

T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

ÖDEV 1

B201210024 - Ahmet Furkan Söğütcü

SAKARYA

Nisan, 2022

Programlama Dillerinin Prensipleri Dersi

PDP 1.ÖDEV

Ahmet Furkan Söğütcü

B201210024 - 1/A

Özet

Bu dersin ödevinde hocamız bizden kendisinin parametre olarak atacağı .java uzantılı dosyasının kaç adet istediği operatöre ve operanda sahip olduğunu bulmamızı istiyordu. Ben bu ödevde genel olarak programcıların kodlarında sadeliği, hızı vb. konuları ön planda tuttuğundan dolayı kod fazlalıklarına yer vermedim ve Regex (Regular Expression) yöntemini kullandım. Regex yöntemini kullanmam her ne kadar bana sadeliği ve hızı sunsa da öğrenmek ve istenilen operatörleri tespit etmek için bir hayli zaman harcadım. Ödevde yorum satırlarındaki operatörlerin alınmaması için de ayrı bir regex gerekiyordu, onu java'da bulunan replaceAll fonksiyonu ile kullanıp yorum satırı içerisindeki ifadelere boş string atadım. Dosya okuma işlemini ise concat adlı bir metot bularak onunla satır satır okuduğum dosyamı bütün olarak bir string'e atadım ve okuma işlemlerimi o şekilde yaptım. Lexical.java dosyamda ise verilerin dış ortamdan etkilenmemesi için değişkenleri private eleman olarak tanımlayıp get-set metotlarını kullandım.

©2022 Sakarya Üniversitesi.

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

Anahtar Kelimeler: Regex, Operatörler, Concat, java

1. GELİŞTİRİLEN YAZILIM

1.1 Dosya İşlemleri

Ödevi yapmaya önce dosya okuma işlemini hallederek başladım. Programda herhangi bir hata olup dosyayı okumaması durumunu göz önüne alarak try catch yapısını kullandım(try bloğunda hata oluşursa catch bloğuna gidip hatayı yazdıracak). Dosyayı okutmak için ise önce scanner ile gelecek dosyayı girdi olarak aldım, ardından metinleri sorunsuz bir şekilde okutmak için bufferedreader sınıfını kullandım. En son da filereader ile dosyamı belirttim. Bunları daha az kod kullanmak için aynı scanner parametresinde tek satırda oluşturdum (Program.java dosyasındaki 23. satır). Okunan dosyamın üzerinde regex'ler ile rahat işlem yapabilmek için deneme dosyasını satır satır okutup concat metodu ile bütün bir string'e dönüştürdüm ve böylelikle dosya işlemlerimi bitirmiş oldum.

1.2 Regex

Dosya işlemlerini hallettikten sonra dosya içerisindeki operatörleri bulmak ve saydırmak için yazılım dünyasında önemli bir yeri olan ve her dilde aynı ifadelere sahip regex'i araştırmaya ve öğrenmeye başladım(bkz. Referanslar [2],[3]). Ödevde bulunması gereken tekli, ikili, ilişkisel ve mantıksal operatörlerin ayrı ayrı regexlerini yaptım. Hocamızın ödev dosyasındaki deneme dosyasından aldığı çıktılara göre de operand ve sayısal operatörler için formüller kullandım: Sayısal operatörler için "tekli + ikili = sayısal" ifadesini kullandım. Operand için ise tekli operatörler hariç her operatör iki operand arasında gerçekleşeceği için "tekli + ikili*2 + ilişkisel*2 + mantıksal*2 = operand" formülünü kullandım.

Operatörler	Regex
Tekli	([\+] [-]){2,3}
İkili	* \/ % \^ (? \+)\+(?!\+) (?<!-)-(?!-) (?<!&)&(?!&) (?<!\)\ (?!\) (?<!= < ! - \+ \ * \/ % \^ &)=(?!=) (? \.)*(?!\. ;)</td
İlişkisel	< > [? \! = <][\=?!\=]
Mantıksal	([\]{2}) ([&]{2}) (!(?=[^=]))

Tablo 1. Operatörlerin Regex'leri

Ödev Sorumlusu: Ahmet Furkan Söğütcü, B201210024

Mail Adresi: ahmet.sogutcu@ogr.sakarya.edu.tr

1.2.1 Tekli Operatör Regex'i

"+" ve "-" arka arkaya 2 veya 3 kereyse al.

1.2.2 İkili Operatör Regex'i

Tekli operatör "+" ve "-" 'lerini ayırmak için sağında ve solunda "+" olan "+" ifadesi ile sağında ve solunda "-" olan "-" ifadesini alma. Mantıksal operatör "&" ve "|" 'lerini ayırmak için sağında ve solunda "&" olan "&" ifadesi ile sağında ve solunda "|" olan "|" ifadesini alma. İlişkisel operatör "==""i ayırmak için sağında ve solunda "=" olan "=" ifadesini alma. Geri kalan ikili operatörler için ise "=""in yanındaki "+", "-", "|", "*", "%", "%", "%", "&" ifadelerini tek başlarına al. İkili operatörlerde "=" ifadesini okutmayıp sadece yanlarındakini ayırt ederek saydırma yaptım.

1.2.3 İlişkisel Operatör Regex'i

```
"<" ve ">" 'ü al. Yanında "!", "=", "<", ">" olan "=" ifadelerini al.
```

1.2.4 Mantıksal Operatör Regex'i

İki adet "&" operatörünün yan yana olduğu ifadeleri al. İki adet "|" operatörünün yan yana olduğu ifadeleri al. "!" ifadesini al. Yanında "=" bulunan "!" ifadesini alma.

1.2.5 Yorum Satırları Regex'i

"//" ifadesinden sonraki ifadeleri bir alt satıra geçmeden al."/*" ve "*/" arasındaki değerleri al.

2.ÇIKTILAR

Hocamızın raporda örnek olarak verdiği deneme.java dosyasından farklı olarak github'da bulunan (bkz. Referanslar [1]) dosya üzerinde de denemelerimi yaptım. O githubdaki dosyada tekli, ikili ve sayısal operatörleri hocamızın istediği şekilden farklıydı ama operatörlerin tekli mi, ikili mi, ilişkisel mi ya da mantıksal mı olduğu doğru bir şekilde analiz ediliyordu. Yorum satırlarında ise silme ile operatör bulma ve saymasında bir sıkıntı yaşamadım. Sadece Arraylist'ler içindeki Arraylist'ler problem yaratıyordu. Onu da Arraylist bulma regex'im ile çözdüm. Arraylist'i bulan regex'i replaceAll fonksiyonu ile Arraylist'in yerine boşluk atıyorum öyle sayım gerçekleştiriyorum. Bir de kütüphanelerin belirtildiği kısımda istediğimiz kütüphanenin tüm özelliklerini kullanmak için ".*" ifadesini kullanıyorduk(import java.util.*;). Buradaki "*" operatörü de programda saydırılıyordu. Bu son günlerde aklıma geldi ve onu da regex ile saydırmayarak düzelttim.

3.SONUÇ

Çalışma sonucunda istenilen operatörleri, operandları ve sayılarını başarılı ve istenilen şekilde kullanıcıya aktarıyorum. Bunun sonucunda kullanıcı, dosyası içinde kullanılan operatörler hakkında kolayca bilgi sahibi olabiliyor ve operatörlerinin sayısı ve nerede olduğu hakkındaki bilgileri karşısında görüyor.

Referanslar

- [1] https://gist.github.com/AN-773/316fc14007325d2c11b8f8fa21124154
- [2] https://regex101.com
- [3] https://www.rexegg.com/regex-quickstart.html
- [4] https://regexlearn.com
- [5] https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html
- [6] https://www.youtube.com/watch?v=CzcWILy4WxM&list=PLd3UqWTnYXOkvuQb1D4wz2BY0XnKRpEiU&index=3
- [7] https://www.youtube.com/watch?v=lHFlAYaNfdo&t=339s
- [8] https://blog.ostermiller.org/finding-comments-in-source-code-using-regular-expressions/