T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

1. Ödev

Tuğra YAVUZ - B221210064

1-A

Ödevde İstenilenler: **Ödevde öncelikle bir github reposunun bilgisayarımıza klonlanmasını ve bu github deposundaki kodlardan .java uzantılı olanlardan class tanımına sahip olan kodların çeşitli analizlere tabii tutulması isteniyor.**
Bu analizler arasında kodda kaç javadoc yorum satırı olduğu, kaç standart yorum satırı olduğu, toplam kaç kod satırı olduğu, her şey dahil kodun kaç satır olduğu, fonksiyon sayısı ve verilen denkleme bağlı olarak yorum sapma yüzdesi bulunmaktadır. Bu analizleri istenen şekilde ekrana yazdıktan sonra ödev tamamlanmış oluyor.

Öğrendiklerim: **Ödevi yaparken öncelikle java kodu ile git komutlarının nasıl kullanılanileceğini öğrenmiş oldum. Bu sayede ödevde söylendiği gibi repoya eriştim ve ilgili analizleri yapabildim. Regex konusunu öğrendim, regex gerçekten çok güçlü bir ifade. Bunun dışında string analiz metotlarını da öğrenmiş oldum bu sayede çeşitli if else blokları ile güçlü ve istenen şekilde analiz blokları kurdum. Javada kod yazma pratiğim bu ödev sayesinde epey artmış oldu.**

Ödevde Yaptıklarım: **Ödevde öncelikle bilgileri kolayca depolayabilmek için bir class tanımı ile başladım. Bu class içinde ekrana yazılması istenen değişkenler ve yorum sapma yüzdesi için çeşitli hesaplamalar ıyapacak metotları ekledim. Son olarak toString metodunu override ederek yani ezerek kolayca nesneleri ekrana yazdıranilecek hale getirdim. Main metoduma gelecek olursak burada repoyu çekmek ve .java içerip, class tanımı içerenleri çekmek için gerekli kodları yazdım. Ardından değişkenleri class nesnelerine atamak için nesneleri tanımladım ve analiz fonksiyonlarını saymaya başladım. İlgili if else blokları ve regex ifadeler ile analizlerimi başarılı şekilde gerçekleştirdim. javadocSatirSayisiniHesapla isimli fonksiyon ile en üstteki ve en alttaki /** ve */ ifadelerini dahil etmeden aradaki javadoc satır sayılarını ilgili if else bloklar ıile saydım bunun yanında eğer ki /** veya */ bulnuan satırlara da not alınmış mı diye kontrol etmek içim javadocSayirSayisiHesaplaAyniSatir isimli fonksiyonu yazdım. Bu fonksiypn da aynı şekilde ilgili eif else blolarına yapılar eklenerek ilk fonksiyondan türemiş bir fonksiyondur. Yorum satır sayısını bulmak için kullandığım yorumSatirSayici fonksiyonu da benzer mantıkta çalışmaktadır yine /** ve */ ifadeleri sayılmıyor ama bu satırlarda kod var ise benzer şekilde if else yapıları ile kontrol edilerek işleniyor. burada ayrıca // ifadesinin yanında bir yazı yazılmış mı yazılmamış mı bunun da kontrolü yapılmaktadır. LOC yani line of code için ilgili split ifadesi ile işlem yapılarak hesaplanmaktadır. boş satır sayısını hesaplamak için ilgili regex ifade ile \n ve \r ifadeleri eşleştirilerek sayılmaktadır. Kod satır sayısı kodun en önemli bölümü diyebilirim. Burada yorum satırlarında kullandığım ayırt etme fonksiyonlarının temelini koruyarak farklı varyasyonlarını oluşturdum ve bir küme problemi çözdüm. çeşitli durumlar çıkarken sayılmasını istediğim durumların da çıktığı oluyordu, durumun bir küme problemine dönüşmesinin sebebi buydu. Modifiye ettiğim**

modüller ile bu problemi de çözdüm. Fonksiyon sayıs hesaplama kısmını yine bir regex ifade ile gerçekleştirdim ve kurucu fonksiyonları da dahil etmeyi unutmadım.

Zorlandığım Yerler: **regex ifadeleri öğrenirken güçlük çeksem de çeşitli pratikler ile işin içinden çıkabildim. javadoc ve yorum satır sayısı kısmını da regex ifade ile yapmaya çalışşsam da bir türlü istenen şekilde yapmayı başaramadım bu yüzden string analiz yöntemlerine başvurarak satır satır kodları analiz etme yöntemine gittim ve bu işlemleri bu şekilde halletmi oldum. Küme problemü çözdüm dediğim kod satır sayısı bölümünde işin küme problemine dönüştüğünü fark edene kadar işin içinden çıkamamıştım ancak kağıt kalem alıp venn şemalarını çizdiğimde kolayca işin içinden çıkabildim. Bunun dışında java ile git komutlarını kullanmaya çalışmak epey zordu çünkü çeşitli sitelerden yaptığım araştırmalarda desteklenmeyen yöntemler anlatılıyotdu. Uzun araştırmalarım sonucunda istediğim şekilde ve desteklenen şekilde bu işin nasıl yapıldığını öğrendim ve koduma uyarladım. Bunun dışında istisnai durumların analizis işimi epey zorlaştırdı ve elimden geldiğince bu durumları göz önüne aldım. Verilen örnek repo ile çıktım aynı olsa da ekstra istisnai durumları da ele almayı unutmadım.**

Daha iyi olabilecek kısımlar: **kodda temel javadoc ve yorum satırı sayma fonksiyonlarındn türettiğim fonksiyonların temelini kodumdan silmedim bu yüzden performans düşüklüğü yaşanabilir ancak bunu nasıl türettiğimi kdou inceleyen insanlar kolayca anlayabilsin diye yaptım. Bunun dışında isteğe bağlı olarak koddaki dizin temizleme metodu aktif edilerek bellek temizliği yapılabilir. Bu durumlar dışında her şeyin stabil olduğunu düşünüyorum.**