

JAVA MULTI THREAD GRUP PROJESİ

Hazırlayanlar

Cihan Dilsiz - Geliştirme

Muhammed Çavuş - Geliştirme

Furkan Boztepe - Arge

Adem Hilmi Bozkurt - Arge

Projenin Amacı	2
İş Bölümü	2
Kullanılan Kütüphaneler	2
Proje Nasıl Çalışıyor?	2
Program Sınıf Diyagramları	3
Program Akış Diyagramları	4
Örnek Program Çıktısı	7
Sonuç	8
Git Akışı ve Sürüm Kontrol Stratejisi	8
Branch Örnekleri:	9
2. Commit & PR Süreci	9
3. Geliştirilen Özellikler (Feature Branch'leri)	10
4. Hata Yönetimi (Fix Branch'leri)	10
5. Test ve Refactor Süreci	10
6. Dokümantasyon	10
7. İnceleme & Onay Süreci (Review)	11
8. Sonuç	11
Git Conflict Çözüm Süreci (Senaryo: feat:create-zip-files)	13
Görsel 1 — Conflict Bilgisi (Genel Durum)	13
Görsel 2 — Conflictli Dosya: .idea/misc.xml	14
Görsel 3 — Conflictli Dosya: FileStats.java	15
Görsel 4 — Merge Sonrası Güncel Dosya Durumu	15
Görsel 5 — GitHub Üzerinden PR Detayı ve Merge	16
Görsel 6 — Merge Başarılı ve Branch Kapatıldı	17

Projenin Amacı

Bu projenin temel amacı, çok iş parçacıklı (multithreaded) bir yapı kullanarak **.txt** uzantılı metin dosyalarının eş zamanlı olarak analiz edilmesini, analiz sonuçlarının kullanıcıya sunulmasını ve ardından bu dosyaların arşivlenip sistemden silinmesini sağlamaktır. Proje sayesinde, farklı dosyalar paralel iş parçacıklarıyla hızlı ve verimli bir şekilde işlenmekte; her bir dosyanın satır ve karakter sayısı gibi istatistiksel verileri hesaplanmakta ve işlem sonrasında tüm dosyalar **.zip** formatında sıkıştırılarak bir arşiv dosyasında toplanmaktadır. Son aşamada, kaynak dosyalar sistemden silinerek temiz bir çalışma alanı bırakılmaktadır.

Bu yapı, özellikle büyük hacimli dosya analiz işlemlerinde performans ve kaynak yönetimi açısından avantaj sağlamakta ve Java'nın çoklu iş parçacığı kabiliyetlerinin gerçek bir senaryoda uygulanmasını hedeflemektedir.

İş Bölümü

Cihan Dilsiz - Main ve Timer sınıflarının kodlanması, ReadME, Git Operasyonları

Muhammed Çavuş - File analyzer sınıfının kodlanması, ...

Furkan Boztepe - ZipTask sınıfının kodlanması, Git conflict çözümü

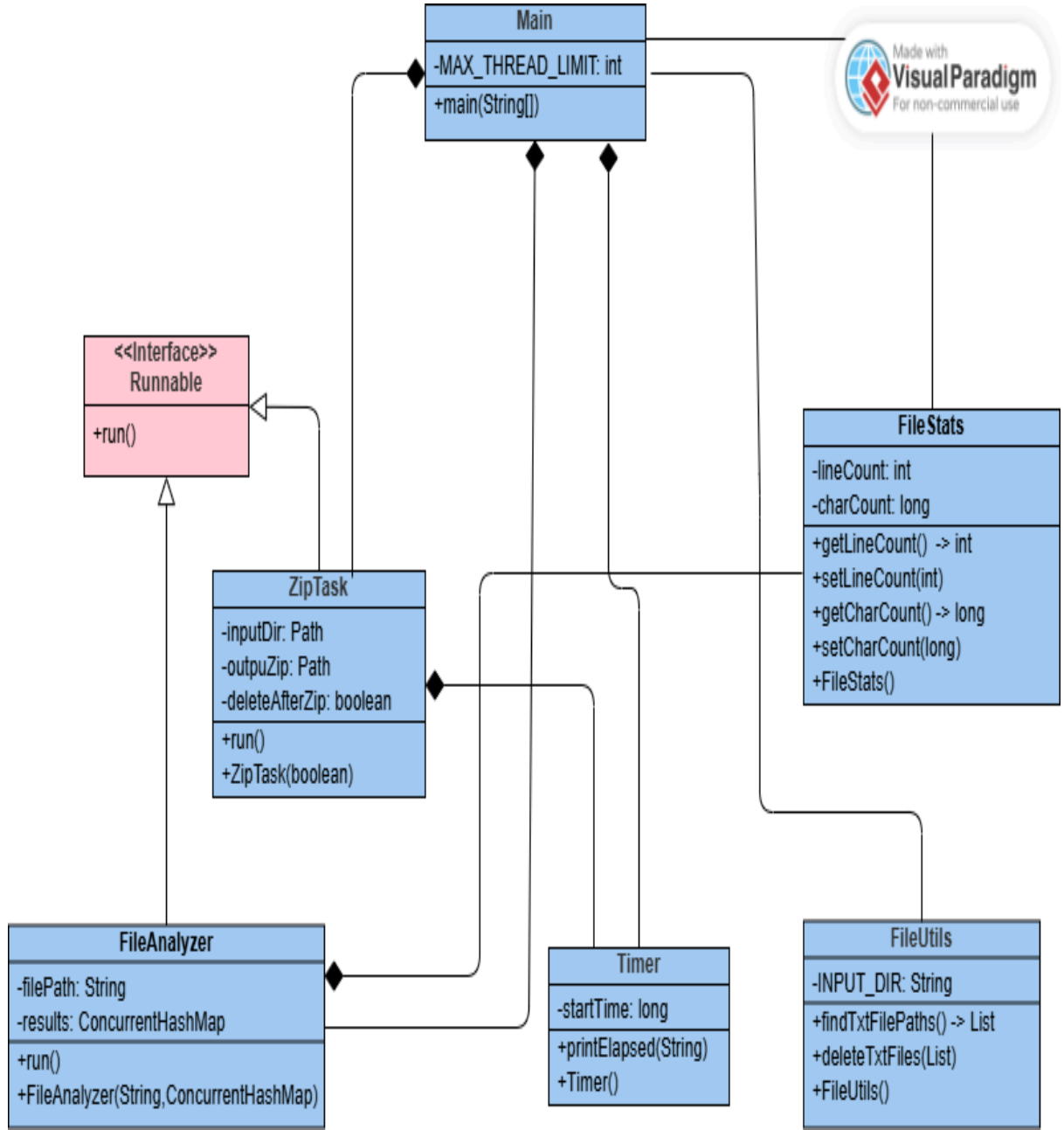
Adem Hilmi Bozkurt - FileUtils sınıfının kodlanması, Akış ve Sınıf diyagramları

Kullanılan Kütüphaneler

- io : input/output işlemleri -> BufferedReader, File, FileReader, IOException
- util.concurrent : eş zamanlı programlama -> ConcurrentHashMap, Timer, zip, ArrayList, List
- nio : yoğun G/Ç işlemleri için özellikler sunan bir Java programlama dili API'leri koleksiyonudur -> file

Proje Nasıl Çalışıyor?

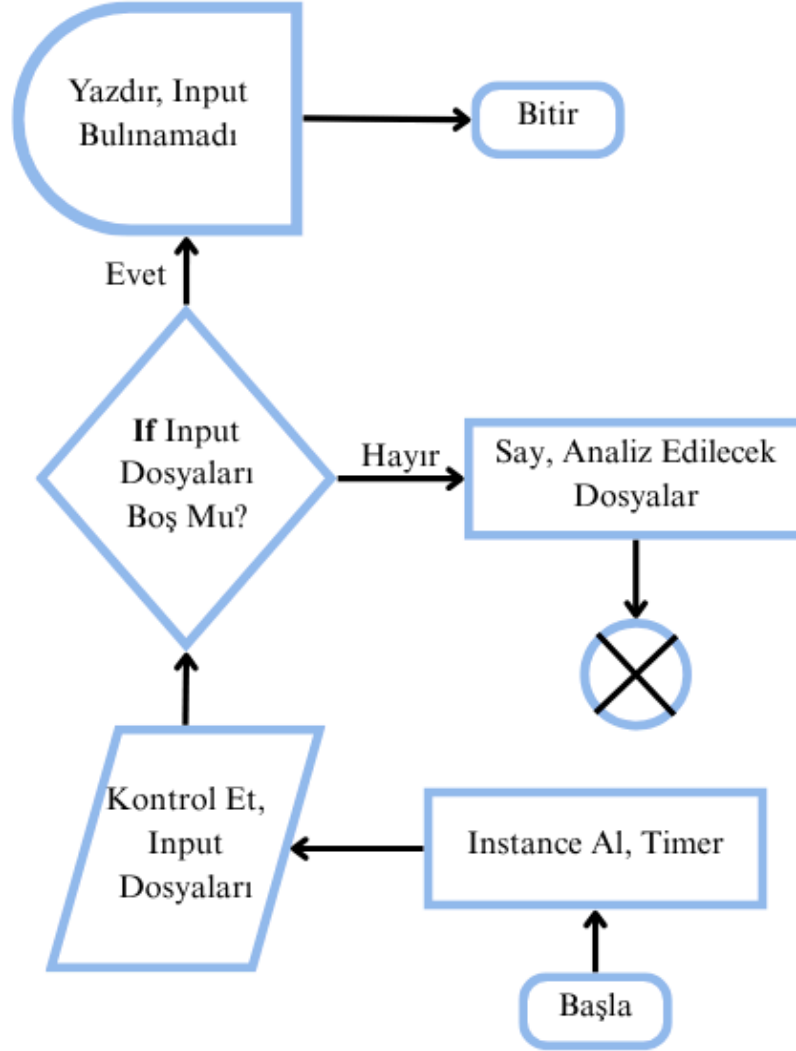
Program Sınıf Diyagramları



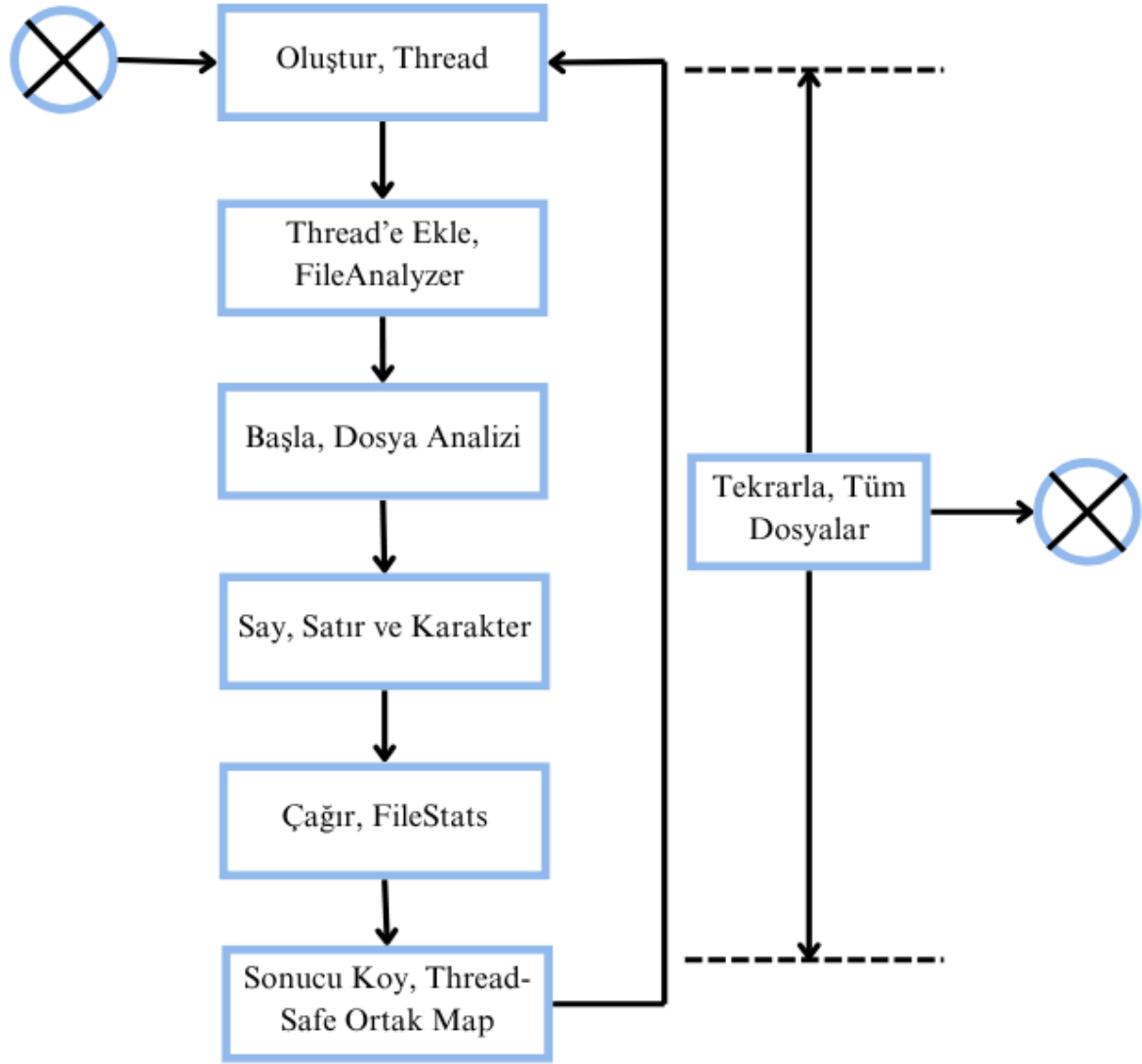
Şekil 1. Sınıf Diyagramlı

[Şekil1]: Görselde sınıf ilişkileri gösterilmiştir. Ana çağırıcı sınıf Main sınıfıdır. ZipTask ve FileAnalyzer sınıfı, Runnable interface’i implement etmişlerdir. FileUtils ve FileStats sınıfları ile Main sınıfı arasında bire bir ilişki vardır. Timer, ZipTask ve FileAnalyzer sınıfları ile Main arasında bire çok ilişkisi vardır. FileStats ile FileAnalyzer sınıfları arasında bire çok, Timer ile ZipTask sınıfları arasında bire çok ilişki vardır.

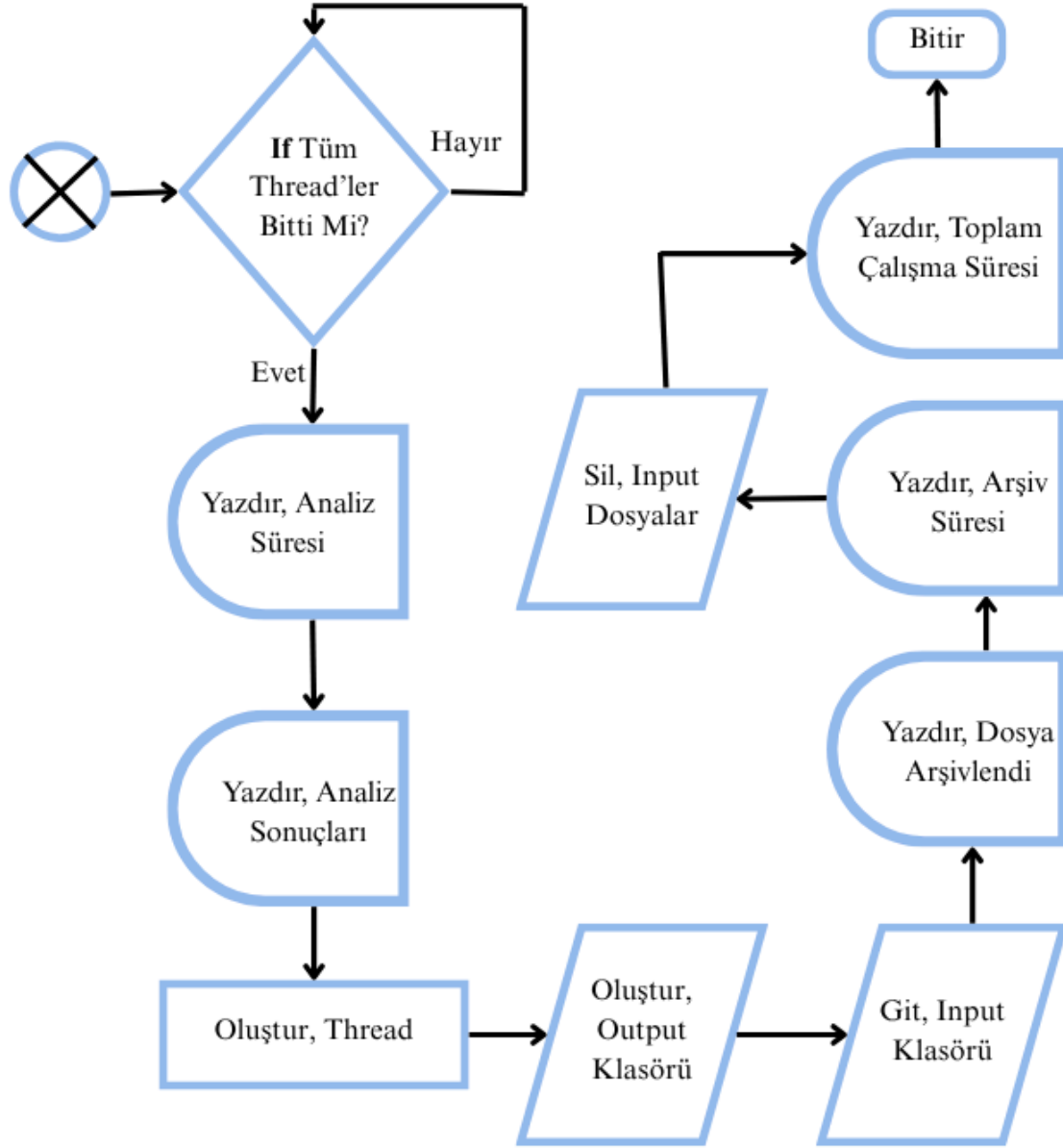
Program Akış Diyagramları



Şekil 2. Akış Diyagramı



Şekil 3. Akış Diyagramı Devam



Şekil 4. Akış Diyagramı Devam

Örnek Program Çıktısı

```
Users/cihandilsiz/Library/Java/JavaVirtualMachines/corretto-17.0.5/Contents/Home/bin/java
-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=63486
-Dfile.encoding=UTF-8 -classpath
/Users/cihandilsiz/IdeaProjects/multithread-file-analyzer/target/classes:/Users/cihandilsiz/.m2
/repository/commons-io/commons-io/2.15.1/commons-io-2.15.1.jar Main
==> Toplam analiz edilecek dosya sayısı: 4
Thread-0 -> 'test4.txt' dosyasını analiz etmeye başladı.
Thread-2 -> 'test3.txt' dosyasını analiz etmeye başladı.
Thread-3 -> 'test.txt' dosyasını analiz etmeye başladı.
Thread-1 -> 'test2.txt' dosyasını analiz etmeye başladı.
Thread-0 -> 'test4.txt' analizini tamamladı.
Thread-1 -> 'test2.txt' analizini tamamladı.
Thread-2 -> 'test3.txt' analizini tamamladı.
Thread-3 -> 'test.txt' analizini tamamladı.
Dosya analiz süresi süresi: 7414709 ns (7 ms, 0.007 s)
```

==== ANALİZ SONUÇLARI ====

```
test.txt - 17 satır / 2458 karakter
test2.txt - 12 satır / 2412 karakter
test3.txt - 4 satır / 234 karakter
test4.txt - 0 satır / 0 karakter
Toplam: 33 satır / 5104 karakter
```

```
Ziplendi: input/test4.txt
Ziplendi: input/test2.txt
Ziplendi: input/test3.txt
Ziplendi: input/test.txt
Dosyalar başarıyla ziplenmiştir: output/archived-files.zip
Zip işlemi süresi: 13558958 ns (13 ms, 0.014 s)
Zip oluşturma süresi süresi: 14517625 ns (14 ms, 0.015 s)
```

```
==> Analiz tamamlandı. .txt dosyaları siliniyor...
Silindi: /Users/cihandilsiz/IdeaProjects/multithread-file-analyzer/input/test4.txt
Silindi: /Users/cihandilsiz/IdeaProjects/multithread-file-analyzer/input/test2.txt
Silindi: /Users/cihandilsiz/IdeaProjects/multithread-file-analyzer/input/test3.txt
Silindi: /Users/cihandilsiz/IdeaProjects/multithread-file-analyzer/input/test.txt
==> Silme işlemi tamamlandı.
Toplam çalışma süresi süresi: 32179000 ns (32 ms, 0.032 s)
```

Sonuç

Projenin çıktısı incelendiğinde, geliştirdiğimiz çok iş parçacıklı (multithreaded) dosya analiz ve arşivleme sistemi, hedeflenen işlevselliğe başarıyla ulaşmıştır. Program, belirlenen bir klasörde yer alan **.txt** uzantılı dosyaları tespit ederek, her birini bağımsız iş parçacıkları (thread) ile eş zamanlı olarak analiz etmektedir. Örnek çalıştırma çıktısında görüldüğü üzere, dört farklı thread, dört farklı dosyayı paralel şekilde analiz etmeye başlamış ve her biri işlemini bağımsız şekilde tamamlamıştır. Bu durum, **thread'lerin** doğru şekilde oluşturulup yönetildiğini ve Java'nın çoklu iş parçacığı altyapısının efektif kullanıldığını göstermektedir.

Analiz işlemi sonucunda her dosya için satır ve karakter sayısı başarıyla hesaplanmış, bu veriler kullanıcıya detaylı şekilde sunulmuştur. Örnek olarak, **test.txt** dosyasında **17 satır ve 2458 karakter** tespit edilmiştir. Boş olan **test4.txt** dosyasının **0 satır ve 0 karakter** olarak analiz edilmesi de yazılımın kenar durumları (edge cases) için uygun şekilde tasarlandığını göstermektedir.

Analizin ardından sistem, tüm **.txt** dosyalarını **output/archived-files.zip** adlı bir arşiv dosyasında sıkıştırmıştır. Zip işlemi başarıyla gerçekleştirilmiş ve işlem süresi **nanosecond** cinsinden ölçülerek kullanıcıya sunulmuştur. Tüm bu işlemlerin ardından, analiz edilen orijinal **.txt** dosyaları sistemden silinmiş ve temizlik işlemi tamamlanmıştır. Bu, sistem kaynaklarının verimli kullanımı açısından oldukça önemli bir adımdır.

Toplam işlem süresi yaklaşık **32 milisaniye** gibi kısa bir sürede tamamlanmış olup, hem performans hem de işlevsellik açısından sistemin etkili bir şekilde çalıştığını göstermektedir. Proje genelinde thread'lerin eş zamanlı çalışması, analiz doğruluğu, sıkıştırma işleminin başarısı ve işlem sonrası temizlik gibi tüm kritik adımlar başarıyla yerine getirilmiştir. Bu da yazılımın sadece teorik değil, pratik olarak da amaca uygun şekilde geliştirildiğini göstermektedir.

Git Akışı ve Sürüm Kontrol Stratejisi

Projede özellik tabanlı (feature-based) Git akışı kullanılmıştır. Tüm geliştirme süreçleri aşağıdaki mantıkla ilerlemiştir:

- **main**: Üretim kodlarının bulunduğu ana dal

- **feat/***: Yeni özellik eklemek için açılan branch'ler
- **fix/***: Hata düzeltmeleri için oluşturulan branch'ler
- **docs/***: Dokümantasyon eklemek amacıyla kullanılan branch'ler

Branch Örnekleri:

- **feat/create-zip-files** – Dosyaları zipleyerek arşivleme
 - **feat/file-analyzer** – Dosya analiz işçisi ve istatistik sınıfları
 - **fix/delete-file-operation** – .txt dosyalarının silinmesi işlemi
 - **docs/add-readme** – README dökümanının oluşturulması
-

2. Commit & PR Süreci

Her geliştirici, yeni bir özellik eklemekten önce main'den kendi branch'ini oluşturmuştur. Aşağıdaki adımlar standart olarak izlenmiştir:

1. Branch oluşturma:
git checkout -b feat/<özellik>
 2. Kod geliştirme ve commit:
Anlamlı ve konvansiyonel commit mesajları ile sürüm kontrolü sağlandı.
Örn: feat(analyzer): Add FileAnalyzer worker and FileStats class
 3. Pull Request açma:
Özellik tamamlandıktan sonra PR oluşturuldu. Açıklamalarda yapılan geliştirme detaylandırıldı.
 4. Code Review süreci:
PR'lar en az bir grup üyesi tarafından incelendi.
Örn: Cihan Dilsiz reviewed and approved
 5. Merge işlemi:
Onaylanan PR'lar main dalına merge edildi. Gerekirse kapatılıp yeni branch üzerinden yeniden gönderildi (örn. PR #6 → closed, yerine #7 merge edildi).
-

3. Geliştirilen Özellikler (Feature Branch'leri)

PR	Açıklama	Katılımcı	Durum
#1	file analyzer ve zip sistemi kurulumu	cdilsiz5	✓ Merge
#3	Koordinatör thread ve zamanlayıcı	cdilsiz5	✓ Merge
#4	FileAnalyzer işçisi ve istatistik sınıfı	mcavus10	✓ Merge
#5	FileUtils işlemleri	ademhilmi	✓ Merge
#7	zip işlemi (revize)	furkanboztepe	✓ Merge

4. Hata Yönetimi (Fix Branch'leri)

PR	Açıklama	Durum
#8	FileUtils refactor ve negatif senaryo düzeltmeleri	✓ Merge
#10	.txt dosyalarının zip sonrası silinmesi	✓ Merge

5. Test ve Refactor Süreci

- Her özellik ayrı modüller halinde yazılarak test edilebilirlik artırıldı.
 - **FileAnalyzer**, **MainCoordinator**, **FileUtils** gibi sınıflar mock veya doğrudan test edildi.
 - Kod tekrarı önlendi, **refactor** ile sadeleştirme yapıldı.
-

6. Dokümantasyon

- **docs/add-readme** branch'i üzerinden README dosyası oluşturuldu.
 - İçerik:
 - Proje amacı ve açıklaması
 - Katılımcı listesi
 - Kullanım senaryosu
 - Proje klasör yapısı
-

7. İnceleme & Onay Süreci (Review)

Her PR şu şekilde yönetildi:

- Kodlar birbirimiz tarafından gözden geçirildi.
- Kod kalitesi, modülerlik ve işlevsellik kontrol edildi.
- Gerekli durumlarda yorum bırakılarak kod iyileştirmesi istendi.
- Onay sonrası **main** dalına merge edildi.

8. Sonuç

Bu projede ekip olarak aşağıdaki yetkinlikleri başarıyla uyguladık:

- Takım içi iş bölümü ve koordinasyon
- Git akışı, feature/fix branch kullanımı
Kod gözden geçirme (code review) disiplini
- Java multithreading konseptlerinin pratik uygulaması
- Test, dökümantasyon ve hata yönetimi

Merge remote-tracking branch 'origin/main' into feat/create-zip-files	4561a57	<>
feat:create-zip-files	d826c85	<>
Merge pull request #5 from cdilsiz5/feat/file-utils	Verified 5cee067	<>
feat:FileUtils Operations	106f5c3	<>
Merge pull request #4 from cdilsiz5/feat/file-analyzer	Verified 373ff96	<>
feat(analyzer): Add FileAnalyzer worker and FileStats class	1c4ed16	<>
Commits on Jul 16, 2025		
Merge pull request #3 from cdilsiz5/feat/main-task-manager	Verified 423c085	<>
feat: implement main coordinator and timing utility	e3d229d	<>
Commits on Jul 15, 2025		
Merge pull request #1 from cdilsiz5/feat/file-analyzer-set-up	Verified 78139d3	<>
feat: file analyzer and zip setup	fe98ebd	<>
Initial commit	66f167d	<>

Şekil 5. Git Commit History

Commits

main

All usersAll time

Commits on Jul 18, 2025

Merge pull request #10 from cdilsiz5/fix/delete-file-operationVerified0382a3c

cdilsiz5 authored 2 days ago

fix: delete input .txt files after processing and zipping72f4e39

cdilsiz5 committed 2 days ago

Merge pull request #9 from cdilsiz5/docs/add-readmeVerified6ce0742

cdilsiz5 authored 3 days ago

docs: add project README with usage, contributors, and structurec25f9ce

cdilsiz5 committed 3 days ago

Commits on Jul 17, 2025

Merge pull request #8 from cdilsiz5/fix/fileutils-and-main-testVerified38d17d4

cdilsiz5 authored 3 days ago

fix: negative case handling and FileUtils refactorb28283e

cdilsiz5 committed 3 days ago













Merge pull request #7 from cdilsiz5/feat/create-zip-filesVerifiedc70bb74

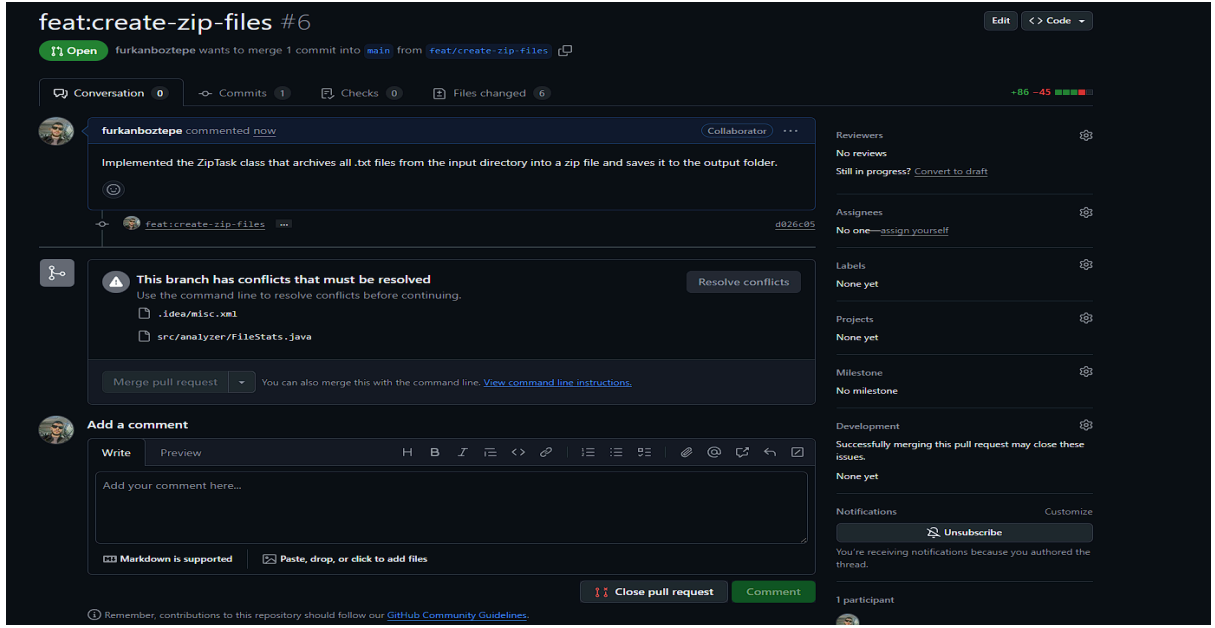
cdilsiz5 authored 3 days ago

Şekil 6. Git Commit History Devamı

Branch	Updated	Check status	Behind	Ahead	Pull request	
fix/delete-file-operation	2 days ago		3	0	#10	
docs/add-readme	3 days ago		3	0	#9	
fix/fileutils-and-main-test	3 days ago		5	0	#8	
feat/create-zip-files	3 days ago		7	0	#7	
feat/file-utils	3 days ago		12	0	#5	
feat/file-analyzer	3 days ago		12	0	#4	
feat/main-task-manager	4 days ago		14	0	#3	
feat/file-analyzer-set-up	5 days ago		16	0	#1	

Şekil 6. Git Branches

<input type="checkbox"/> 0 Open ✓ 10 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Reviews ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>  fix: delete input .txt files after processing and zipping #10 by cdilsiz5 was merged 2 days ago							
<input type="checkbox"/>  docs: add project README with usage, contributors, and structure #9 by cdilsiz5 was merged 3 days ago							
<input type="checkbox"/>  fix: negative case handling and FileUtils refactor #8 by cdilsiz5 was merged 3 days ago							
<input type="checkbox"/>  feat:create-zip-files #7 by furkanboztepe was merged 3 days ago							
<input type="checkbox"/>  feat:create-zip-files #6 by furkanboztepe was closed 3 days ago							
<input type="checkbox"/>  feat:FileUtils Operations #5 by cdilsiz5 was merged 3 days ago							
<input type="checkbox"/>  feat(analyzer): Add FileAnalyzer worker and FileStats class #4 by mcavus10 was merged 3 days ago							
<input type="checkbox"/>  feat: implement main coordinator and timing utility #3 by cdilsiz5 was merged 4 days ago							
<input type="checkbox"/>  feat:FileUtils Operations #2 by ademhilmibozkurt was closed 4 days ago							
<input type="checkbox"/>  feat: file analyzer and zip setup #1 by cdilsiz5 was merged 5 days ago							

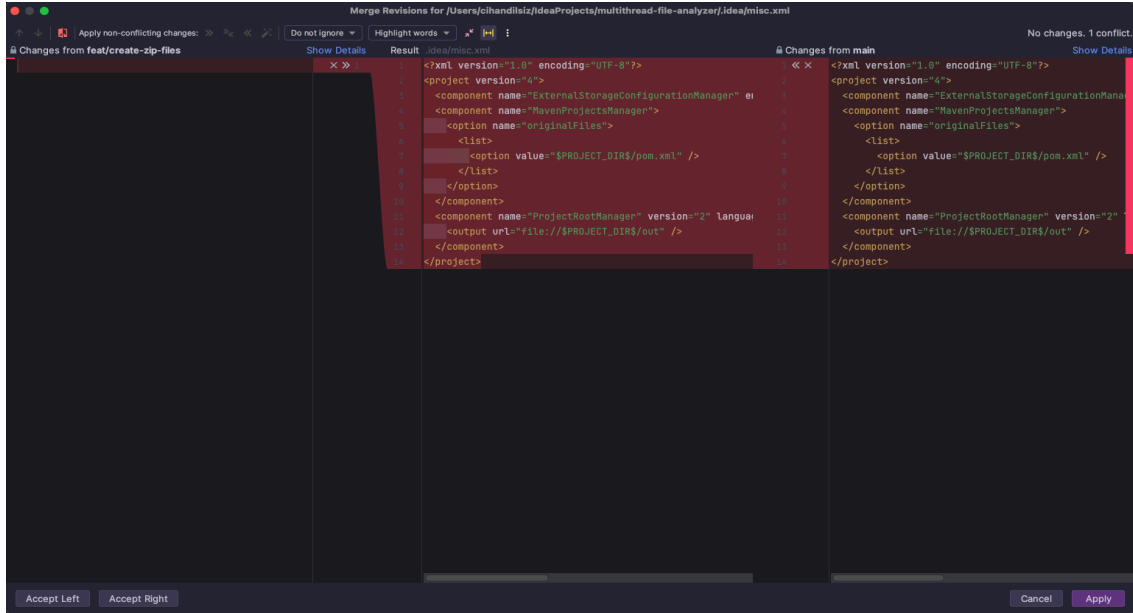


Şekil 8. PR Request

Görsel 2 — Conflictli Dosya: .idea/misc.xml

Açıklama:

Bu görselde **.idea/misc.xml** dosyasında oluşan çakışmalar gösterilmektedir. Sol tarafta **feat:create-zip-files**, sağ tarafta ise main branch'inden gelen içerik yer almaktadır. Her iki tarafın yaptığı değişiklikler karşılaştırmalı olarak gösterilmiş, IDE bu farkları kullanıcıya sunmuştur. Çözüm için sol (**Accept Left**) ya da sağ (**Accept Right**) taraftaki değişiklikler tercih edilebilir veya manuel birleştirme yapılabilir.

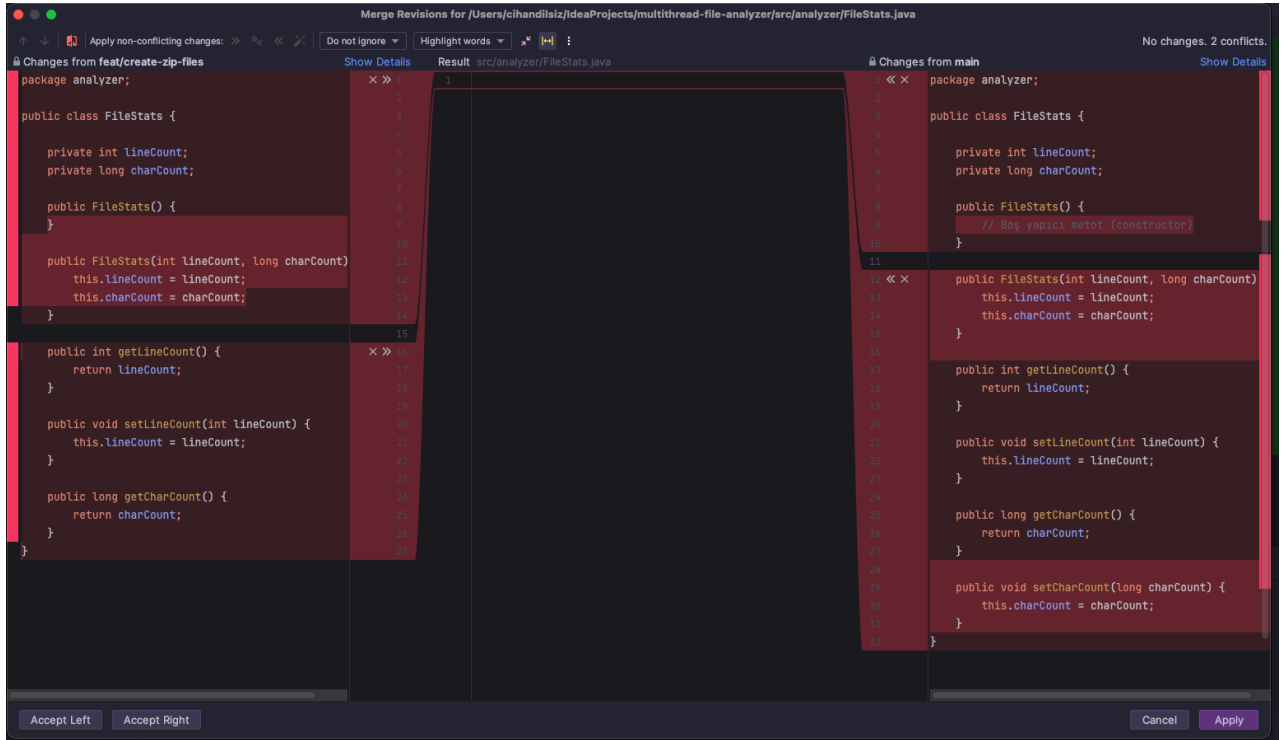


Şekil 9. Resolving Conflict

Görsel 3 — Conflictli Dosya: FileStats.java

Açıklama:

FileStats.java sınıfında hem main hem de **feat:create-zip-files** branch'lerinde farklı yapıcı metodlar (constructors) ve setter metodlar eklenmiştir. Bu nedenle otomatik birleştirme yapılamamıştır. Sol tarafta feat branch'ine ait versiyon, sağ tarafta main'in versiyonu gösterilmektedir. Cihan Dilsiz, iki tarafın metodlarını birleştirerek tüm işlevleri koruyan yeni bir sürüm oluşturmuştur.

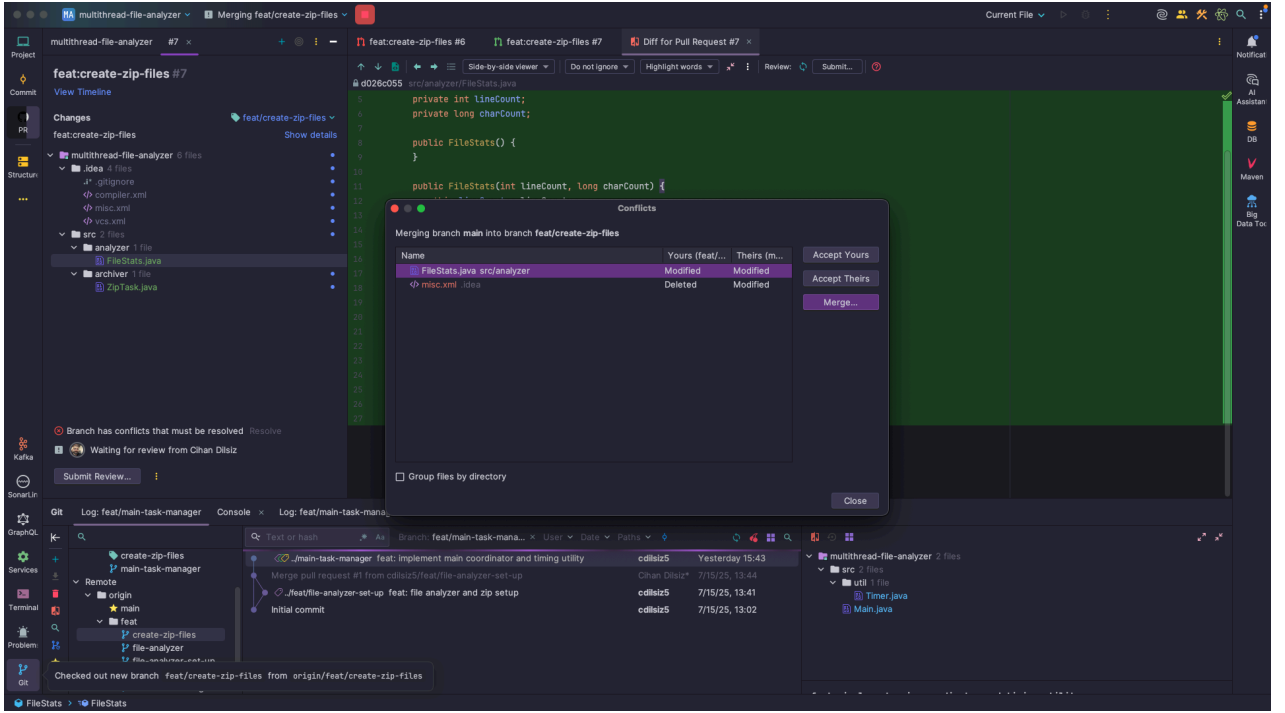


Şekil 10. Resolving Conflict

Görsel 4 — Merge Sonrası Güncel Dosya Durumu

Açıklama:

Bu görselde, merge işlemi tamamlandıktan sonra **FileStats.java** dosyasının yeşil arka planla işaretlenmiş hali görülmektedir. Bu, yapılan değişikliklerin başarılı şekilde birleştirildiğini ve PR'a gönderilmeye hazır olduğunu gösterir. Merge sonrası local testler yapıp dosya çalışır durumda ise Commit ve Push işlemleri yapılır.

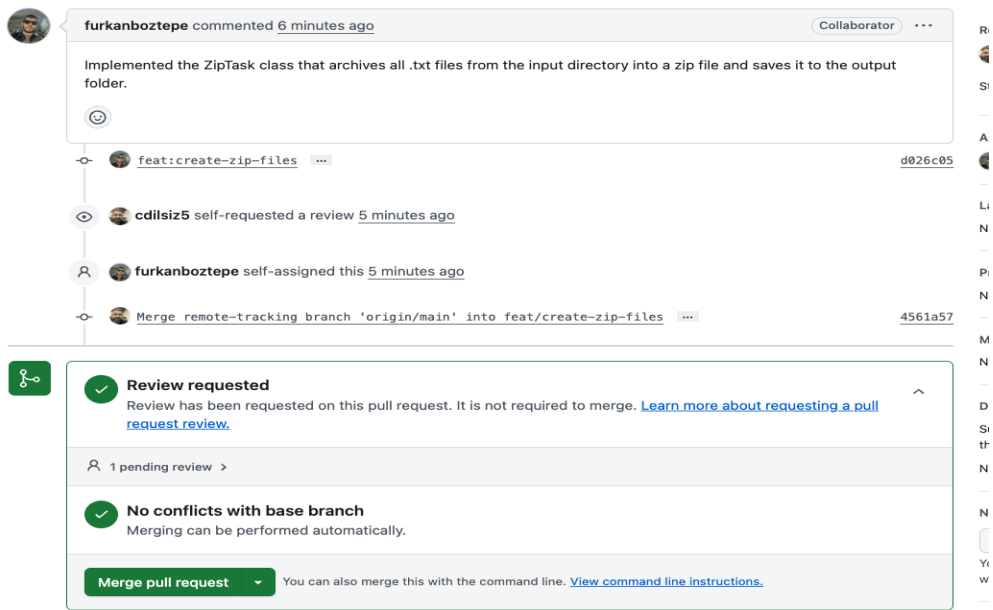


Şekil 11. Guncellenmis Version

Görsel 5 — GitHub Üzerinden PR Detayı ve Merge

Açıklama:

GitHub arayüzünde ilgili PR’a yapılan yorumlar ve self-review bilgileri gözükmektedir. **Furkan Boztepe**, geliştirme tamamlandıktan sonra PR’ı oluşturmuş ve kendini atamıştır. Ardından **Cihan Dilsiz** review talebi yapmıştır. No conflicts with base branch uyarısı çözümden sonra görülmekte ve “**Merge pull request**” butonu aktif hale gelmiştir.



Şekil 12. PR Request Merge Edilmeye Hazır

Görsel 6 — Merge Başarılı ve Branch Kapatıldı

Açıklama:

Son aşamada, **Cihan Dilsiz PR'ı** başarıyla **main branch'ine** merge etmiş ve PR otomatik olarak kapatılmıştır. “**Pull request successfully merged and closed**” ifadesiyle işlem tamamlanmıştır. Ek olarak **feat:create-zip-files** branch'i güvenle silinebileceğini gösteren uyarı da yer almaktadır. Commit mesajında sınıf eklentileri, kullanılan design pattern ve yeni methodlar kısa özet olarak yer almaktadır.

feat:FileUtils Operations #5

[Edit](#) [Code](#)

Merged cdilsiz5 merged 1 commit into [main](#) from [feat/file-utils](#) [now](#)

Conversation 0 Commits 1 Checks 0 Files changed 1 +67 -0

cdilsiz5 commented now Owner

-Class are created. -Singleton design pattern added. -findTxtFiles method created. -deleteInputs method created.

[feat:FileUtils Operations](#) 106f5c3

cdilsiz5 merged commit 5cee067 into main now [Revert](#)

Reviewers

No reviews

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

Notifications [Customize](#)

Pull request successfully merged and closed [Delete branch](#)

You're all set — the [feat/file-utils](#) branch can be safely deleted.

Add a comment

Write Preview

Add your comment here...

Markdown is supported. Data does not add files.