

YAZILIM GELİŞTİRME ORTAM VE ARAÇLARI FİNAL PROJESİ

Ad Soyad: Umut Baglan, Numara: H5210021

Github Gönderimi

1. Repository Oluşturma

Projemizi tutacağımız repository'yı github üzerinden public olarak oluşturuyoruz.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * umutdace / Repository name * yovaFinalProje ←

✓ yovaFinalProje is available.

Great repository names are short and memorable. Need inspiration? How about shiny-guide?

Description (optional)

☒ Public Anyone on the internet can see this repository. You choose who can commit. ←

☐ Private You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore .gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

? You are creating a public repository in your personal account.

Create repository ↓

2. Repository'yı eclipse ekleme

Oluşturduğumuz repository'nin adresini alıp eclipse de bulunan Git Repositories kısmına gelip Ctrl+v'ye basıyoruz. Bu işlemten sonra gelen ekran da eclipse gerekli ekranları otomatik olarak doldurur. Next ve finish'e basarak repo'yu eklemiş oluruz. Repomuzu ekledikten sonra projeye konfigüre etmemiz gerekmektedir. Bunun için projenin üstüne sağ tıklayıp Team → Share Project yapıyoruz. Çıkan ekranda konfigüre etmek istediğimiz repository'i seçip finish'e tıklıyoruz.

Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI: https://github.com/umutdace/yovaFinalProje Local Folder... Local Bundle File...

Host: github.com

Repository path: /umutdace/yovaFinalProje

Connection

Protocol: https

Port:

Authentication

User: umutdace

Password:

☒ Store in Secure Store

? < Back Next > Finish Cancel

Share Project

Configure Git Repository

Select an existing repository or create a new one

☐ Use or create repository in parent folder of project

Repository: yovaFinalProje - C:\Users\ASUS\git\yovaFinalProje\git Create...

Working tree: C:\Users\ASUS\git\yovaFinalProje

Path within repository: Browse...

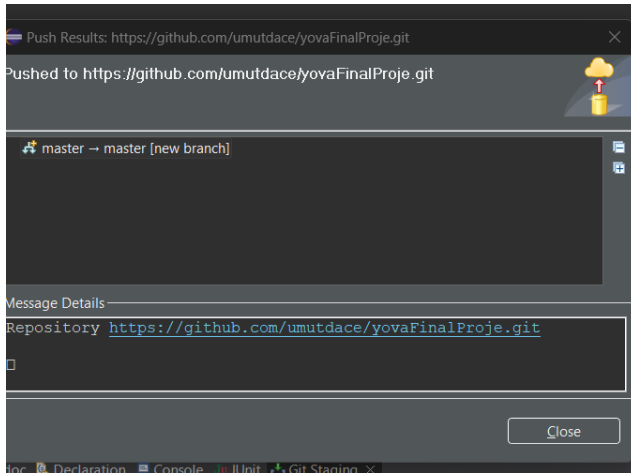
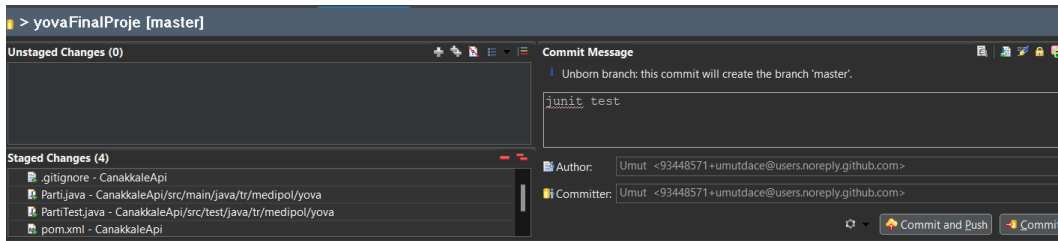
Project	Current Location	Target Location
<input checked="" type="checkbox"/> Canak...	C:/Users/ASUS/eclipse-workspace...	C:/Users/ASUS/git/yovaFinalProje/CanakkaleApi

? Finish Cancel

Repo'muzu proje ile konfigüre ettikten sonra git staging penceresinde proje de yaptığımız değişiklikler gösterilecektir. Burda github repomuza atmak istemediğimiz dosyalar olabilir bunun için .gitignore adında bir dosya oluşturunca bu dosyanın içine .*, target/ ve !.gitignore yazıyoruz .* ile nokta ile başlayan görünmez dosyalarının gönderimi engellenir. target/ ile genelde projenin çıktısı gibi java projelerinde sistemin oluşturduğu dosyaların gönderimi engellenir. !.gitignore ile .gitignore dosyasının gönderilmesi için istisna olması sağlanır.

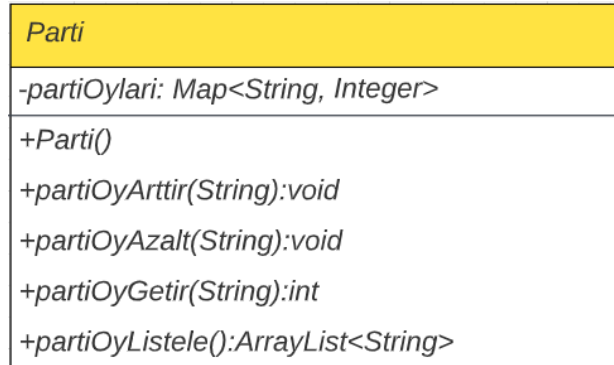
4. Commit ve push

Git staging penceresin de unstaged changes bölümünde bulunan değişikliklerden repomuza göndermek istediğimiz değişiklikleri seçerek staged changes bölümüne taşıyoruz. Commit message alanına mesaj girdikten sonra commit and push butonuna tıklayarak repository'e gönderme işlemini yapıyoruz.



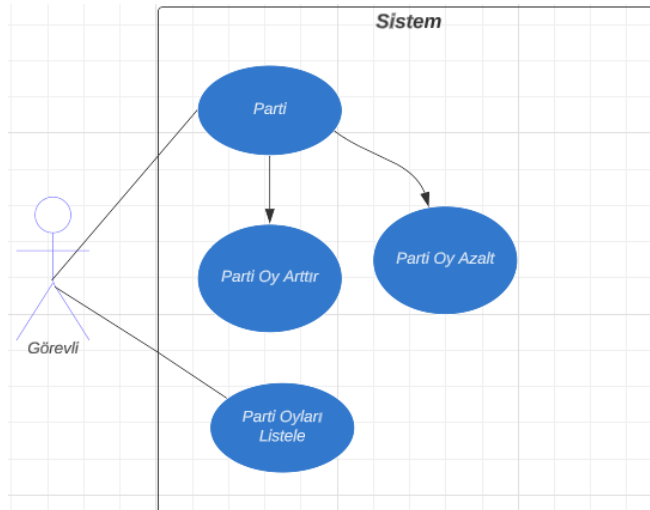
UML Sınıf (Class) ve Kullanım (Use Case) Diyagramları

Uml Class



- Parti sınıfı parti oylarının yönetildiği sınıftır.
- partiOylari Map<String,Integer> tipinde bir nesnedir. Bu map parti adını key ve oy sayısını value olarak tutar.
- Parti() yapıcı(constructor) metodu partiOylari mapini başlatır.
- partiOyArttir(String):void metodu parametre olarak alınan partinin oyunu 1 arttır
- partiOyAzalt(String):void metodu parametre olarak alınan partinin oyunu 1 azaltır.
- partiOyGetir(String):int metodu parametre olarak alınan partinin sadece oy sayısını getirir.
- partiOyListele():ArraList<String> metodu partilerin adlarını ve karşılık gelen oy sayılarını listeyerek getirir.

Kullanım Diyagramı



1. Görevli parti sınıfını başlatır.
2. Görevli partilere ait oyları arttırmak veya azaltmak için partiOyArttir veya partiOyAzalt metotlarını kullanır.
3. Görevli partilere ait oy bilgilerini listeleyen partiOyListele metodunu çağırır.
4. Sistem, parti oylarının listesini döndürür.
5. Görevli döndürülen oy bilgilerini görüntüler.

Kodlar ve Açıklamaları

Parti Sınıfı

```
public class Parti {  
    private Map<String, Integer> partiOylari;  
  
    public Parti() {  
        partiOylari = new HashMap<String, Integer>();  
    }  
  
    public void partiOyArttir(String parti) {  
        partiOylari.putIfAbsent(parti, 0);  
        partiOylari.put(parti, partiOylari.get(parti) + 1);  
    }  
  
    public void partiOyAzalt(String parti) {  
        if (partiOylari.containsKey(parti)) {  
            partiOylari.put(parti, partiOylari.get(parti) - 1);  
        } else {  
            throw new IllegalArgumentException(parti + " partisi bulunamadı...");  
        }  
    }  
  
    public int partiOyGetir(String parti) {  
        if (partiOylari.containsKey(parti)) {  
            return partiOylari.get(parti).intValue();  
        } else {  
            throw new IllegalArgumentException(parti + " partisi bulunamadı...");  
        }  
    }  
  
    public ArrayList<String> partiOyListele() {  
        ArrayList<String> partiOyBilgiListesi = new ArrayList<String>();  
        for (Map.Entry<String, Integer> entry : partiOylari.entrySet()) {  
            String partiAdi = entry.getKey();  
            int oySayisi = entry.getValue();  
            String partiBilgi = partiAdi + ": " + oySayisi;  
            partiOyBilgiListesi.add(partiBilgi);  
        }  
        return partiOyBilgiListesi;  
    }  
}
```

Parti sınıfında partilerin ve oyların tutulacağı map tanımlanır. Parti sınıfı kullanılmak istendiğinde new Parti() ifadesi ile constructor çalışır ve yeni bir map nesnesi başlatıp hashmap örneği oluşturur.

PartiOyArttir fonksiyonu bir string parametre alır bu parametre parti adını temsil etmektedir. putIfAbsent fonksiyonu ile eğer parti map içinde yoksa parti adı key 0 değeri value olarak partiOylarina eklenir. Bu işlemden sonra put fonksiyonu ile partiye 1 yok eklenir.

PartiOyAzalt fonksiyonu parametre aldığı partinin oyunu bir azaltan fonksiyondur. Öncelikle parametre alınan partinin map içinde olup olmadığı containsKey fonksiyonu ile kontrol edilir eğer parti varsa put metodu ile parti'den 1 oy azaltılır. Partinin olmaması durumunda geriye hata mesajı döndürülür.

PartiOyGetir fonksiyonu test işlemi için yazılmış sadece parti oylarını getiren fonksiyondur. Parametre alınan parti containsKey fonksiyonu ile olup olmadığı kontrol edilir eğer varsa partinin sahip olduğu oy sayısı geri döndürülür. Partinin olmaması durumunda geriye hata mesajı döndürülür.

PartiOyListele fonksiyonu parti oylarını listeye çevirip geri döndüren fonksiyondur. Öncelikle bir ArrayList tanımlanır daha sonra foreach döngüsünde entry değişkeni

entrySet() fonksiyonundan dönen setin içinde döner. Getkey() fonksiyonu ile parti adı getValue() fonksiyonu ile ise partinin oy sayısı getirilir ve değişkenlere atılır. Daha sonra bu iki değer bir araya getirilip string tipinde bir değişkene atanır. Bu string değişkeni en son arrayliste eklenir. Foreach döngüsü bitince arraylist geri döndürülür.

PartiTest Sınıfı

```
public class PartiTest {  
    @Test  
    public void testPartiOyArttir() {  
        Parti parti = new Parti();  
        for(int i = 0; i < 5; i++)  
            parti.partiOyArttir("A");  
        for(int i = 0; i < 8; i++)  
            parti.partiOyArttir("B");  
        for(int i = 0; i < 10; i++)  
            parti.partiOyArttir("C");  
  
        int aPartiSonuc = parti.partiOyGetir("A");  
        int bPartiSonuc = parti.partiOyGetir("B");  
        int cPartiSonuc = parti.partiOyGetir("C");  
  
        assertEquals(5, aPartiSonuc);  
        assertEquals(8, bPartiSonuc);  
        assertEquals(10, cPartiSonuc);  
    }  
}
```

PartiTest sınıfı test işlemlerinin yapıldığı sınıftır. testPartiOyArttir fonksiyonu ile oy arttırma işleminin testi yapılmaktadır. Öncelikle yeni bir parti nesnesi oluşturulur daha sonra for döngüleri ile beraber döngü bitene kadar belirtilen partilere partiOyArttir fonksiyonu ile oy eklenir. Ekleme işlemi tamamlanınca her parti sonucu için değişken oluşturulur. Oluşturulan bu değişkenlere partiOyGetir fonksiyonu ile partinin oyu getirilir ve bu değişkenlere atanır. assertEquals fonksiyonu ile beklenen oy sayısı ve sonuç olarak gelen oy sayısının eşleşip eşleşmediği kontrol edilir.

```
@Test  
public void testPartiOyAzalt() {  
    Parti parti = new Parti();  
    for(int i = 0; i < 12; i++)  
        parti.partiOyArttir("A");  
    for(int i = 0; i < 20; i++)  
        parti.partiOyArttir("B");  
    for(int i = 0; i < 30; i++)  
        parti.partiOyArttir("C");  
  
    for(int i = 0; i < 5; i++)  
        parti.partiOyAzalt("A");  
    for(int i = 0; i < 12; i++)  
        parti.partiOyAzalt("B");  
    for(int i = 0; i < 15; i++)  
        parti.partiOyAzalt("C");  
  
    int aPartiSonuc = parti.partiOyGetir("A");  
    int bPartiSonuc = parti.partiOyGetir("B");  
    int cPartiSonuc = parti.partiOyGetir("C");  
  
    assertEquals(7, aPartiSonuc);  
    assertEquals(8, bPartiSonuc);  
    assertEquals(15, cPartiSonuc);  
}
```

Parti oy azaltma işleminin testi için öncelikle bir parti nesnesi oluşturulur daha sonra for döngüleri ile önce partilere oylar eklenir. Partilere oy eklendikten sonra aynı işlem bu sefer partiOyAzalt fonksiyonu kullanılarak parti oyları azaltılarak yapılır. Bu işlemden sonra partiOyGetir fonksiyonu ile getirilen partilerin oyları her parti için oluşturulan sonuç değişkenlerine atanır. En son assertEquals ile beklenen değer ve sonuç değerinin kontrolü yapılır.

```

@Test
public void testPartiOyGetir() {

    Parti parti = new Parti();
    for(int i = 0; i < 5; i++)
        parti.partiOyArttir("A");
    for(int i = 0; i < 8; i++)
        parti.partiOyArttir("B");
    for(int i = 0; i < 10; i++)
        parti.partiOyArttir("C");

    for(int i = 0; i < 2; i++)
        parti.partiOyAzalt("A");
    for(int i = 0; i < 4; i++)
        parti.partiOyAzalt("B");
    for(int i = 0; i < 7; i++)
        parti.partiOyAzalt("C");

    int aPartiSonuc = parti.partiOyGetir("A");
    int bPartiSonuc = parti.partiOyGetir("B");
    int cPartiSonuc = parti.partiOyGetir("C");

    assertEquals(3, aPartiSonuc);
    assertEquals(4, bPartiSonuc);
    assertEquals(3, cPartiSonuc);
}

```

Parti oy getirme işlemini yapan metodun testi için yazılmış bir fonksiyondur. Bu fonksiyonda öncelikle parti nesnesi oluşturulur daha sonra for döngüleri ile partiOyArttir ve PartiOyAzalt fonksiyonları kullanılarak parti oy ekleme ve azaltma işlemleri yapılır bu işlemler sonucunda partiOyGetir fonksiyonu kullanılır. partiOyGetir fonksiyonun gelen sonuçlar değişkene atıldıktan sonra beklenen ve sonuç değerler kontrol edilir.

```

@Test
public void testPartiOyListele() {

    Parti parti = new Parti();

    for(int i = 0; i < 5; i++)
        parti.partiOyArttir("A");
    for(int i = 0; i < 8; i++)
        parti.partiOyArttir("B");
    for(int i = 0; i < 10; i++)
        parti.partiOyArttir("C");

    ArrayList<String> sonucListesi = parti.partiOyListele();

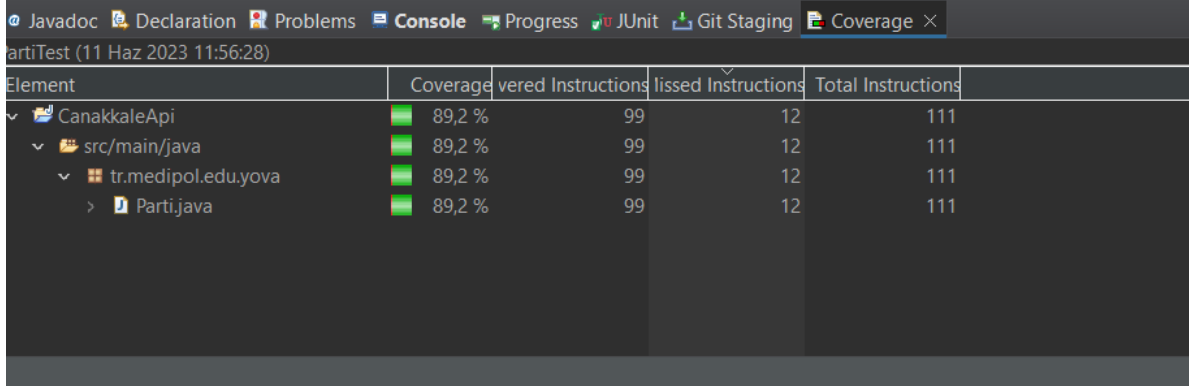
    assertEquals(3, sonucListesi.size());
    assertTrue(sonucListesi.contains("A: 5"));
    assertTrue(sonucListesi.contains("B: 8"));
    assertTrue(sonucListesi.contains("C: 10"));
}

```

Parti oy listeleme işlemi yapan metodun testi için yazılmış fonksiyondur. Fonksiyonda öncelikle yeni bir parti nesnesi oluşturulur daha sonra for döngüleri ile partiOyArttir() fonksiyonu kullanılarak partilere oy ekleme işlemi yapılır. partiOyListele fonksiyonundan bize bir liste döneceği için ArrayList<String> tipind bir değişken tanımlanır ve partiOyListele fonksiyonundan dönen liste bu değişkene atanır. Daha sonra assertEquals ile ilk önce listenin boyutu kontrol edilir. Boyut kontrol edildikten sonra assertTrue ile dönen listede istediğimiz partilerin sonuçlarının olup olmadığı contains fonksiyonu ile bakılır.

Test Kapsama (Coverage)

Projenin üstüne gelip sağ tık 'Coverage As' → 'JUnit Test' dedikten sonra coverage penceresi açılır bu pencerede test kapsama oranını görebiliriz.



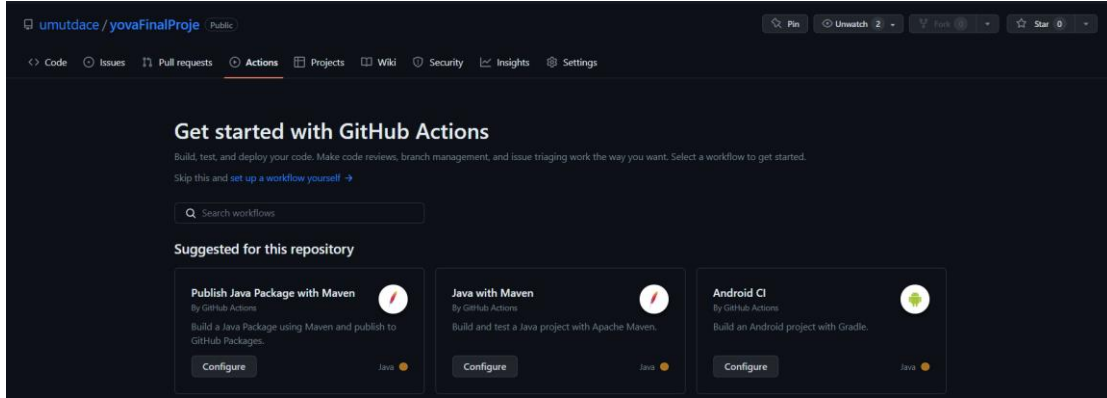
The screenshot shows the Coverage window in an IDE. The table displays the following data:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
CanakkaleApi	89,2 %	99	12	111
src/main/java	89,2 %	99	12	111
tr.medipol.edu.yova	89,2 %	99	12	111
Parti.java	89,2 %	99	12	111

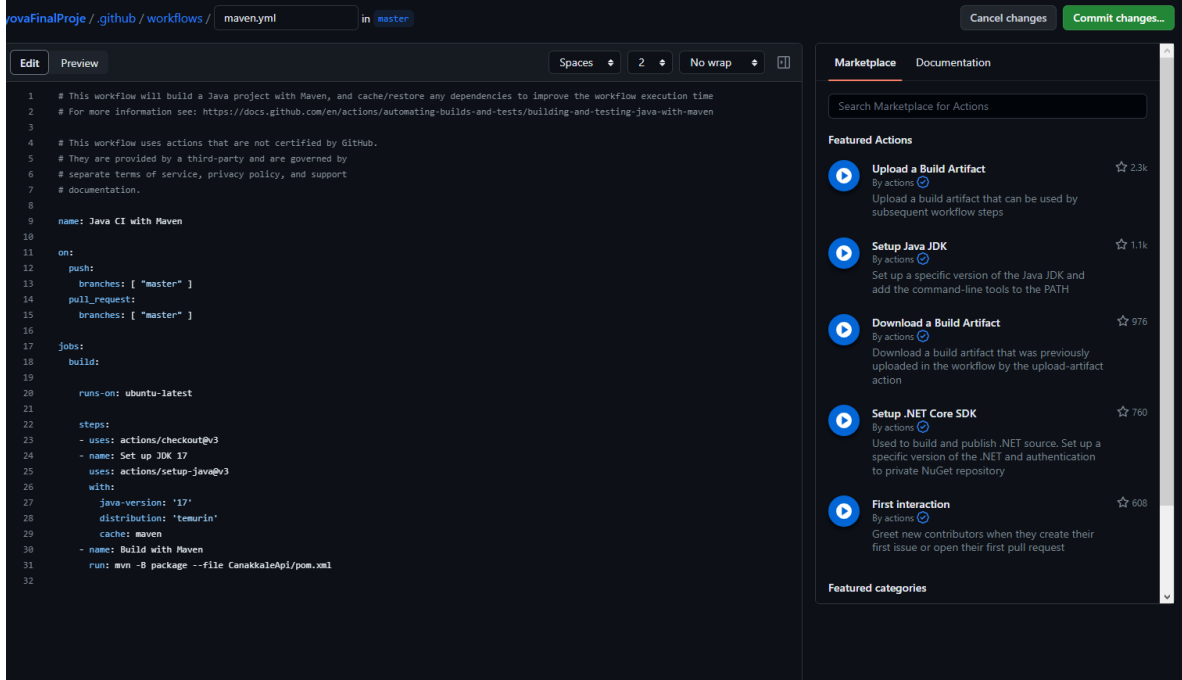
Github Actions Maven Build ve PR Kapsama Oranı

Maven Build

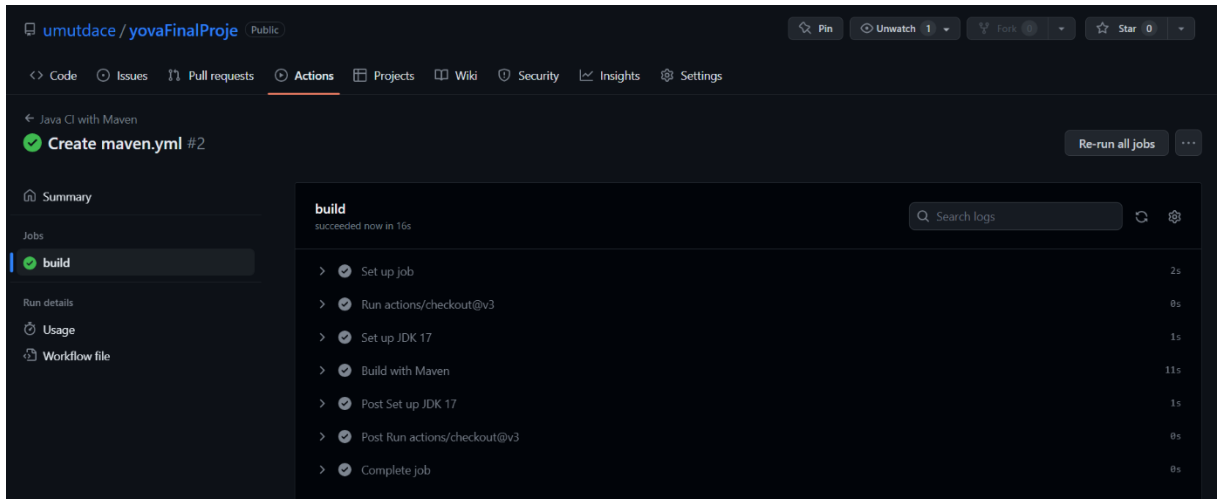
Repository sayfamıza girdikten sonra actions bölümünden 'Java with Maven' özelliğini konfigüre edilmelidir.



'Configure' butonuna tıkladıktan sonra karşımıza aşağıdaki görsel de bulunan ekran çıkacaktır. Bu ekran maven build için gerekli komutların içerildiği dosyadır. Bu dosya da kendi projemize göre değişiklik yapmamız gerekebilir. Örneğin java versiyonunu değiştirmemiz gerekebilir veya pom.xml dosyamızın dosya yolunu özel olarak belirtmemiz gerekebilir. Bu değişiklikleri yaptıktan sonra commit changes butonuna tıkliyoruz.



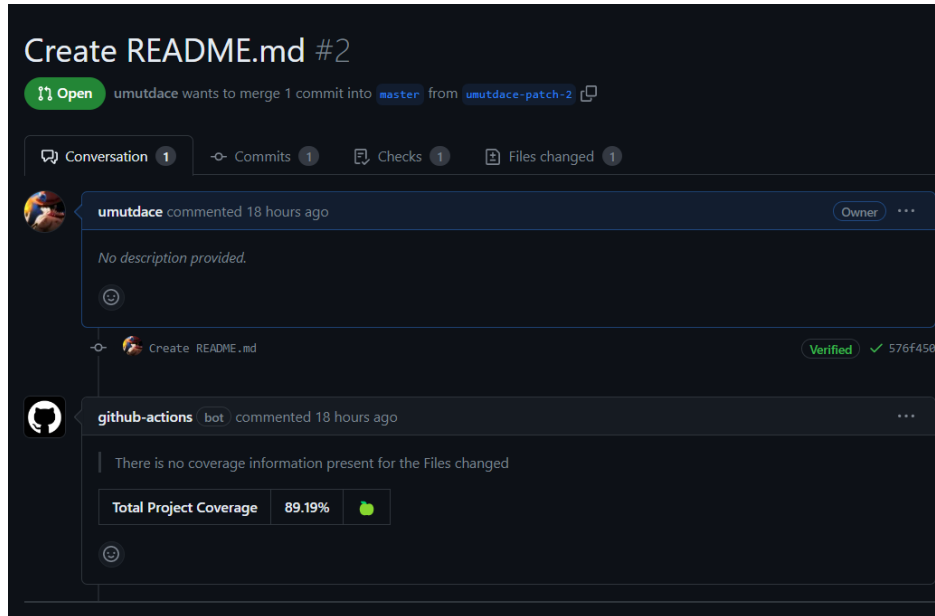
Commit changes butonuna tıkladıktan sonra tekrar actions sayfasına geliyoruz. Actions ekranına geldiğinde maven build action'ı çalışmaya başlamış olacaktır. Daha ayrıntılı görmek için üstüne tıklayarak çalışan adımların başarılı olup olmadığını görebiliriz.



PR Kapsama Oranı

```
yovaFinalProje / .github / workflows / maven.yml
4 # This workflow uses actions that are not certified by GitHub.
5 # They are provided by a third-party and are governed by
6 # separate terms of service, privacy policy, and support
7 # documentation.
8
9 name: Java CI with Maven
10
11 on:
12   push:
13     branches: [ "master" ]
14   pull_request:
15     branches: [ "master" ]
16
17 jobs:
18   build:
19
20     runs-on: ubuntu-latest
21
22     permissions:
23       pull-requests: 'write'
24
25     steps:
26       - uses: actions/checkout@v3
27       - name: Set up JDK 17
28         uses: actions/setup-java@v3
29         with:
30           java-version: '17'
31           distribution: 'temurin'
32           cache: maven
33       - name: Build with Maven
34         run: mvn -B package --file CanakkaleApi/pom.xml
35
36       - name: Add coverage to PR
37         id: jacoco
38         uses: madrapps/jacoco-report@v1.3
39         with:
40           paths: ${{ github.workspace }}/CanakkaleApi/target/site/jacoco/jacoco.xml
41           token: ${{ secrets.GITHUB_TOKEN }}
42           min-coverage-overall: 40
43           min-coverage-changed-files: 60
```

Öncelikle daha önce oluşturduğumuz maven.yml dosyasına geliyoruz daha sonra permissions alanında gerekli olan izinleri belirtiyoruz. Bu işlemten sonra yeni bir adım oluşturup jacoco ayarını yapıp projemizde bulunan jacoco.xml'in yolunu belirtiyoruz ve kaydediyoruz.



Repository'nin ana sayfasından yeni bir readme.md dosyası eklememiz gerekiyor. Bunun için 'Add a README' butonuna tıkladıktan sonra 'Commit Changes' butonuna tıklıyoruz. Daha sonra önüme gelen ekranda 'Create a new branch for this commit and start a pull request' seçeneğini seçip yeni bir pull request başlatıyoruz. İşlem sonucunda PR kapsama oranı yukarıdaki gibi gözükecektir.