T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

1.ÖDEV

G211210047 – Mert Eser Meral

GRUP: 1-B

SAKARYA

Nisan, 2024

Programlama Dillerinin Prensipleri Dersi

GitHub Depo Analiz Programı

Özet

Hazırlamış olduğum program, kullanıcıdan bir GitHub deposunun bağlantısını alarak, bu depoyu klonlar ve içindeki Java sınıflarını analiz eder. Analiz, her bir sınıf için Javadoc satırı sayısı, yorum satırı sayısı, kod satırı sayısı, LOC (Line of Code – Bir dosyadaki her şey dahil satır sayısı) ve sınıftaki fonksiyon sayısını hesaplar. Ayrıca, yorum sapma yüzdesini de hesaplayarak sonuçları raporlar.

Proje Yapısı

Programın yapısı, “Program.java” tarafından başlatılmakta olup, kullanıcıdan girdi almak için “Input.java”, depo klonlamak için “Depo.java”, Java dosyalarını analiz etmek için “Analiz.java” sınıflarını kullandım**.**

- **Program.java:**
  Bu sınıf, projenin başlangıç noktasıdır.Önce “Input.getInput()” metodunu çağırıp Kullanıcıdan GitHub depo linkini aldım. Sonra “Depo.cloneRepository()” metoduyla depoyu klonladım ve ardından “Analiz.analyzeClasses()” metodunu çağırarak klonlanan depodaki sınıfları analiz ettim.
- **Input.java:**
  Burada “getInput()” metodu ekrana kullanıcı girişi için bir uyarı yazısı çıkartır ve kullanıcıdan depo linkini alır. Ve bu linki döndürür.
- **Depo.java:**
  Burada “cloneReposity()” metodunu Github deposunu klonlamak için kullandım. Metodda Git'in sağladığı “git clone” komutunu kullandım. İşlem, bir ProcessBuilder nesnesi kullanılarak başlatılır ve çalışma dizini “user.dir” ile mevcut çalışma dizini olarak ayarlanır. Klonlama işlemi tamamlandıktan sonra, işlemin çıkış kodu kontrol edilir. Başarılı bir klonlama işlemi durumunda kullanıcıya "Depo başarıyla klonlandı." mesajı gösterilirken, olası hatalar durumunda bir hata mesajı iletilir.
- **Analiz.java:**
  İlk başta “analyzeClasses()” metoduyla depodaki dosya ve klasörleri inceledim. Burda if yapısıyla sadece “.java” uzantılı dosyaları ve bu dosyalardan da sadece sınıf olanları ayıkladım ve bunları “analyzeJavaFile()” metoduyla analiz ettim. Sınıf olanları ayıklamak için “classMi()” metodunu çağırdım.

“analyzeJavaFile()” metodunda ilk adımda verilen dosyayı satır satır okumak için bir while döngüsü başlattım. Ardından her satırın başında veya içeriğinde javadoc, yorum veya kod olup olmadığını string metodları yardımıyla kontrol ettim. Burada if else yapıları kullandım ve olabilecek her durumu ele alabilmek için iç içe kontroller de yaptım. Sağlanan koşullarda gerekli sayaçları arttırdım. Fonksiyon sayısını sayarken regex deseni kullandım. Bu deseni yaparken çok zorlandım çünkü olabilecek tüm fonksiyon başlıklarıyla eşleşebilecek regex desenini yakalamam gerekiyordu. Ardından pattern ve matcher kullanarak eşleşmeleri yakaladım. Eşleşme durumunda fonksiyon sayacını arttırdım. Son kısımda hocamızın verdiği formüllerle yorum sapma yüzdesini de hesaplayıp hocamızın istediği şekilde ekran çıktısı verdim.

Yorumlar ve İyileştirmeler

- Kodun okunabilirliğini artırmak için, değişken ve metod isimlerinin açıklayıcı olmasına dikkat ettim.
- Kod tekrarını azaltmak için, tekrar eden işlemleri metodlar içinde yeniden kullandım.
- Hata durumlarında kullanıcıya bilgilendirme sağlamak için hata mesajları ekledim.
- Kullanıcı girdileri doğru bir şekilde işleyerek, doğru çıktı vermeye çalıştım.

Sonuç

Yaptığım proje, kullanıcıdan alınan GitHub depo linkini klonlayarak içindeki Java dosyalarını analiz eden bir Java programını içermektedir. Projenin amacı, Java dosyalarının içeriğini inceleyerek çeşitli metrikleri hesaplamak ve bu sayede yazılım kalitesini değerlendirmektir. Programın modüler yapısı sayesinde, farklı analizler veya özellikler kolayca eklenebilir veya mevcut özellikler geliştirilebilir.