

Sözcük Çözümleyici Uygulama Yazılımı Test Sonuç Raporu



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.....	
1.1. Amaç.....	
1.2. Kapsam.....	
2. Kaynaklar.....	
3. Proje Tanımı.....	
4. Test Sonucu Raporlama Yaklaşımı.....	
4.1. Test Sonucu Raporlama.....	
4.2. Test Sonuçları Oluşturma.....	
4.3. Test Sonucu İzlenebilirlikleri.....	
5. Test Sonuçları.....	
6. Paydaşlar.....	

1. Giriş

Bu belge, sözcük çözümlemek isteyen yazılım geliştiricilerinin, sözdizimlerini kontrol altında tutması, tüm dış davranışlarını, gereksinimlerini tanımlamak için hazırlanmıştır.

1.1. Amaç

Amaç, sözdizimi takibini yönetmek ve yazılımları daha iyi inceleyebilmektir.

1.2. Kapsam

Bu belge, içerisinde bahsedilen “Sözcük Çözümleyici Uygulama Yazılımı” projesinde geliştirilecek olan yazılımın test eylemlerini kapsamaktadır.

2. Kaynaklar

Bu planın hazırlanmasında aşağıdaki kaynaklardan faydalanılmıştır.

- Depo Takip Yönetim Sistemi Test Planı
- İnsan Kaynakları Sistemi Projesi Yazılım Test Planı

3. 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.

Bu projenin Sözcük Çözümleyicisi, aşağıdaki token sınıflarını tanır:

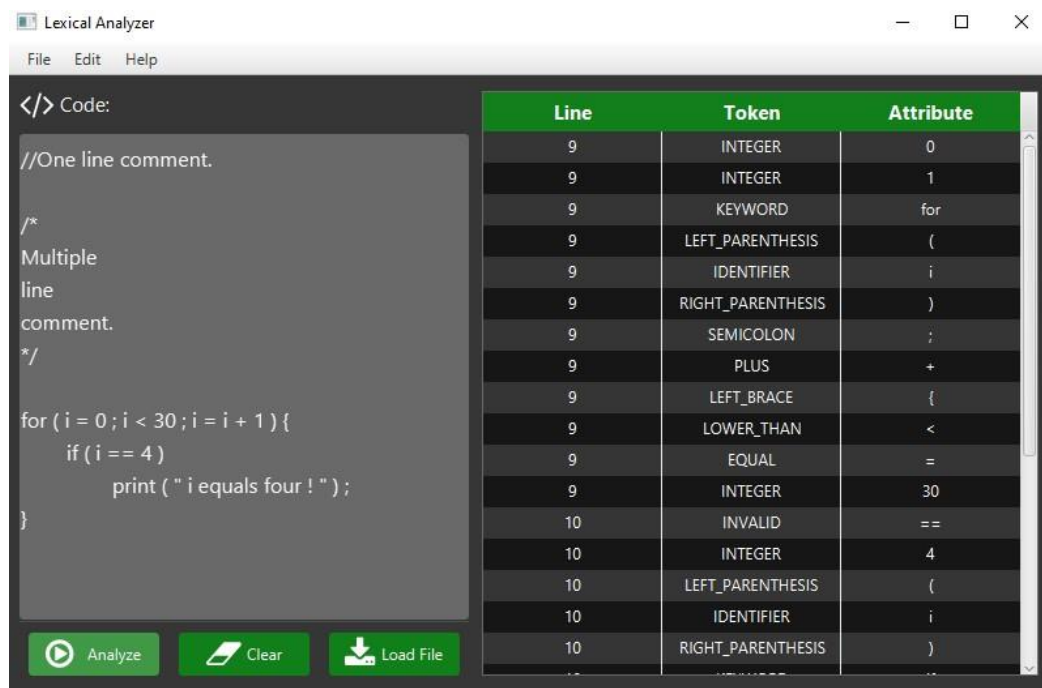
- ****IDENTIFIER**** - Değişken adları;
- ****STRING**** - Çift tırnak arasındaki kelimeler "";
- ****INTEGER**** - Noktasız sayı (.);
- ****FLOAT**** - Kayan nokta sayıları;
- ****PLUS**** - (+);
- ****MINUS**** - (-);
- ****TIMES**** - (*);
- ****DIVIDE**** - (/);
- ****KEYWORD**** - for, while, do, if, else, print, switch, case, default ve null;
- ****INVALID****;
- ****ASSIGN_OP**** - Atama operatörü (=);
- ****SEMICOLON**** - (;);
- ****LEFT_PARENTHESIS**** - '(';
- ****RIGHT_PARENTHESIS**** - ')';
- ****LEFT_BRACE**** - ({);
- ****RIGHT_BRACE**** - (});
- ****COMMA**** - (,);
- ****DOT**** - (.);
- ****DOTDOT**** - (..);
- ****COLON**** - (:);
- ****EQUAL**** - (==);
- ****LOWER_OR_EQUALS**** - (<=);
- ****GREATER_OR_EQUALS**** - (>=);
- ****NOT_EQUALS**** - (<>);
- ****GREATER_THAN**** - (>);

- ** LOWER_THAN ** - (<);

- ** AT_SIGN ** - (@).

***Not 1**:// ile başlayan cümleler veya /* */ arasındaki cümle parçaları yorum olarak kabul edilir ve çıktıda belirtilmez.*

***Not 2**:// Lexeme'ler, ayrılmış şeyler olarak tanınmak için en az bir boşluk (' ') ile ayrılmalıdır.*



Bu, Lexical Analyzer'ın nasıl uygulanabileceğini gösteren çok basit bir örnektir. Bu proje aynı zamanda çok güçlü ve kullanışlı bir araç olan Sonlu Durum Otomata'nın bir kullanım örneğidir.

4. Test Sonucu Raporlama Yaklaşımı

4.1. Test Sonucu Raporlama

Sözcük Çözümleyici Uygulama Yazılımı projesi kapsamındaki her test için alınacak çıktılar :

- Durum önceliği
- Test türü
- Yürütme türü
- Açıklama
- Test adımları
- Beklenen sonuç

4.2. Test Sonuçları Oluşturma

Sözcük Çözümleyici Uygulama Yazılımı projesi kapsamında gereksinim tabanlı test yaklaşımı ile testler gerçekleştirilmiştir. Her bir gereksinim için gereksinimi doğrulacayacak en asgari sayıda test durumu veya senaryoları yazılmıştır. Testler sırasında yazılan bu test durumları oluşturulmuştur. Her bir test durumunun oluşturulmasından elde edilen sonuçlar test sonuç tabloları üzerine girilmiştir. Eğer testlerde başarısızlık ortaya çıkmış ise bu durum da Hata Bildirim Formu kullanılarak hatanın düzeltilmesi için hata kaydı açılmıştır.

4.3. Test Sonucu İzlenebilirlikleri

Sözcük Çözümleyici Uygulama Yazılımı projesi test durumları ile gereksinimler arasındaki izlenebilirlik, gereksinim yönetim yazılımı üzerinden gerçekleştirilmiştir. Bu test durumlarına ait test sonuçları, test durumunun bir özniteliği olarak tanımlanmış ve direkt olarak test durumu ve test sonucu arasında izlenebilirlik oluşturulmuştur. Aynı zamanda gereksinim tabanlı test yaklaşımı benimsendiğinden test durumları ile gereksinimler arası izlenebilirlik de kurulmuştur.

5. Test Sonuçları

TEST DURUMU

Test durumu başlığı : *Ondalık sayı değerinin tespiti*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#01*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *'Float' olması gereken değerın görüntülenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.FLOAT, automaton.evaluate("100.888"));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *FLOAT*

```
setPrimitives();  
assertEquals(Token.FLOAT, automaton.evaluate("100.888"));
```

TEST DURUMU

Test durumu başlığı : *Ondalıklı sayı yerine farklı bir değişken alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#02*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *Normalde 'Float' olması gereken değer 'Invalid' olma durumunun görüntülenmek istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.FLOAT, automaton.evaluate(" 100.888"));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #02

Hata önem derecesi : *Kritik*

Hata açıklaması : *Başta boşluk olduğu için normalde 'Float' olması gerekirken 'Invalid' olarak atandı.*

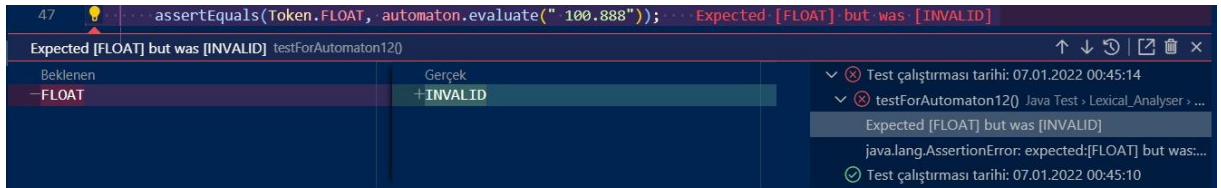
Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. assertEquals(Token.FLOAT, automaton.evaluate(" 100.888")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *FLOAT*

Gerçek sonuç : *INVALID*



TEST DURUMU

Test durumu başlığı : *Kelime katarı değerinin okunması*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#03*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *'String' olması gereken değerın görüntülenmesi istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. assertEquals(Token.STRING, automaton.evaluate("\\"This is String\\"")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *STRING*

```
setPrimitives();  
assertEquals(Token.STRING, automaton.evaluate("\\"This is String\\""));
```

TEST DURUMU

Test durumu başlığı : *Kelime katarı yerine farklı bir değişken alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#04*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *Normalde 'String' olması gereken değer 'Invalid' olma durumunun görüntülenmek istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. assertEquals(Token.STRING, automaton.evaluate("\\\\"This is String\\"")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #04

Hata önem derecesi : *Kritik*

Hata açıklaması : *String değer içinde ekstra parantez eklenmesi sonucunda geçersiz bir belirteç olarak tanımlandı..*

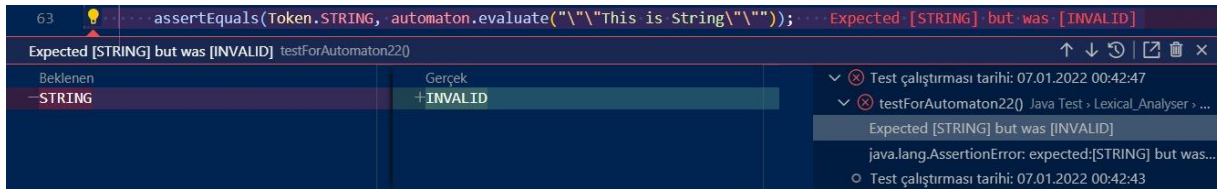
Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. assertEquals(Token.STRING, automaton.evaluate("\\\"This is String\\\"")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *STRING*

Gerçek sonuç : *INVALID*



TEST DURUMU

Test durumu başlığı : *Tam sayı girdi değerinin okunması*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#05*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *'Integer' olması gereken değerın görüntülenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.INTEGER, automaton.evaluate("1001"));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *INTEGER*

```
setPrimitives();  
assertEquals(Token.INTEGER, automaton.evaluate("1001"));
```

TEST DURUMU

Test durumu başlığı : *Tam sayı girdi yerine farklı bir değişken alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#06*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Mustafa Aydın*

Test açıklaması : *Normalde 'Integer' olması gereken değer 'Invalid' olma durumunun görüntülenmek istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.INTEGER, automaton.evaluate("1001 "));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #06

Hata önem derecesi : *Kritik*

Hata açıklaması : *Sonda boşluk olduğu için normalde 'Integer' olması gerekirken 'Invalid' olarak atandı.*

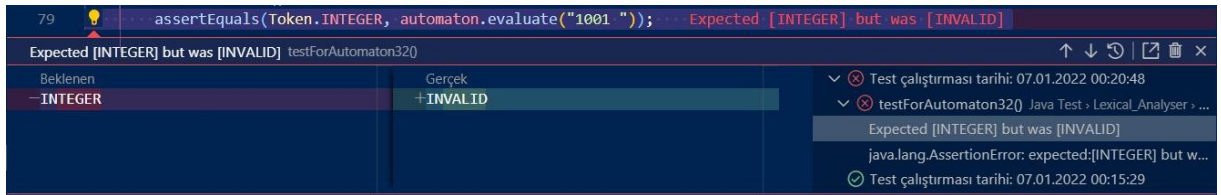
Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. assertEquals(Token.INTEGER, automaton.evaluate("1001 ")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *INTEGER*

Gerçek sonuç : *INVALID*



TEST DURUMU

Test durumu başlığı : *Tanımlayıcı (değişken) alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#07*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Duygu Erduran*

Test açıklaması : *'Identifier' olması gereken değerın görüntülenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. assertEquals(Token.IDENTIFIER, automaton.evaluate("aVar12389")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *IDENTIFIER*

```
setPrimitives();  
assertEquals(Token.IDENTIFIER, automaton.evaluate("aVar12389"));
```


TEST DURUMU

Test durumu başlığı : *Tanımlayıcı (değişken) yerine farklı bir şey alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#08*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Duygu Erduran*

Test açıklaması : *Normalde 'Identifier' olması gereken değer 'Invalid' olma durumunun görüntülenmek istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. assertEquals(Token.IDENTIFIER, automaton.evaluate("struct.substruct")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #08

Hata önem derecesi : *Kritik*

Hata açıklaması : *İç içe geçmiş tanımlamada geçersiz tanımlama ile karşılaşıldı.*

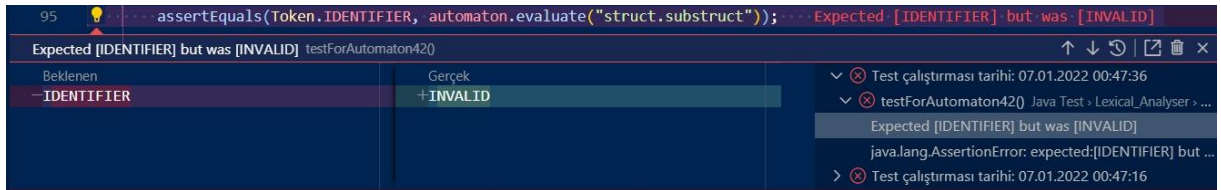
Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. assertEquals(Token.IDENTIFIER, automaton.evaluate("struct.substruct")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *IDENTIFIER*

Gerçek sonuç : *INVALID*



TEST DURUMU

Test durumu başlığı : *Tanımsız değer alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#09*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Duygu Erduran*

Test açıklaması : *'Invalid' olması gereken değerın görüntülenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.INVALID, automaton.evaluate("NMN3+%"));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *INVALID*

```
setPrimitives();  
assertEquals(Token.INVALID, automaton.evaluate("nmn3|+%" ));
```

TEST DURUMU

Test durumu başlığı : *Tanımsız yerine farklı bir şey alımı*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#10*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Duygu Erduran*

Test açıklaması : *Normalde 'Invalid' olması gereken değerlerin 'Identifier' olma durumunun görüntülenmek istenmesi için yapılan test.*

Test adımları :

1. VS Code açılır.
2. `assertEquals(Token.INVALID, automaton.evaluate("aVAr898"));` komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #10

Hata önem derecesi : *Kritik*

Hata açıklaması : *Normalde tanımsız olarak beklenen, aslında 'tanımlayıcı' olarak bulundu.*

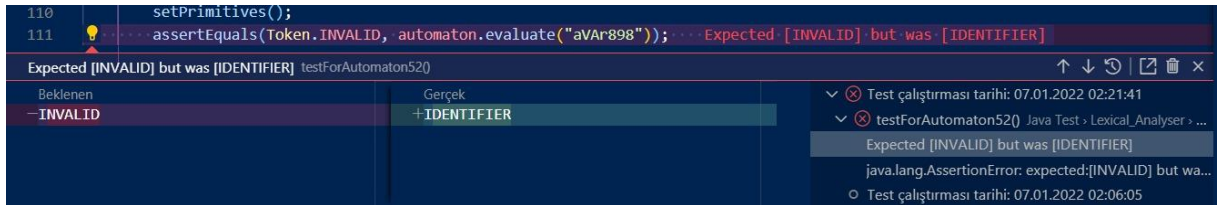
Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. assertEquals(Token.INVALID, automaton.evaluate("aVAr898")); komutu yazılır.
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *INVALID*

Gerçek sonuç : *IDENTIFIER*



TEST DURUMU

Test durumu başlığı : *Yanlış token değerinin gözlenmesi*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#11*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod içinde alınan tokenlerin hangi koşullarda yanlış olduğunu gözlemlemek.*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Token değeri*

HATA RAPORU

Test durumu numarası : #11

Hata önem derecesi : Yüksek

Hata açıklaması : *Kodda parse işlemi yapıldığında ilk olarak atanılacak değişken gelmesi lazımken , atama operatörü ile arada boşluk olmadığından 'geçersiz' uyarısı verilmekte.*

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Boşluk olmaksızın girilen iki token değeri*

Beklenen sonuç : IDENTIFIER

Gerçek sonuç : INVALID

```
208 lexical_Analyzer = new Lexical_Analyzer();
209 Map<Integer, String> lines = new HashMap<>();
210 lines.put(1, "val:=4val ++ 4");
211 List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
212 assertEquals(Token.IDENTIFIER, lexemes.get(0).getToken()); ... Expected: [IDENTIFIER] but was: [INVALID]
```

Beklenen	Gerçek
-IDENTIFIER	+INVALID

Expected [IDENTIFIER] but was [INVALID] testForLexicalAnalyzer50

Test çalıştırma tarihi: 07.01.2022 02:23:05

testForLexicalAnalyzer50() Java Test » Lexical_Analyser..

Expected [IDENTIFIER] but was [INVALID]

java.lang.AssertionError: expected:[IDENTIFIER] but ...

Test çalıştırma tarihi: 07.01.2022 02:21:41

TEST DURUMU

Test durumu başlığı : *Vermesi gereken token adedinin test edilmesi*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#12*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod içindeki her öğenin tespitini yaparken hangi koşullarda öge adedinin yanlış çıktığını görmek.*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Token (Öge sayısı)*

HATA RAPORU

Test durumu numarası : #12

Hata önem derecesi : *Yüksek*

Hata açıklaması : *Token tablosunda 17 tane token beklenirken arada boşluk olmadığından 4 tane token sıralandı.*

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. testForLexicalAnalyzer6()
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *17 token*

Gerçek sonuç : *4 token*

```
222
223     lexical_Analyzer = new Lexical_Analyzer();
224     Map<Integer, String> lines = new HashMap<>();
225     lines.put(1, "for(i;I<4 ;i++){");
226     lines.put(2, "print(\"Hello World\")");
227     List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
228     assertEquals(17, lexemes.size()); Expected [17] but was [4]
```

Beklenen	Gerçek
-17	+4

Expected [17] but was [4] testForLexicalAnalyzer6()

Test çalıştırma tarihi: 07.01.2022 01:41:23

testForLexicalAnalyzer6() Java Test » Lexical_Analyser...

Expected [17] but was [4]

java.lang.AssertionError: expected:[17] but was:[4] a...

Test çalıştırma tarihi: 07.01.2022 01:40:55

TEST DURUMU

Test durumu başlığı : *Kod satırları içindeki bütünlülüğü test etme*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#13*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod içindeki satırlar arasından birinin yanlış yazımı sonucu olabilecek hataları görmek*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata alımı*

HATA RAPORU

Test durumu numarası : #13

Hata önem derecesi : Yüksek

Hata açıklaması : 'print' komutunu içindeki String değişkeninin içi boşluklu olduğu için ilgili kod satırı (lexeme), String içinden parse edilmekte veya ayrı bir token olarak tutulmakta .

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. testForLexicalAnalyzer3()
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : Kod satırı dizisi

Beklenen sonuç : 10 token

Gerçek sonuç : 13 token

```
148 lexical_Analyzer = new Lexical_Analyzer();
149 Map<Integer, String> lines = new HashMap<>();
150 lines.put(1, "if ( i < 2 )");
151 lines.put(2, " print ( \"this not equals two\" )");
152 List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
153 assertEquals(10, lexemes.size()); ... Expected [10] but was [13]
```

Beklenen	Gerçek
-10	+13

```
154 assertEquals(Token.NOT_EQUALS, lexemes.get(0).getToken());
155 assertEquals(Token.INTEGER, lexemes.get(1).getToken());
156 assertEquals(Token.LEFT_PARENTHESIS, lexemes.get(2).getToken());
157 assertEquals(Token.IDENTIFIER, lexemes.get(3).getToken());
158 assertEquals(Token.RIGHT_PARENTHESIS, lexemes.get(4).getToken());
159 assertEquals(Token.KEYWORD, lexemes.get(5).getToken());
160 assertEquals(Token.KEYWORD, lexemes.get(6).getToken());
161 assertEquals(Token.STRING, lexemes.get(7).getToken());
162 assertEquals(Token.LEFT_PARENTHESIS, lexemes.get(8).getToken());
163 assertEquals(Token.RIGHT_PARENTHESIS, lexemes.get(9).getToken());
164
```

Expected [10] but was [13] testForLexicalAnalyzer3()

Test çalıştırma tarihi: 07.01.2022 02:30:04

testForLexicalAnalyzer3() Java Test - Lexical_Analyser...

Expected [10] but was [13]

java.lang.AssertionError: expected:[10] but was:[13] ...

Test çalıştırma tarihi: 07.01.2022 02:24:59

TEST DURUMU

Test durumu başlığı : *Tüm fonksiyonların kara kutu kontrolü*

Test durumu önceliği : *Yüksek*

Test durumu numarası : #14

Test türü : *İşlevsellik*

Test yürütme türü : *Manuel*

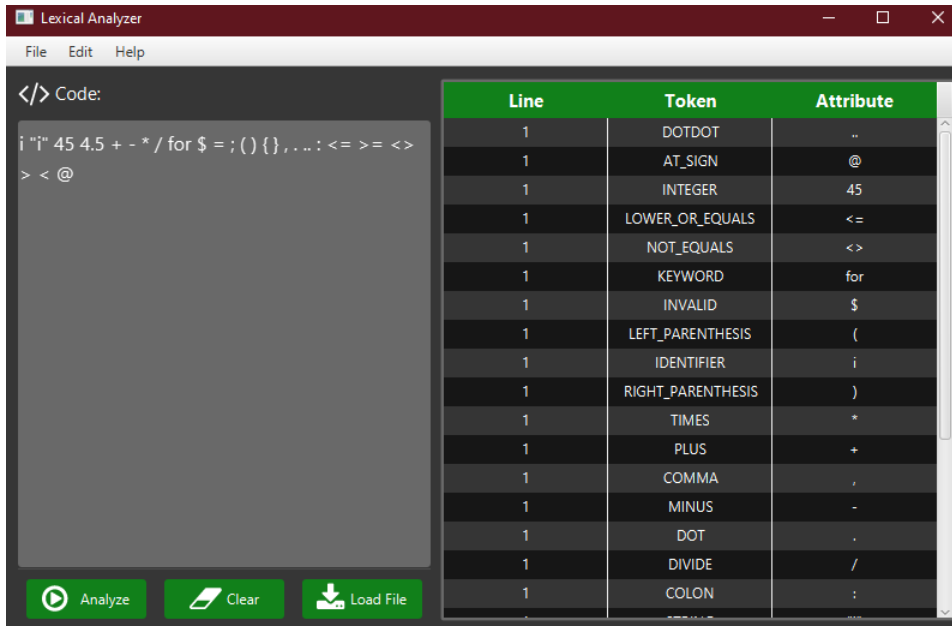
Testi yapan : Ömer Demir

Test açıklaması : *Tüm token'ların çalıştığının elle kontrolü*

Test adımları :

- Aralarına boşluk koyarak token'lar yazılır
- Analyse butonuna basılır

Beklenen sonuç : *Girilen tüm token'ların görüntülenmesi*



Line	Token	Attribute
1	DOTDOT	..
1	AT_SIGN	@
1	INTEGER	45
1	LOWER_OR_EQUALS	<=
1	NOT_EQUALS	<>
1	KEYWORD	for
1	INVALID	\$
1	LEFT_PARENTHESIS	(
1	IDENTIFIER	i
1	RIGHT_PARENTHESIS)
1	TIMES	*
1	PLUS	+
1	COMMA	,
1	MINUS	-
1	DOT	.
1	DIVIDE	/
1	COLON	:

TEST DURUMU

Test durumu başlığı : *Kod bloğu analizi #1*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#15*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod içindeki kod bloklarının tek tek analizi*

Test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Kodun analizi*

```
setPrimitives();
Map<Integer, String> lines = new HashMap<>();
lines.put(1, "var := 5");
List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
assertEquals(3, lexemes.size());
assertEquals(Token.ASSIGNMENT_OPERATOR, lexemes.get(0).getToken());
assertEquals(Token.INTEGER, lexemes.get(1).getToken());
assertEquals(Token.IDENTIFIER, lexemes.get(2).getToken());
```

TEST DURUMU

Test durumu başlığı : *Kod bloğu analizi #2*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#16*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod içindeki kod bloklarının tek tek analizi*

Test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Kodun analizi*

```
setPrimitives();
Map<Integer, String> lines = new HashMap<>();
lines.put(1, "var := 3 + val");
List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
assertEquals(5, lexemes.size());
assertEquals(Token.IDENTIFIER, lexemes.get(0).getToken());
assertEquals(Token.ASSIGNMENT_OPERATOR, lexemes.get(1).getToken());
assertEquals(Token.INTEGER, lexemes.get(2).getToken());
assertEquals(Token.IDENTIFIER, lexemes.get(3).getToken());
assertEquals(Token.PLUS, lexemes.get(4).getToken());
```

TEST DURUMU

Test durumu başlığı : *Kod bloğu analizi #3*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#17*

Test türü : *İşlevsellik*

Test yürütme türü : *Otomatik*

Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Yazılan kod bloğu içindeki kod satırlarının tek tek analizi*

Beklenen sonuç : *Kodun analizi*

```
lexical_Analyzer = new Lexical_Analyzer();
Map<Integer, String> lines = new HashMap<>();
lines.put(1, "for ( i := 1 ; I < 4 ; i + + ) {");
lines.put(2, "    if ( I / 2 = 1 )");
lines.put(3, "        print ( \"valuesisodd\" ) }");
List<Lexeme> lexemes = lexical_Analyzer.analyzeCode(lines);
assertEquals(25, lexemes.size());
assertEquals(Token.INTEGER, lexemes.get(0).getToken());
assertEquals(Token.ASSIGNMENT_OPERATOR, lexemes.get(1).getToken());
assertEquals(Token.INTEGER, lexemes.get(2).getToken());
assertEquals(Token.KEYWORD, lexemes.get(3).getToken());
assertEquals(Token.LEFT_PARENTHESIS, lexemes.get(4).getToken());
assertEquals(Token.IDENTIFIER, lexemes.get(5).getToken());
assertEquals(Token.IDENTIFIER, lexemes.get(6).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, lexemes.get(7).getToken());
assertEquals(Token.SEMICOLON, lexemes.get(8).getToken());
assertEquals(Token.PLUS, lexemes.get(9).getToken());
assertEquals(Token.LEFT_BRACE, lexemes.get(10).getToken());
assertEquals(Token.LOWER_THAN, lexemes.get(11).getToken());
assertEquals(Token.INTEGER, lexemes.get(12).getToken());
assertEquals(Token.INTEGER, lexemes.get(13).getToken());
assertEquals(Token.LEFT_PARENTHESIS, lexemes.get(14).getToken());
assertEquals(Token.IDENTIFIER, lexemes.get(15).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, lexemes.get(16).getToken());
assertEquals(Token.KEYWORD, lexemes.get(17).getToken());
assertEquals(Token.EQUAL, lexemes.get(18).getToken());
assertEquals(Token.DIVIDE, lexemes.get(19).getToken());
assertEquals(Token.KEYWORD, lexemes.get(20).getToken());
assertEquals(Token.LEFT_PARENTHESIS, lexemes.get(21).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, lexemes.get(22).getToken());
assertEquals(Token.STRING, lexemes.get(23).getToken());
assertEquals(Token.RIGHT_BRACE, lexemes.get(24).getToken());
```


TEST DURUMU

Test durumu başlığı : *Token'ların arayüzde görünümü testi #1*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#18*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : *Düzgün yazılmış token'lerin kullanıcı arayüzüne nasıl yansıdığını gözlemleme*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Kod bloğunun çalışması*

```
clickOn("#codeTextArea");
Thread.sleep(10000);
write("i := 1");
Thread.sleep(10000);
clickOn("#btnAnalyze");
Thread.sleep(10000);
TableView<Lexeme> tableView = lookup("#tokensTable").query();
assertEquals(Token.INTEGER, tableView.getItems().get(0).getToken());
assertEquals(Token.ASSIGNMENT_OPERATOR, tableView.getItems().get(1).getToken());
assertEquals(Token.IDENTIFIER, tableView.getItems().get(2).getToken());
```


TEST DURUMU

Test durumu başlığı : *Boşluk olmadan yazılan token'lerin görünümü*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#19*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : *Hiçbir şekilde arasına boşluk koyulmayan token'lerin kullanıcı arayüzüne nasıl yansıdığını gözlemleme*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #19

Hata önem derecesi : *Kritik*

Hata açıklaması : *Durum sorgulama kısmında sayılar, eşitlik ve çıkarma operatörü birleşik olduğundan token tablosunda hepsi bir bütün olarak tanımsız gözükmekte olup token tablosunun ilk başında sol parantez bulunmaktadır.*

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *ANY TOKEN*

Gerçek sonuç : *INVALID*

```
111      clickOn("#codeTextArea");
112      write("if ( 3-1.0=2 ) \n");
113      write("    print( \"one\" )");
114      clickOn("#btnAnalyze");
115      TableView<Lexeme> tableView = lookup("#tokensTable").query();
116      assertEquals(Token.INTEGER, tableView.getItems().get(0).getToken()); ... Expected [INTEGER] but was [LEFT_PARENTHESIS]
```

Beklenen	Gerçek
-INTEGER	+LEFT_PARENTHESIS

Expected [INTEGER] but was [LEFT_PARENTHESIS] testAnalyzer50

Test çalıştırma tarihi: 07.01.2022 02:45:39

testAnalyzer50 Java Test > Lexical_Analyser > br.ufop.t...

Expected [INTEGER] but was [LEFT_PARENTHESIS]

java.lang.AssertionError: expected:[INTEGER] but w...

Test çalıştırma tarihi: 07.01.2022 02:40:30

```
117      assertEquals(Token.INTEGER, tableView.getItems().get(1).getToken());
118      assertEquals(Token.FLOAT, tableView.getItems().get(2).getToken());
119      assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(3).getToken());
120      assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(4).getToken());
121      assertEquals(Token.TIMES, tableView.getItems().get(5).getToken());
122      assertEquals(Token.KEYWORD, tableView.getItems().get(6).getToken());
123      assertEquals(Token.EQUAL, tableView.getItems().get(7).getToken());
124      assertEquals(Token.KEYWORD, tableView.getItems().get(8).getToken());
125      assertEquals(Token.STRING, tableView.getItems().get(9).getToken());
126      assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(10).getToken());
127      assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(11).getToken());
```

TEST DURUMU

Test durumu başlığı : *Token'ların arayüzde görünümü testi #2*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#20*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : *Düzgün yazılmış token'lerin kullanıcı arayüzüne nasıl yansıdığını gözlemleme*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Kod bloğu*

```
clickOn("#codeTextArea");
write("if ( 2 * 3 = 6 ) \n    print ( \"six\" )");
clickOn("#btnAnalyze");
TableView<Lexeme> tableView = lookup("#tokensTable").query();
assertEquals(Token.INTEGER, tableView.getItems().get(0).getToken());
assertEquals(Token.INTEGER, tableView.getItems().get(1).getToken());
assertEquals(Token.INTEGER, tableView.getItems().get(2).getToken());
assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(3).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(4).getToken());
assertEquals(Token.TIMES, tableView.getItems().get(5).getToken());
assertEquals(Token.KEYWORD, tableView.getItems().get(6).getToken());
assertEquals(Token.EQUAL, tableView.getItems().get(7).getToken());
assertEquals(Token.KEYWORD, tableView.getItems().get(8).getToken());
assertEquals(Token.STRING, tableView.getItems().get(9).getToken());
assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(10).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(11).getToken());
```

TEST DURUMU

Test durumu başlığı : *Tek bir boşluksuz durumda hata kontrolü*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#21*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : *Sadece bir tane boşluk olması durumunda hata olup olmaması kontrolü*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #21

Hata önem derecesi : *Kritik*

Hata açıklaması : *Normalde iki atama operatörü token'lar listelenmesi gerekirken operatörle atanan arasında boşluk olmadığından iki tane geçersiz token bulunmakta.*

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. testAnalyzer4()
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *ANY TOKEN*

Gerçek sonuç : *INVALID*

```
99      clickOn("#codeTextArea");
100     write("i:= 89nns");
101     clickOn("#btnAnalyze");
102     TableView<Lexeme> tableView = lookup("#tokensTable").query();
103     assertEquals(Token.IDENTIFIER, tableView.getItems().get(0).getToken());...Expected: [IDENTIFIER] but was: [INVALID]
```

Beklenen	Gerçek
-IDENTIFIER	+INVALID

Expected [IDENTIFIER] but was [INVALID] testAnalyzer4()

Test çalıştırma tarihi: 07.01.2022 02:40:30

testAnalyzer4() Java Test > Lexical_Analyser > br.ufop.t...

Expected [IDENTIFIER] but was [INVALID]

java.lang.AssertionError: expected:[IDENTIFIER] but ...

Test çalıştırma tarihi: 07.01.2022 02:30:04

```
104     assertEquals(Token.ASSIGNMENT_OPERATOR, tableView.getItems().get(1).getToken());
105     assertEquals(Token.IDENTIFIER, tableView.getItems().get(2).getToken());
```

TEST DURUMU

Test durumu başlığı : *Token'ların arayüzde görünümü testi #3*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#22*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : Düzgün yazılmış token'lerin kullanıcı arayüzüne nasıl yansıdığını gözlemleme

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : Kod bloğu

```
clickOn("#codeTextArea");
write("while ( true ) {\n    i ++ ;\n    print ( 3 * 4 ) ; }");
clickOn("#btnAnalyze");
Thread.sleep(10000);
TableView<Lexeme> tableView = lookup("#tokensTable").query();
assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(0).getToken());
assertEquals(Token.IDENTIFIER, tableView.getItems().get(1).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(2).getToken());
assertEquals(Token.LEFT_BRACE, tableView.getItems().get(3).getToken());
assertEquals(Token.KEYWORD, tableView.getItems().get(4).getToken());
assertEquals(Token.IDENTIFIER, tableView.getItems().get(5).getToken());
assertEquals(Token.PLUS, tableView.getItems().get(6).getToken());
assertEquals(Token.SEMICOLON, tableView.getItems().get(7).getToken());
assertEquals(Token.KEYWORD, tableView.getItems().get(8).getToken());
assertEquals(Token.INTEGER, tableView.getItems().get(9).getToken());
assertEquals(Token.INTEGER, tableView.getItems().get(10).getToken());
assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(11).getToken());
assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(12).getToken());
assertEquals(Token.TIMES, tableView.getItems().get(13).getToken());
assertEquals(Token.SEMICOLON, tableView.getItems().get(14).getToken());
assertEquals(Token.RIGHT_BRACE, tableView.getItems().get(15).getToken());
```

TEST DURUMU

Test durumu başlığı : *Aralıksız parantez kontrolü*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#23*

Test türü : *Arayüz*

Test yürütme türü : *Otomatik*

Testi yapan : *Buğra Karaca*

Test açıklaması : *Parantezler arası aralık olmaması durumunu ölçme*

Test adımları :

1. VS Code açılır.
2. İlgili kodun yazılması
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Hata vermesi*

HATA RAPORU

Test durumu numarası : #23

Hata önem derecesi : *Kritik*

Hata açıklaması : *Koşullu döngüdeki koşul, aralıksız parantezlerle ifade edildiği için bütün olarak tanımsız olması sebebiyle token tablosunda ilk alınan karakterin yuvarlak parantez yerine köşeli parantez olması.*

Hatayı oluşturmak için test adımları :

1. VS Code açılır.
2. testAnalyzer6()
3. Çalıştırılıp sonuçlar yazılır.

Test verisi : *Kod satırı*

Beklenen sonuç : *LEFT_PARENTHESIS*

Gerçek sonuç : *INVALID*

```
132 public void testAnalyzer6() {
133     clickOn("#codeTextArea");
134     write("while(false) {\n 4++;\n print (null); }");
135     clickOn("#btnAnalyze");
136     TableView<Lexeme> tableView = lookup("#tokensTable").query();
137     assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(0).getToken()); Expected [LEFT_PARENTHESIS] but was [LEFT_BRACE] testAnalyzer6()
Expected [LEFT_PARENTHESIS] but was [LEFT_BRACE] testAnalyzer6()
Beklenen Gerçek
-LEFT_PARENTHESIS +LEFT_BRACE
Test çalıştırma tarihi: 07.01.2022 02:50:57
testAnalyzer6() Java Test > Lexical_Analyser > br.ufop.t...
Expected [LEFT_PARENTHESIS] but was [LEFT_BRACE]
java.lang.AssertionError: expected:[LEFT_PARENTHESIS] but was:[LEFT_BRACE]
Test çalıştırma tarihi: 07.01.2022 02:45:39
138 assertEquals(Token.IDENTIFIER, tableView.getItems().get(1).getToken());
139 assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(2).getToken());
140 assertEquals(Token.LEFT_BRACE, tableView.getItems().get(3).getToken());
141 assertEquals(Token.KEYWORD, tableView.getItems().get(4).getToken());
142 assertEquals(Token.INTEGER, tableView.getItems().get(5).getToken());
143 assertEquals(Token.PLUS, tableView.getItems().get(6).getToken());
144 assertEquals(Token.SEMICOLON, tableView.getItems().get(7).getToken());
145 assertEquals(Token.KEYWORD, tableView.getItems().get(8).getToken());
146 assertEquals(Token.KEYWORD, tableView.getItems().get(9).getToken());
147 assertEquals(Token.LEFT_PARENTHESIS, tableView.getItems().get(10).getToken());
148 assertEquals(Token.RIGHT_PARENTHESIS, tableView.getItems().get(11).getToken());
```


TEST DURUMU

Test durumu başlığı : *Tüm butonların kontrolü*

Test durumu önceliği : *Orta*

Test durumu numarası : #24

Test türü : *Arayüz*

Test yürütme türü : *Manuel*

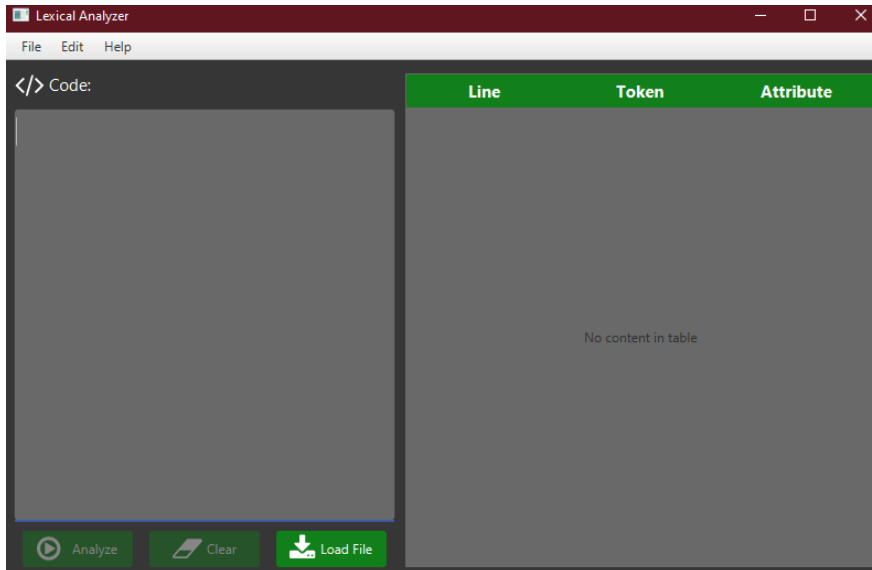
Testi yapan : Ömer Demir

Test açıklaması : *Tüm butonların, gerekli işlevleri yerine getirip getirmediğini ölçme*

Test adımları :

- Tüm butonları tek tek kontrol etmek
- Sonuçları kaydetmek

Beklenen sonuç : *Butonların çalışması*



TEST DURUMU

Test durumu başlığı : *Uzun döngüde performans kontrolü #1*

Test durumu önceliği : *Yüksek*

Test durumu numarası : #25

Test türü : *Performans*

Test yürütme türü : Otomatik

Testi yapan : Furkan Erçelebi

Test açıklaması : Sürekli girdi yapılması halinde programın hata verip vermeyeceğinin kontrolü

Test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Token'ların gösterimi*

[illegible]

TEST DURUMU

Test durumu başlığı : *Uzun döngüde performans kontrolü #3*

Test durumu önceliği : *Yüksek*

Test durumu numarası : *#27*

Test türü : *Performans*

Test yürütme türü : *Otomatik*

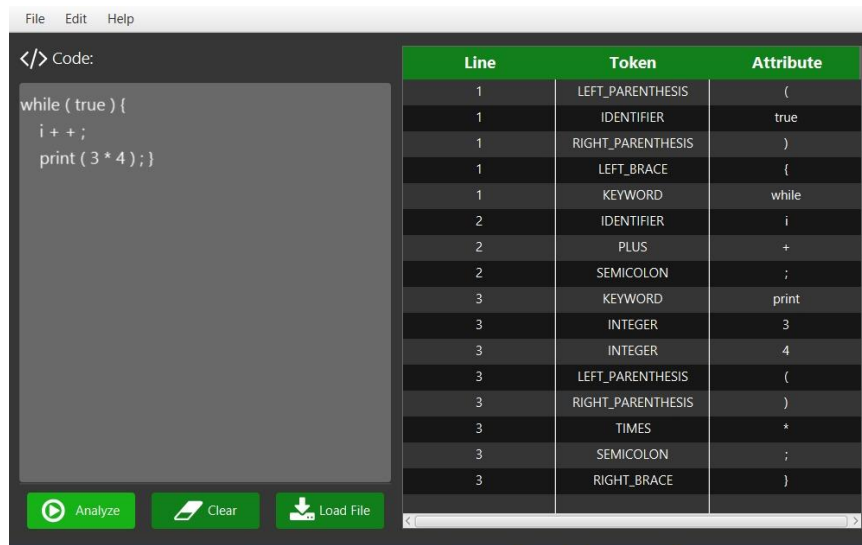
Testi yapan : *Furkan Erçelebi*

Test açıklaması : *Sürekli girdi yapılması halinde programın hata verip vermeyeceğinin kontrolü*

Test adımları :

1. VS Code açılır.
2. İlgili kod yazılır
3. Çalıştırılıp sonuçlar yazılır.

Beklenen sonuç : *Token'ların gösterimi*



The screenshot shows a code editor with a while loop and its tokenization results. The code is as follows:

```
while ( true ) {  
    i ++ ;  
    print ( 3 * 4 );  
}
```

The tokenization results are shown in a table with three columns: Line, Token, and Attribute.

Line	Token	Attribute
1	LEFT_PARENTHESIS	(
1	IDENTIFIER	true
1	RIGHT_PARENTHESIS)
1	LEFT_BRACE	{
1	KEYWORD	while
2	IDENTIFIER	i
2	PLUS	+
2	SEMICOLON	;
3	KEYWORD	print
3	INTEGER	3
3	INTEGER	4
3	LEFT_PARENTHESIS	(
3	RIGHT_PARENTHESIS)
3	TIMES	*
3	SEMICOLON	;
3	RIGHT_BRACE	}

TEST DURUMU

Test durumu başlığı : *Sürekli Ctrl+V ile aynı kodun çalıştırılması*

Test durumu önceliği : *Orta*

Test durumu numarası : *#28*

Test türü : *Performans*

Test yürütme türü : *Manuel*

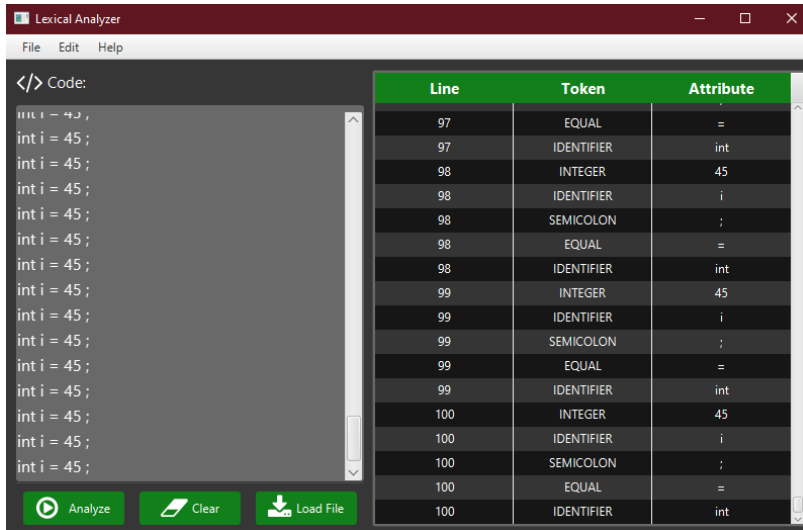
Testi yapan : *Ömer Demir*

Test açıklaması : *Sürekli aynı kodu uzun süre tekrar ederek çalıştırma işlemi*

Test adımları :

- Belli sayıda kod satırı kopyalanır
- Belli sayıda kopyalanan satırlar, yapıştırılır
- Analiz edilir.
-

Beklenen sonuç : *Token'ların gösterimi*



The screenshot shows the Lexical Analyzer application interface. On the left, the 'Code:' area contains the input code: `int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;
int i = 45 ;`. On the right, the 'Token' table displays the results of the lexical analysis.

Line	Token	Attribute
97	EQUAL	=
97	IDENTIFIER	int
98	INTEGER	45
98	IDENTIFIER	i
98	SEMICOLON	;
98	EQUAL	=
98	IDENTIFIER	int
99	INTEGER	45
99	IDENTIFIER	i
99	SEMICOLON	;
99	EQUAL	=
99	IDENTIFIER	int
100	INTEGER	45
100	IDENTIFIER	i
100	SEMICOLON	;
100	EQUAL	=
100	IDENTIFIER	int

TEST DURUMU

Test durumu başlığı : *Uygulamayı Windows'ta çalıştırma*

Test durumu önceliği : *Alçak*

Test durumu numarası : *#29*

Test türü : *Uyumluluk*

Test yürütme türü : *Manuel*

Testi yapan : *Ömer Demir*

Test açıklaması : *Uygulamanın Windows işletim sisteminde çalıştırılması*

Test adımları :

- Uygulamayı Windows ortamında kurmak
- Gradle'ı kurmak
- Gerekli ayarları yapmak
- Çalıştırmak

Beklenen sonuç : *Uygulamanın çalışır olması*

TEST DURUMU

Test durumu başlığı : *Uygulamayı Ubuntu'ta çalıştırma*

Test durumu önceliği : *Alçak*

Test durumu numarası : *#30*

Test türü : *Uyumluluk*

Test yürütme türü : *Manuel*

Testi yapan : *Ömer Demir*

Test açıklaması : *Uygulamanın Linux Ubuntu işletim sisteminde çalıştırılması*

Test adımları :

- Uygulamayı Windows ortamında kurmak
- Gradle'ı kurmak
- Gerekli ayarları yapmak
- Çalıştırmak

Beklenen sonuç : *Uygulamanın çalışır olması*

TEST DURUMU

Test durumu başlığı : *Uygulamayı Kali'de çalıştırma*

Test durumu önceliği : *Alçak*

Test durumu numarası : *#31*

Test türü : *Uyumluluk*

Test yürütme türü : *Manuel*

Testi yapan : *Ömer Demir*

Test açıklaması : *Uygulamanın Linux Kali işletim sisteminde çalıştırılması*

Test adımları :

- Uygulamayı Windows ortamında kurmak
- Gradle'ı kurmak
- Gerekli ayarları yapmak
- Çalıştırmak

Beklenen sonuç : *Uygulamanın çalışır olması*

TEST DURUMU

Test durumu başlığı : *Başka bir kullanıcı girişi*

Test durumu önceliği : *Alçak*

Test durumu numarası : *#32*

Test türü : *Güvenlik*

Test yürütme türü : *Manuel*

Testi yapan : *Ömer Demir*

Test açıklaması : *Uygulamanın başka bir yerel hesap üzerinden çalıştırılması*

Test adımları :

- Farklı bir hesapta oturum açmak
- Uygulamayı çalıştırmak

Beklenen sonuç : *Uygulamanın çalışması*

TEST DURUMU

Test durumu başlığı : *Yönetici çalıştırması*

Test durumu önceliği : *Alçak*

Test durumu numarası : *#33*

Test türü : *Güvenlik*

Test yürütme türü : *Manuel*

Testi yapan : *Ömer Demir*

Test açıklaması : *Uygulamanın başka yönetici olarak çalıştırılması*

Test adımları :

- Uygulamayı yönetici özelliğe sahip bir hesaptan yönetici olarak çalıştırmak
- Uygulamayı çalıştırmak

Beklenen sonuç : *Uygulamanın çalışması*

6. Paydařlar

Mustafa Aydın (Proje Yöneticisi)

İbrahim Furkan Erçelebi (Birim Test Geliřtirici)

Duygu Erduran (Sistem Analisti)

Ömer Muhammed Demir (Kara Kutu Test Geliřtirici)

Buğra Karaca (Arayüz Test Geliřtirici)