T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

Java Konsol Uygulaması: .java Dosyasındaki Fonksiyon Yorumlarını Ayıklama

G191210027 - Selcan NARİN

SAKARYA

Nisan, 2023

Programlama Dillerinin Prensipleri Dersi

Java Konsol Uygulaması: .java Dosyasındaki Fonksiyon Yorumlarını Ayıklama

Selcan Narin

G191210027 – 1B

Özet
Bu ödev, konsoldan verilen bir Java dosyasındaki fonksiyonlara ait tek satırlı, çok satırlı ve javadoc yorum satırlarını tespit eder ve her yorum türünü ayrı dosyalara fonksiyon isimlerine göre ayrı ayrı kaydetmeyi amaçlar.

Ödevin çözümünde Regex Pattern kullanarak verilen fonksiyonların isim, gövde ve gövdesine ek olarak parametre kısmında da yorum bulunması ihtimaline karşın parametreler gruplarına böldüm. Fonksiyonları match ettim ve while döngüsü içinde find() ile dolaştım. Fonksiyonların üzerinde bulunan javadoclar için ayrı bir pattern oluşturdum. Fonksiyondaki tüm match edilen javadocları tespit ettim. Fonksiyonun parametre ve gövdesini içeren kısım için de group() ile functionCodeBlock String' ine atadım. Bu String' i kullanarak kodun içerisindeki bütün yorumları String' in uzunluğu kadar dönen bir for döngüsünde StringBuilder nesnesini ve flaglar kullanarak dolaştım. insideNestedComments flagı ile iç içe geçmiş, detaylı kontrol isteyen yorum satırlarını ayırdım. Bütün yorumları tespit ettikçe listelerine ekledim ve sonunda dosyalarına yazdırdım.

Anahtar Kelimeler: Regex, Pattern, Matcher, StringBuilder

1. GELİŞTİRİLEN YAZILIM

Kod, Java kod dosyalarındaki yorum sayısını saymak için tasarlanmıştır. CommentCounter adında bir sınıftır ve main yöntemi içerir.

Dosya Okuma:

İlk olarak, Java kodunu okumak için kullanılacak dosya yolunu filename adlı bir değişkene atar. Bu dosya yolu, kodu yorumlayacağı Java dosyasının ilk argümanı olarak sağlanır. Sonra, ReadFile sınıfındaki readFile yöntemi kullanılarak Java dosyasının içeriği okunur ve bir dize olarak javaCode değişkenine atanır.

Regex Pattern Oluşturma:

Ardından, bir Pattern nesnesi kullanarak Java kodundaki fonksiyon adlarını eşleştirmek için bir düzenli ifade(regex) tanımlanır.

/\\b(public|private|protected|final|static|synchronized|abstract|native)?\\s*(class)?\\s*(\\w+\\s+)*(\\w+)\\s*\\(([\\s\\S]*?)\\)\\s*(throws [\\s\\S]+)?\\s*\\{([\\s\\S]*?)\\}/gm

- /\b - Sözcük sınırının başlangıcıyla eşleşir
- (public|private|protected|final|static|synchronized|abstract|native)? - Listelenen erişim değiştiricilerinden herhangi biriyle eşleşir (isteğe bağlı)
- \s* - Herhangi bir boşluk karakteriyle eşleşir (isteğe bağlı)
- (class)? - sınıf" anahtar sözcüğüyle eşleşir (isteğe bağlı)
- \s* - Herhangi bir boşluk karakteriyle eşleşir (isteğe bağlı)
- (\w+\s+)* - Ardından boşluk gelen herhangi bir sayıda sözcükle eşleşir (isteğe bağlı)
- (\w+) - Sınıf adıyla eşleşir -- **group(4)=>fonksiyon ismi**
- \s* - Herhangi bir boşluk karakteriyle eşleşir (isteğe bağlı)
- \(- Bir açılış paranteziyle eşleşir
- ([\s\S]*?) Parantez içindeki herhangi bir sayıda karakterle (yeni satırlar dahil) eşleşir (isteğe bağlı) – **group(5)=>parametreler**
- \) - Kapatma paranteziyle eşleşir
- \s* - Herhangi bir boşluk karakteriyle eşleşir (isteğe bağlı)
- (throws [\s\S]+)? - thrown istisnalarla eşleşir (isteğe bağlı)
- \s* - Herhangi bir boşluk karakteriyle eşleşir (isteğe bağlı)
- \{ - Süslü parantezin başlangıcıyla eşlenir.
- ([\s\S]*?) – Süslü parantezin içindeki herhangi bir sayıda karakteri (yeni satırlar dahil) eşleştirir (isteğe bağlı) -- **group(7)=>gövde**
- \} - Süslü parantezin kapanışıyla eşlenir.

Fonksiyonların üzerinde yer alan JavaDoc yorumlarını eşleştirmek için başka bir Pattern nesnesi daha tanımlanır.

- ** ile /** karakterleri birebir eşleşir.
- .*? herhangi bir karakterle (yeni satır hariç) sıfır veya daha fazla, olabildiğince az eşleşir.
- */ ile */ /** karakterleri birebir eşleşir
- Pattern.DOTALL bayrağı, " ." karakter, yeni satırlar da dahil olmak üzere herhangi bir karakterle eşleşir.

Dosya İşlemleri:

Üç farklı çıktı dosyası oluşturulur: tek satırlık yorumlar için "teksatir.txt", çok satırlı yorumlar için "coksatir.txt" ve JavaDoc yorumları için "javadoc.txt". Dosyaların yaratılıp yaratılmadığı kontrol edilir ve mevcut değilse, yaratılır.

Üç farklı BufferedWriter nesnesi yaratılır, her biri bir çıktı dosyasına yazmak için kullanılır.

Fonksiyon ve üzerindeki Javadoclarını Tespit Etme:

Ardından, Matcher nesnesi ve Group(4) methodu kullanarak Java kodundaki tüm fonksiyon adlarını eşleştirir.

Fonksiyon adlarını eşleştirdikten sonra, önceki fonksiyonun bitiş indeksini ve yorum sayıları için bazı değişkenler tanımlar.

Sonra, find() ile tüm fonksiyon eşleştirmeleri için bir döngü oluşturulur.

Döngü her bir fonksiyonu sırayla ele alır ve önceki işlevin bitiş noktasından başlayarak şu anki işlevin başlangıcına kadar olan alanda bulunan JavaDoc yorumlarını bulmak için bir Matcher nesnesi daha kullanır.

Fonksiyonun içindeki Yorumları Ayırma:

Fonksiyonun, parametre listesi ve tüm kod bloğu bir String olarak alınır.

Bu kod bloğundaki tüm yorumları bulmak için, önce "single-line" yorumların başlangıcını ve "multi-line" yorumların başlangıcını, “Javadoc” yorumların başlangıcını belirlemek için bazı bayraklar oluşturur.

Aynı zamanda insideNestedComment adlı bir flag daha oluşturulur. “//* bu bir iç içe geçmiş yorum satırıdır.*/” Bu gibi yorum satırları için daha farklı bir kontrol bloğu oluşturuldu.

Yorumların bulunduğu fonksiyon kod bloğunu döngüye sokar ve yorumlar bulunduğunda ilgili listelere ekler.

Özel olarak çok satırlı yorumlarda “/**/” ifadesini tanımadığı için ona özel bir control bloğu daha oluşturuldu.

Son olarak her bir fonksiyonun yorum sayıları ekrana ve yorum satırları ilgili dosyalara yazıldı.

2. ÇIKTILAR

Fonksiyon:

```
/**
 * Varsayılan kurucu fonksiyon
 */
Motor(/**/ // /**/
) {
   ///*
   //*
   //
   //* */
   this.motorNo = UUID.randomUUID(
         //
   ).toString(/**/);
   /* Başlangıçta false */calisiyor = false;
}
```

Çıktı:

```
Function: Motor
Number of single line comments: 6
Number of multi-line comments: 3
Number of Javadoc comments: 1
```

3. SONUÇ

Çalışmamdan sonra Düzenli İfadelerin kullanım alanlarını öğrendim. Örneğin, bir metin dizesindeki e-posta adreslerini veya telefon numaralarını bulmak, belirli bir kelime veya karakter öbeğini içeren tüm satırları bulmak veya bir URL'yi parçalamak gibi işlemler için Regex kullanılabilir.

Referanslar

[1] https://thecodeprogram.com/java-ile-regex-kullanimi.

[2] https://regex101.com/

[3] https://www.javatpoint.com/java-comments.