu kod bloğunu çözümleyebilme

3.3. Güvenlik Gereksinimleri

- Farklı kullanıcıların girişini test edebilme
- Yönetici olarak çalıştırma modunu görebilme

3.4. Arayüz Gereksinimleri

- Token'ların arayüzde düzgün görünmesi
- Kod bloğunda boşluk sınırını ölçmek

3.5. Uyumluluk Gereksinimleri

- Windows uyumluluğunu test etmek
- Linux uyumluluğunu test etmek

4. Paydaşlar

Mustafa Aydın (Proje Yöneticisi)
İbrahim Furkan Erçelebi (Birim Test Geliştirici)
Duygu Erduran (Sistem Analisti)
Ömer Muhammed Demir (Kara Kutu Test Geliştirici)
Buğra Karaca (Arayüz Test Geliştirici)