**1fd75c3872a94e26ad68c7fa7667bdc82c07dd4ba85f4a6793f7a2b4e943b8fd**

**T.C.**

**SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

**1.ÖDEV**

JAVA DOSYASININ YORUM SATIRLARINI TESPİT ETME

**G191210044 - Zeynep Betül KALKANLI**

**SAKARYA**

**Nisan, 2023**

Programlama Dillerinin Prensipleri Dersi

JAVA DOSYASININ YORUM SATIRLARINI TESPİT ETME

Zeynep Betül KALKANLI

G191210044 – 2.Öğretim A Grubu

Özet

Ödevde dışarıdan verilecek olan .java uzantılı dosyanın fonksiyonlarını ve yorum satırlarını tespit etmemiz istenmiştir. Regex ifadeleri kullanarak algoritma tasarladım. Öncelikle .java dosyasında birden fazla sınıf olma ihtimaline karşılık regex ifadesiyle sınıflar tespit ediliyor. Kaç tane oldukları ve hangi satırlarda bulundukları değişkenlere kaydediliyor. Onun ardından teker teker her sınıfın fonksiyonları aynı şekilde tespit ediliyor. Bunların da içlerine girerek tek satırlı, çok satırlı ve javadoc yorumları belirli regex ifadeleriyle bulunuyor. Javadoc yorumları için aynı zamanda fonksiyonun üstüne de bakılıyor ve o fonksiyona dahil ediliyor. En son olarak bu yorumların kaç tane oldukları konsola, yorumların içerikleri de .txt uzantılı dosyalara kaydediliyor. txt uzantılı dosyalara kayıt yapılırken o yorumun hangi sınıfa ve fonksiyona ait olduğu da belirtiliyor.

© 2017 Sakarya Üniversitesi.

Bu rapor benim özgün çalışmamdır. Faydalanmış olduğum kaynakları içeresinde belirttim. Her hangi bir kopya işleminde sorumluluk bana aittir.

Anahtar Kelimeler: java, regex, javadoc

1. GELİŞTİRİLEN YAZILIM

Verilen ödevde bizden java uzantılı dosyaların fonksiyonlarını bulmamız ve her birinin yorum satırlarını text dosyasına yazmamız isteniyordu.

Ödevim yapısı gereği java uzantılı dosyayı, ödev klasörümün içinde bulunan records isimli klasörde arıyor. Onun dışında bir yerde bulunan java uzantılı dosyanın okunması istenirse hata alınacaktır.

Hazırladığım ödevde 6 adet class bulunuyor: algoritmanın çalıştığı ve ana sürecin işlediği Program.class’ı, okunan dosyadaki sınıf yapılarının bilgilerinin tutulduğu Sinif.class’ı, sınıflardaki fonksiyonların yapısı için oluşturduğum Fonksiyon.class’ı ve her bir yorum yapısı için ayrı ayrı oluşturduğum JavadocYorum, TekSatirliYorum, CokSatirliYorum class’ları.

Ödev için yazdığım algoritma sınıfları bularak başlıyor. Bir java dosyasının içerisinde birden fazla sınıf olabilme ihtimaline karşılık dosyayı taradıktan sonra içerisinde kaç sınıf var ve onların satırları nereden başlıyor diye regex ifadeleri kullanarak tespit yaptım. Sınıfların başlangıç ve bitiş satırlarını tespit ederek kodun tamamından kod parçacıkları çıkardım ve onları bir array list’te tuttum. Ardından benzer işlemi her sınıfta teker teker fonksiyonlar için yaptım.

Fonksiyon kod parçacıklarının içerisinde de yorumları aradım. Fakat javadoc yorumlarının fonksiyon kod parçacıklarının içerisinde olmama ihtimalinden dolayı javadoc arama algoritmasını daha farklı çalıştırdım. Bir javadoc için iki ihtimal tanımladım. Birincisi fonksiyonun hemen üstünde olabilir, ikincisi ise fonksiyonun içinde bulunabilir. Bundan dolayı fonksiyonun üstündeki javadoc’u bulmak için fonksiyon kod parçacığını değil sınıf kod parçacığını kullandım. Diğer yorumları yalnızca fonksiyonların içinde aradım. Yorumları tespit ederken kaç satır olduklarını, hangi satırda başladıklarını değişkenlerde tuttum ve her yapı için yaptığım gibi yorum yapıları için de, hangi yorum tipiyse, bir nesne oluşturup içeriklerini, bulundukları satır aralıklarını ve satır sayılarını o nesnenin niteliklerine ayarladım. Yaptığım tüm bu arama işlemlerinin ardından text dosyasına yazdırdım. Text dosyalarının records isimli klasöre oluşturulması için kod yazdım. Ödevde istendiği gibi her yorum tipi ayrı ayrı dosyalarda tutuluyor.

Şekil 1’de algoritmamın kodları parçalama ve kod parçacıklarından yorumları arama kısmı gösteriliyor. Tablo 1’de ise kullandığım regex ifadeleri, kullanım amaçları ve kullanırken arama yapılan kod parçacıkları yazıyor.



Şekil 1 Kod parçalarına bölme ve arama algoritmasının şekilsel gösterimi

Tablo 1. Kullanılan regex’ler

|  |  |  |
| --- | --- | --- |
| **Kullanım amacı** | **Regex ifadesi** | **Aranan metin** |
| Kaç tane sınıf var | class [a-zA-Z]+ | Tüm dosya |
| Sınıfın adı | [^(public)|(private)|(protected) class][a-zA-Z]+ | Tüm dosya |
| Fonksiyon ya da sınıf bloğu başlangıcı | \{ | Sınıfın ya da fonksiyonun kod parçası |
| Fonksiyon ya da sınıf bloğu bitişi | } | Sınıfın ya da fonksiyonun kod parçası |
| Fonksiyon adı | (?m)(?<=\bpublic ).\*$|(?m)(?<=\bprivate ).\*$|(?m)(?<=\bprotected ).\*$ | Sınıfın kod parçası |
| Fonksiyon adını tüm karakterlerden ayırıyor ve yalnızca ismini bırakıyoruz | .+?(?=\() | Sınıfın kod parçası |
| Javadoc bitiş | \\*\/ | Fonksiyonun üstünde Javadoc var mı diye -> sınıfın kod parçası    Fonksiyonun içinde Javadoc var mı diye -> fonksiyonun kod parçası |
| Javadoc başlangıç | \/\\*\\* | Sınıfın kod parçası  ya da  fonksiyonun kod parçası |
| Tek satırlı yorum başlangıcı | \/\/ | Fonksiyonun kod parçası |
| Tek satırlı yorum içeriği | \/\/[\\sA-Za-z\_ğüşıöçĞÜŞİÖÇ]+\/\/ | Fonksiyonun kod parçası |
| Çok satırlı yorum başlangıcı | \/\\*$|\/\\*[\sA-Za-z\_ğüşıöçĞÜŞİÖÇ0-9]+ | Fonksiyonun kod parçası |
| Çok satırlı yorum bitişi | \\*\/ | Fonksiyonun kod parçası |

* 1. Ödev Sırasında Karşılaşılan Zorluklar

Ödev sırasında bazı kısımlarda çok fazla vakit harcadığımı söyleyebilirim. Ödevi ilk olarak daha farklı biçimde yapıyordum. En başta belirttiğim 6 sınıfı oluşturmamıştım. Sadece Program sınıfım bulunuyordu ve yapıları düzgün bir şekilde kayıtlı tutmamak işimi daha da zorlaştırmıştı. Ödevi bu şekilde yaparken regex yapısıyla nasıl çözülebileceğini oldukça anlamıştım. İşimin çok karmaşık ve dağınık bir hâl aldığını görünce sıfırdan yeni bir projeye başladım. Bu sefer başta belirttiğim bu sınıfları oluşturdum.

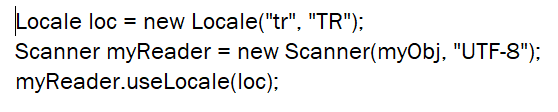
Bir başka zorlandığım alan ise yorumların farklı tiplerde olma ihtimaliydi. Örneğin çok satırlı gibi başlayan bir yorum tek satırda da bulunabiliyordu: “ /\* abc \*/ “ şeklinde. Bu ve benzeri ihtimallerin tamamı düşünülmeliydi. Çünkü verilecek her türlü java dosyasının çalışması gerekirdi.

Programı yazma aşaması tamamlandıktan sonra jar dosyası oluşturma sırasında bazı problemlerle karşılaştım. Jar dosyasını oluşturabiliyor fakat çalıştıramıyordum. Bunu da uzun araştırmalar sonucunda çözdüm ve java’mın sürümünün eski olduğunu öğrendim.

Farklı java dosyalarıyla denemeler yaparken bazı hatalarla karşılaştım. Bunlardan birisi türkçe karakteri algılamamaydı. Eğer java dosyasının içinde türkçe karakterli bir yorum bulunuyorsa o yorumun tükçe karakterine kadar olan kısım txt dosyasına yazdırılıyor, geri kalanı yazdırılmıyordu. Bunu da aşağıdaki kodları yazarak çözdüm.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

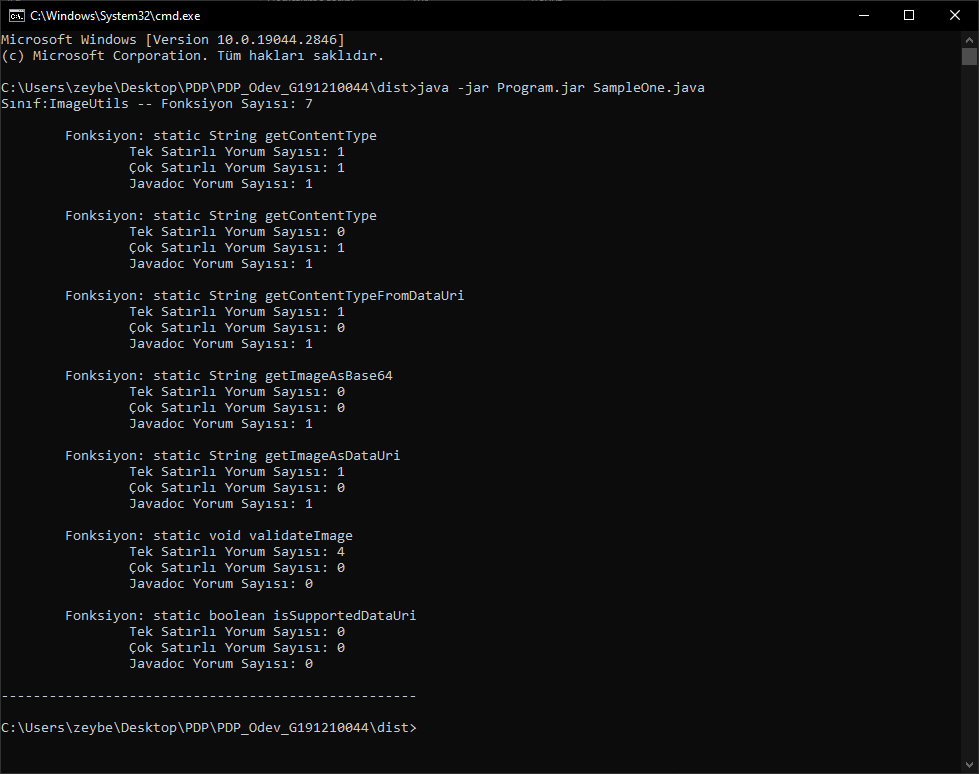


Şekil 2 Türkçe karakter okuma sorununu çözen kod parçası

1. ÇIKTILAR

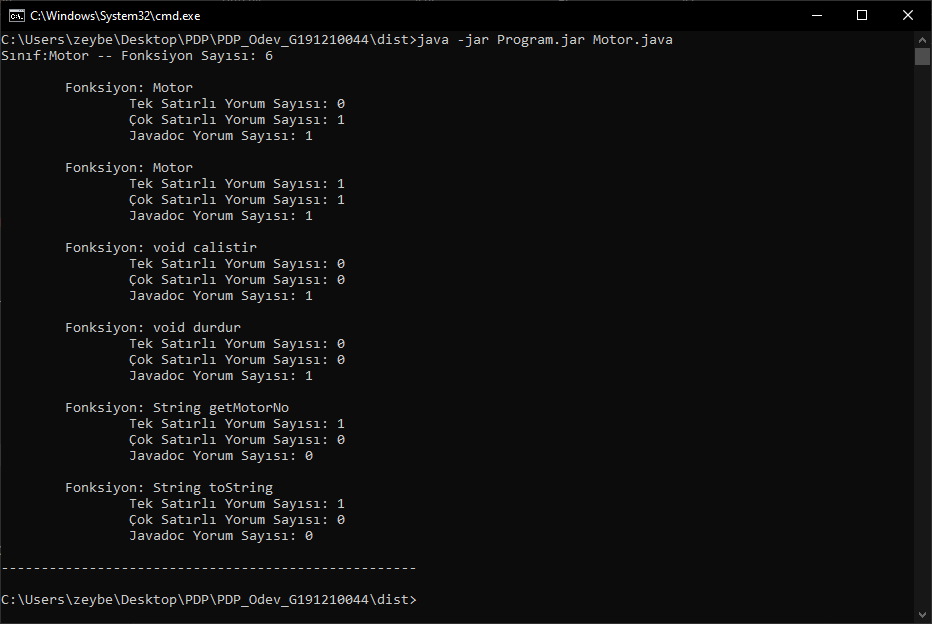
Hazırlamış olduğum ödevi sürekli farklı girdilerle denedim. İki tanesinin örneğini vereceğim.

İlk olarak Şekil 3’teki çıktı github’dan bulduğum bir java projesinin[1] verilmesi sonucu alınan çıktıdır. Konsola bu şekilde yazdığı gibi yorumların içeriğini de teker teker txt dosyalarına yazmıştır.



Şekil 3 Birinci Örneğin Çıktısı

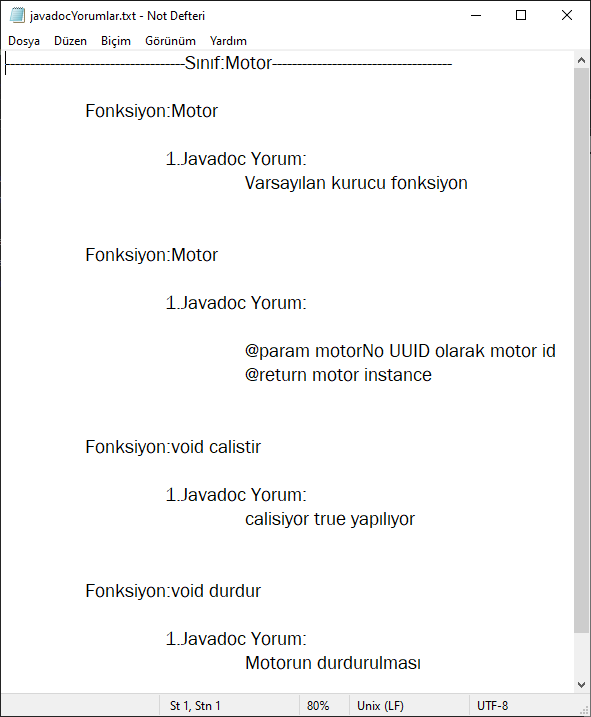
İkinci olarak da ödev dökümanında bulunan java kodunun programa verilmesiyle alınan çıktı aşağıdadır.



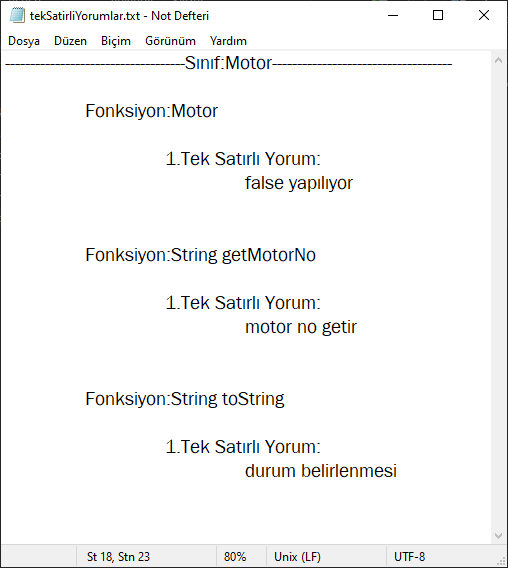
Şekil 4 İkinci Örneğin Çıktısı

metin içeren bir resim

Açıklama otomatik olarak oluşturulduYukarıdaki görseller ise ödev dökümanında verilen java dosyasının içerisinde bulunan tespit edilmiş Javadoc, çok satırlı ve tek satırlı yorumların ayrı ayrı kayıtlarının yapıldığı txt dosyalarıdır.



Şekil 5.3. Javadoc yorumların kaydedildiği text dosyası



Şekil 6.2. Tek satırlı yorumların kaydedildiği text dosyası

Şekil 7.1. Çok satırlı yorumların kaydedildiği text dosyası

1. SONUÇ

Çalışmamızın gerçek hayatta kullanım biçimi arama yapma, derleme gibi alanlar olabilir. Bu tarz projeler için küçük bir egzersiz yapmış olduğumu düşünüyorum.

Referanslar

1. Github.com “Smile”. Erişim Tarihi: 20.04.2023. <https://github.com/haifengl/smile/tree/master/base/src/main/java/smile/data>
2. stackoverflow.com. “[Read utf-8 using Scanner](https://stackoverflow.com/questions/27473513/read-utf-8-using-scanner)”. Erişim Tarihi: 20.04.2023. <https://stackoverflow.com/questions/27473513/read-utf-8-using-scanner>
3. stackoverflow.com. “[Java: Check if command line arguments are null](https://stackoverflow.com/questions/3868878/java-check-if-command-line-arguments-are-null)”. Erişim Tarihi: 20.04.2023. <https://stackoverflow.com/questions/3868878/java-check-if-command-line-arguments-are-null>
4. ahmetyasincivan.com. “Java Dosya Klasör Yolu Öğrenme”. Erişim Tarihi: 20.04.2023. <http://ahmetyasincivan.com/java-dosya-klasor-yolu-ogrenme/>
5. Oracle. “Class Pattern”. Erişim Tarihi: 12.04.2023. <https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/regex/Pattern.html>
6. Stackoverflow.com. “[Using Java to find substring of a bigger string using Regular Expression](https://stackoverflow.com/questions/600733/using-java-to-find-substring-of-a-bigger-string-using-regular-expression)”. Erişim Tarihi: 12.04.2023. <https://stackoverflow.com/questions/600733/using-java-to-find-substring-of-a-bigger-string-using-regular-expression>
7. coderspace.com. “Regex”. Erişim Tarihi: 10.04.2023. <https://coderspace.io/sozluk/regex>