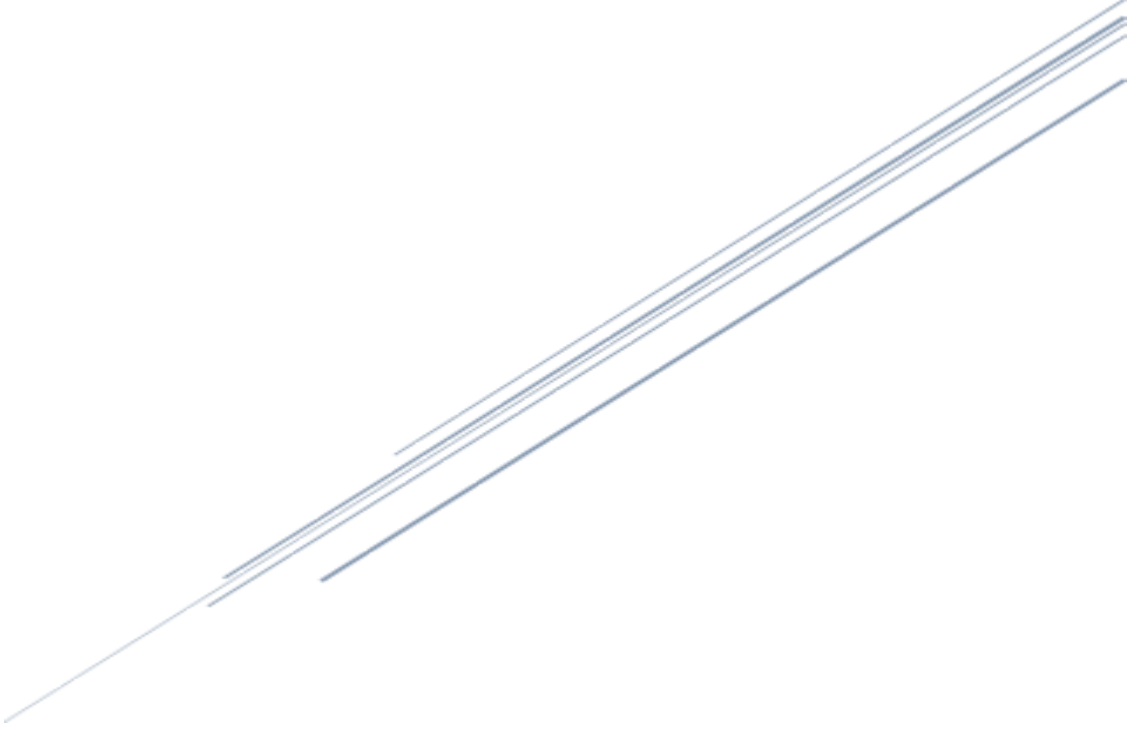


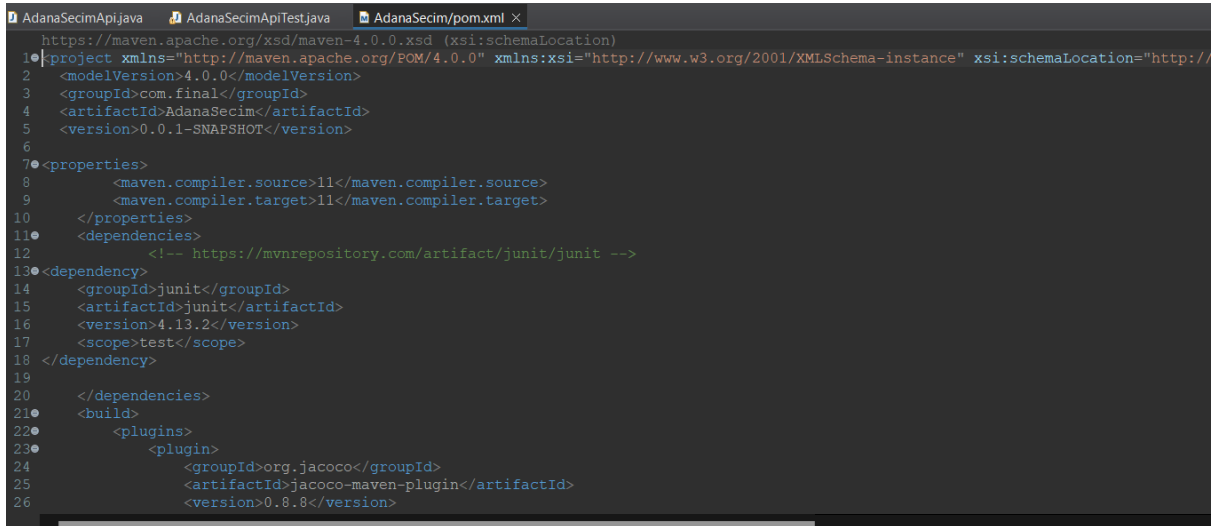
T.C.
İSTANBUL MEDİPOL ÜNİVERSİTESİ
YAZILIM GELİŞTİRME ORTAM VE ARAÇLARI
FİNAL



BURAK EMRE DALBUDAK

H5210047

Oy verme sistemini gerçekleştirmek için Eclipse üzerinde Maven projesi oluşturuldu,gerekli bağımlılıklar ve düzenlemelerden sonra (resim.1.1) tr.edu.medipol.yova paketi altında AdanaSecimApi sınıfı oluşturuldu.AdanaSecimApi sınıfı,parti oy sayılarını artırma,azaltma ve listeleme işlevlerini sağlamaktadır.



resim 1.1

AdanaSecimApi sınıfı,aşağıda belirtilen 3 özelliğe sahip metodu içermektedir:

partiOyArttir(String partiAdi): Bu metot,belirtilen partiye ait oy sayısını bir artırır. Metot, "A", "B" veya "C" değerlerinden birini partiAdi parametresi olarak alır.

partiOyAzalt(String partiAdi): Bu metot,belirtilen partiye ait oy sayısını bir azaltır. Metot, "A", "B" veya "C" değerlerinden birini partiAdi parametresi olarak alır.Eğer oy sayısı zaten sıfırsa, yani ilgili partiye ait oy sayısı 0 ise,IllegalArgumentException hatası fırlatılır.

partiOyListele(): Bu metot,mevcut oy durumunu metin formatında döndürür.Geri dönen metinde "Parti A Oy:", "Parti B Oy:" ve "Parti C Oy:" ifadeleriyle ayrılmış olarak her bir partiye ait oy sayıları yer alır.

AdanaSecimApi sınıfı üzerinde gerekli metodlar ayarlandıktan sonra,doğrulama amaçlı AdanaSecimApiTest adında bir test sınıfı oluşturuldu.Bu test sınıfı,AdanaSecimApi sınıfının doğru çalıştığını doğrulamak için aşağıdaki testleri içermektedir:

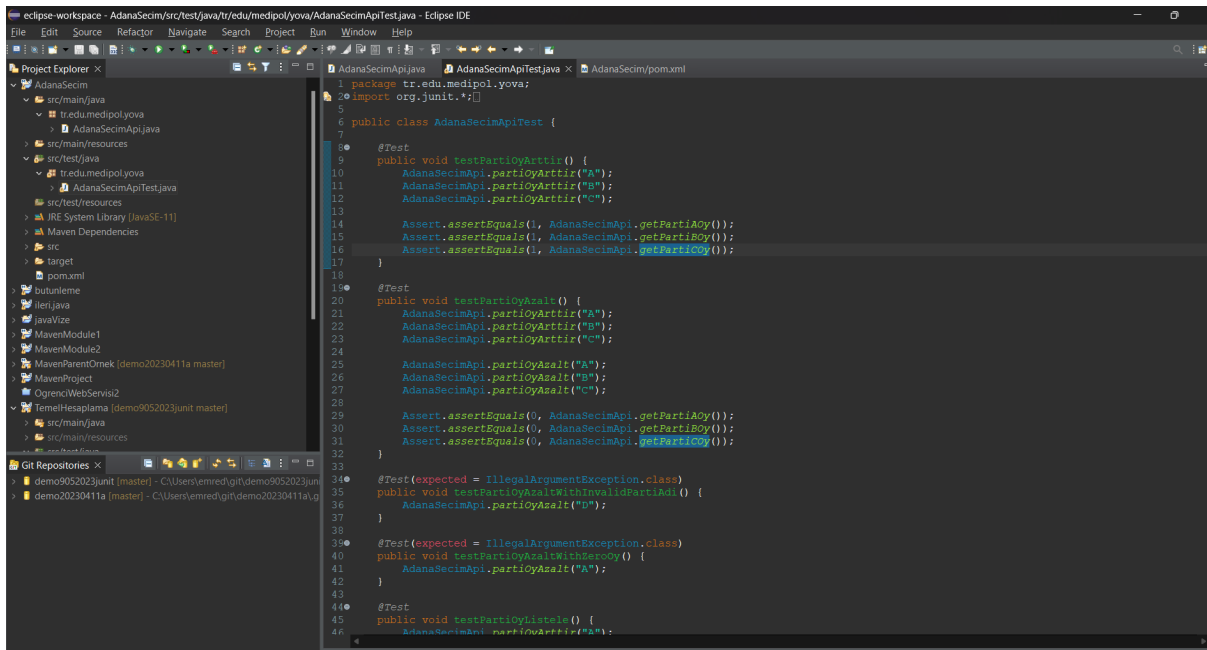
testPartiOyArttir(): Bu test,partiOyArttir() metodunu test eder. Test, "A", "B" ve "C" partilerine sırasıyla birer oy artırıldığında, her bir partinin oy sayısının 1 artması gerektiğini doğrular.

testPartiOyAzalt(): Bu test,partiOyAzalt() metodunu test eder.Test, "A", "B" ve "C" partilerine sırasıyla birer oy artırıldıktan sonra her bir partinin oylarının birer azaltıldığında,her bir partinin oy sayısının 0 olması gerektiğini doğrular.

testPartiOyAzaltWithInvalidPartiAdi(): Bu test,partiOyAzalt() metodunu geçersiz bir parti adıyla çağırıldığında IllegalArgumentException hatası fırlatılıp fırlatılmadığını test eder.Bu testte "D" adında bir parti olmadığı için IllegalArgumentException hatası beklenir.

testPartiOyAzaltWithZeroOy(): Bu test,partiOyAzalt() metodunu oy sayısı sıfır olan bir partiye çağırıldığında IllegalArgumentException hatası fırlatılıp fırlatılmadığını test eder.Bu testte "A" partisine ait oy sayısı zaten sıfır olduğu için IllegalArgumentException hatası beklenir.

testPartiOyListele(): Bu test,partiOyListele() metodunu test eder. Test, "A", "B" ve "C" partilerine sırasıyla birer oy artırıldığında, her bir partinin doğru şekilde metin formatında oy sayısını döndürdüğünü doğrular.

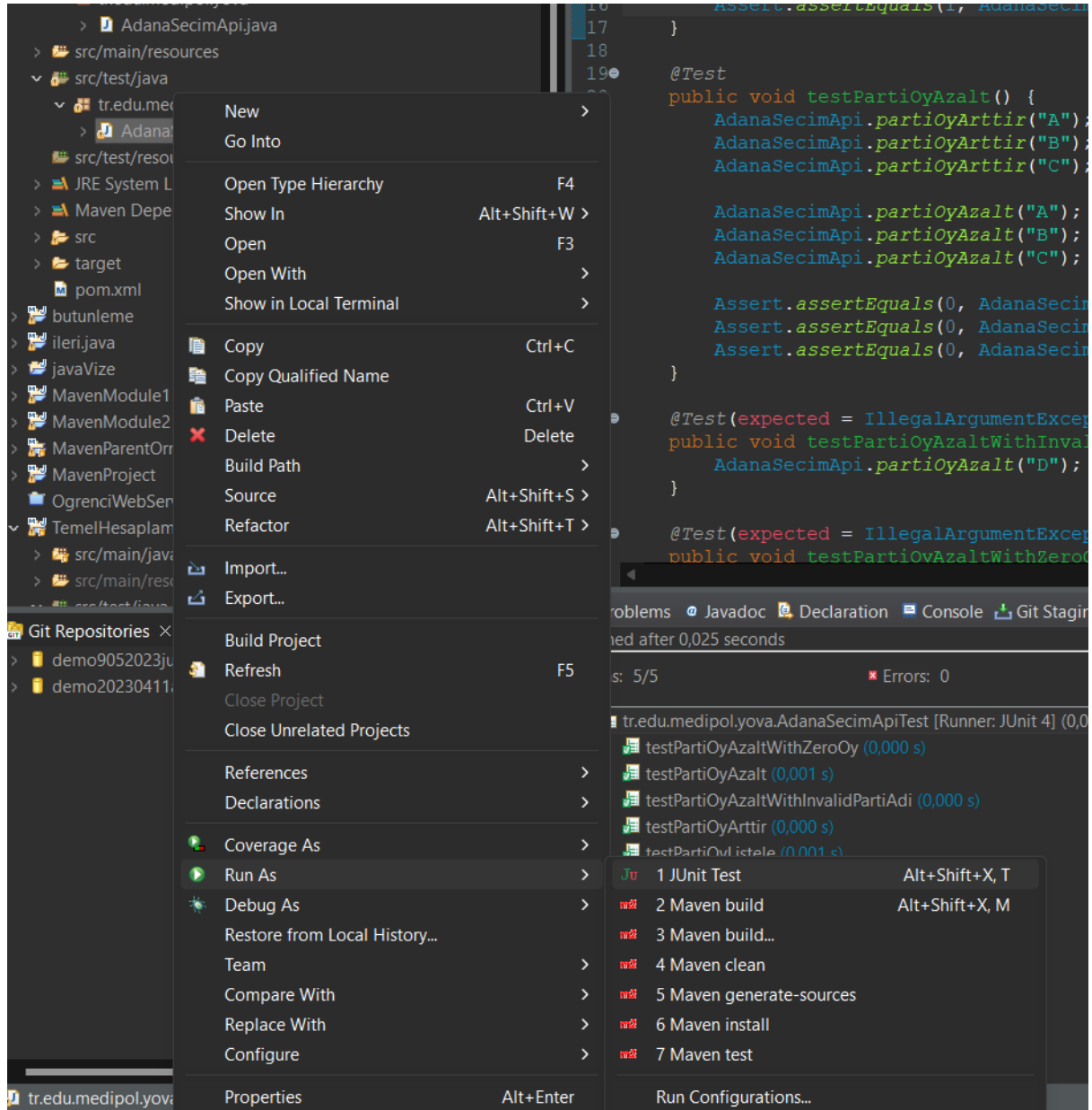


```
1 package tr.edu.medipol.yova;
2 import org.junit.*;
3
4 public class AdanaSecimApiTest {
5
6     @Test
7     public void testPartiOyArttir() {
8         AdanaSecimApi.partiOyArttir("A");
9         AdanaSecimApi.partiOyArttir("B");
10        AdanaSecimApi.partiOyArttir("C");
11
12        Assert.assertEquals(1, AdanaSecimApi.getPartiAOy());
13        Assert.assertEquals(1, AdanaSecimApi.getPartiBOy());
14        Assert.assertEquals(1, AdanaSecimApi.getPartiCOy());
15    }
16
17     @Test
18     public void testPartiOyAzalt() {
19         AdanaSecimApi.partiOyArttir("A");
20         AdanaSecimApi.partiOyArttir("B");
21         AdanaSecimApi.partiOyArttir("C");
22
23         AdanaSecimApi.partiOyAzalt("A");
24         AdanaSecimApi.partiOyAzalt("B");
25         AdanaSecimApi.partiOyAzalt("C");
26
27         Assert.assertEquals(0, AdanaSecimApi.getPartiAOy());
28         Assert.assertEquals(0, AdanaSecimApi.getPartiBOy());
29         Assert.assertEquals(0, AdanaSecimApi.getPartiCOy());
30    }
31
32     @Test(expected = IllegalArgumentException.class)
33     public void testPartiOyAzaltWithInvalidPartiAdi() {
34         AdanaSecimApi.partiOyAzalt("D");
35    }
36
37     @Test(expected = IllegalArgumentException.class)
38     public void testPartiOyAzaltWithZeroOy() {
39         AdanaSecimApi.partiOyAzalt("A");
40    }
41
42     @Test
43     public void testPartiOyListele() {
44         AdanaSecimApi.partiOyListele();
45    }
46 }
```

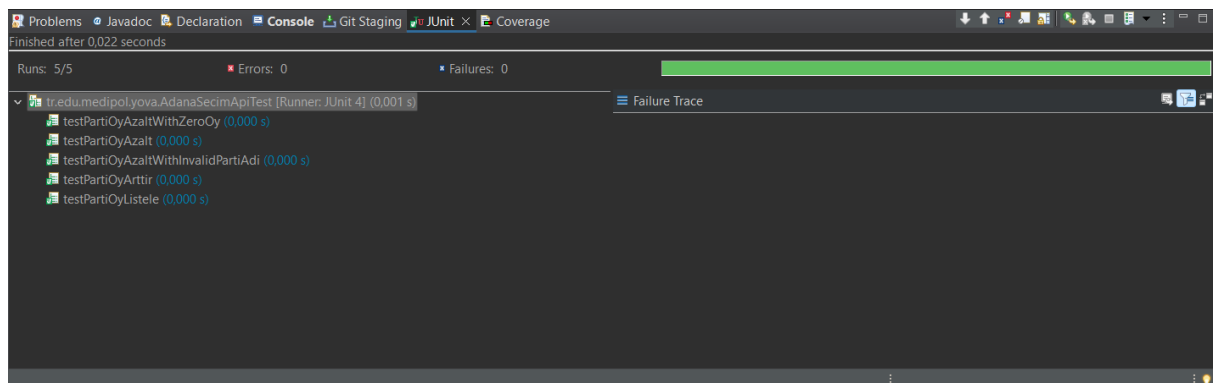
resim 1.2

Ayrıca AdanaSecimApi sınıfının doğru çalıştığını ve beklenen davranışları sergilediğini doğrulamak için JUnit kullanıldı.Bu testlerin temel amacı,AdanaSecimApi'nin metotlarının beklenen sonuçları doğru şekilde üretip üretmediğini ve hatalı girişlere uygun tepkiler verip vermediğini kontrol edilmesidir.Testlerin sonuçları,AdanaSecimApi sınıfının beklenen davranışlarını sergileyerek işlevselliğini doğrulamayı amaçlamaktadır.Aşağıda

oluşturduğumuz bu testlerin başarıyla tamamlandığını gösteren ekran görüntüleri (resim 2.1-2.2) yer almaktadır.

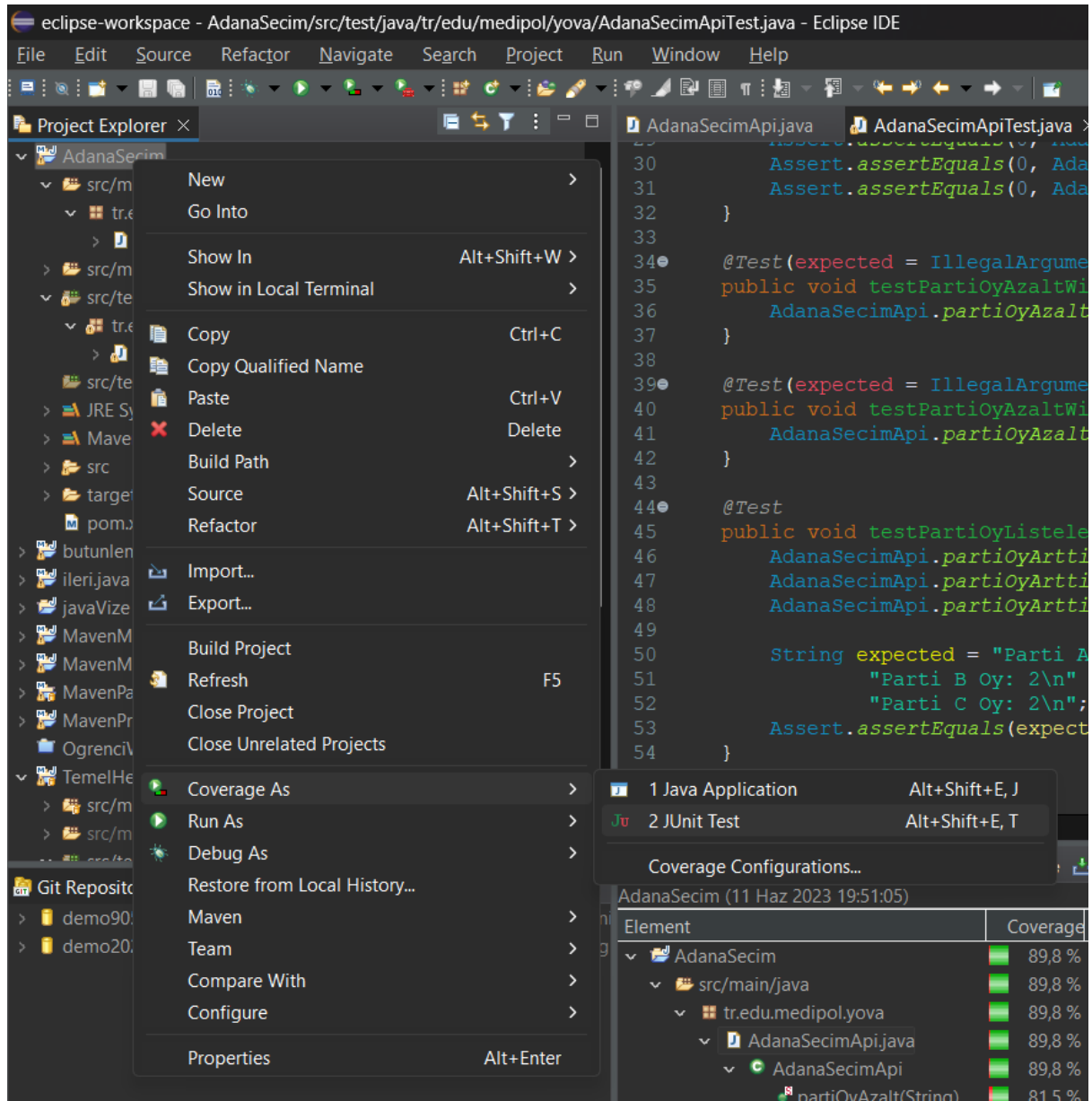


resim 2.1

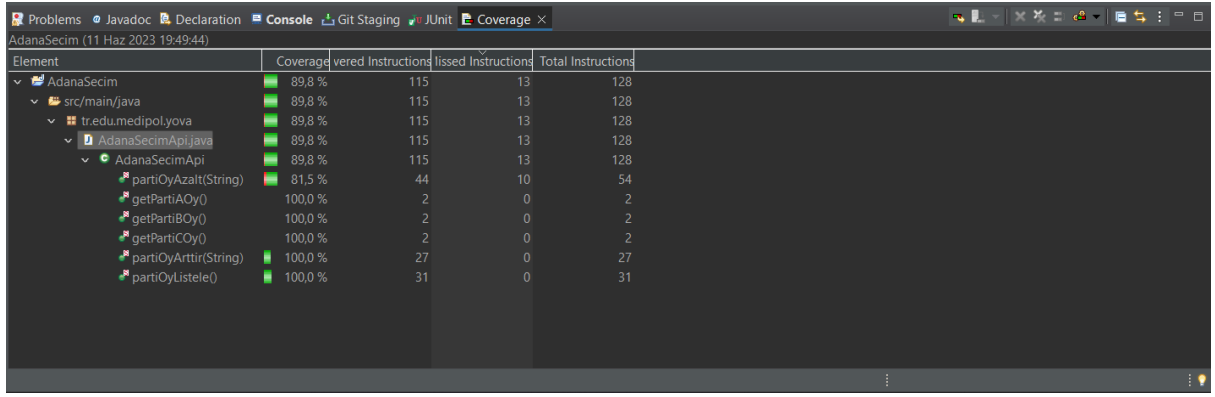


resim 2.2

Bu aşamadan sonra ayrıca coverage (kod kapsamı) metriği kullanılarak yazılım testlerinin ne kadarının kod tarafından kontrol edildiği ölçülmüştür. Bu metrik, test sürecinin etkinliğini değerlendirmek ve yazılımın hangi kısımlarının test edilmediğini belirlemek açısından önemlidir. Aşağıda belirtilen ekran görüntüsünde (resim.3.1) AdanaSecimApi sınıfı için gerçekleştirilen testlerin yüksek bir code coverage oranı elde ettiği gözlemlenmiştir.



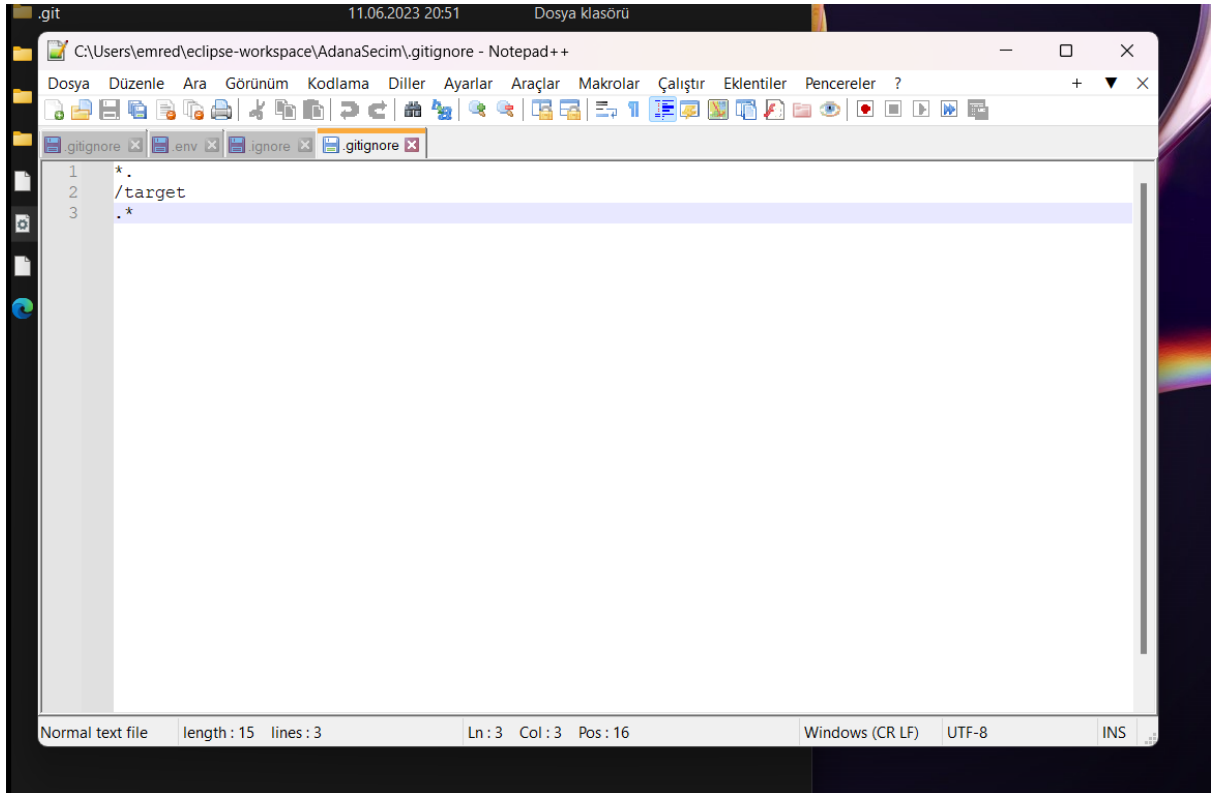
resim 3.1



Element	Coverage	Vered Instructions	Used Instructions	Total Instructions
AdanaSecim	89,8 %	115	13	128
src/main/java	89,8 %	115	13	128
tr.edu.medipol.yova	89,8 %	115	13	128
AdanaSecimApi.java	89,8 %	115	13	128
AdanaSecimApi	89,8 %	115	13	128
partiOyAzalt(String)	81,5 %	44	10	54
getPartiAOy()	100,0 %	2	0	2
getPartiBOy()	100,0 %	2	0	2
getPartiCOy()	100,0 %	2	0	2
partiOyArttir(String)	100,0 %	27	0	27
partiOyListele()	100,0 %	31	0	31

resim 3.2

Gerekli testleri ve düzenlemeler bittikten sonra adı "yova20230611" olan repository'e başarıyla dosyaları göndermek için belirli adımları takip ettik. İlk olarak, ".gitignore" adında bir metin dosyası oluşturduk (resim 4.1) ve bu dosyada GitHub'a yüklenmesini istemediğimiz dosyaların listesini belirttik. Bu sayede istenmeyen dosyaların gönderimini engelledik. Ardından, dosya klasörü içinde Git Bash'i açtık ve "git init" komutunu kullanarak bir Git deposu oluşturduk. Daha sonra, "git add" komutuyla dosyaları staged (hazır) durumuna getirerek takip edilmelerini sağladık. Sonrasında, "git commit" komutunu uyguladık. "git remote" komutunu kullanarak uzak repository bağlantısını belirttik ve son olarak "git push" komutuyla değişiklikleri uzak repository'e gönderdik (resim 4.2). Bu adımların başarıyla tamamlanmasıyla dosyalarımızı "yova20230611" repository'sine gönderdik ve paylaşıma hazır hale getirdik. (resim 4.3)



resim 4.1

```
MINGW64:/c/Users/emred/eclipse-workspace/AdanaSecim

emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim
$ git init
Initialized empty Git repository in C:/Users/emred/eclipse-workspace/AdanaSecim/.git/

emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim (master)
$ git add .

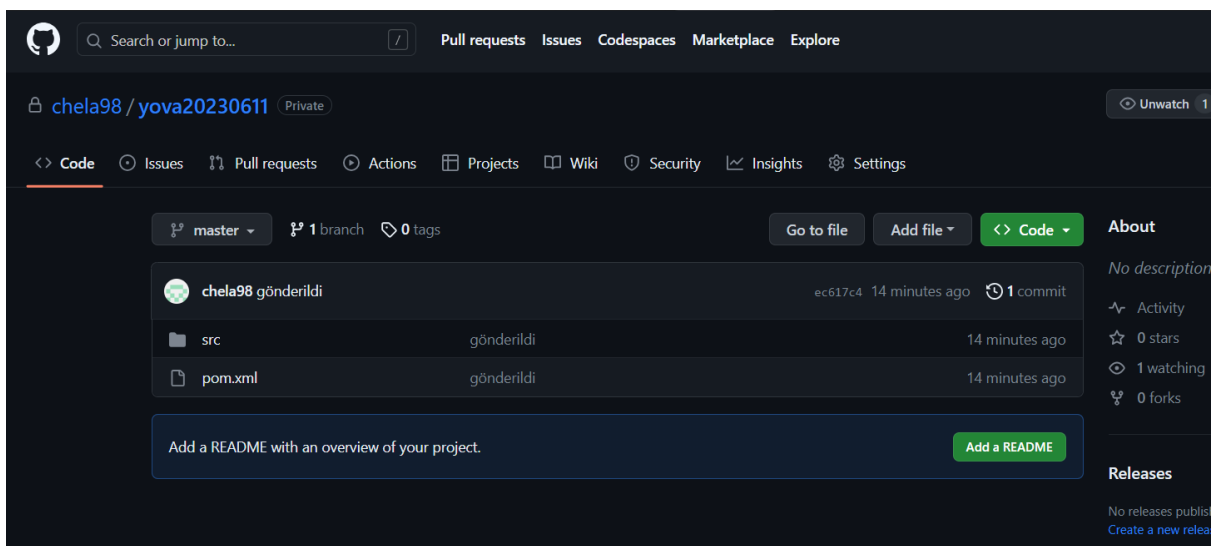
emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim (master)
$ git commit -m "gönderildi"
[master (root-commit) ec617c4] gönderildi
3 files changed, 159 insertions(+)
create mode 100644 pom.xml
create mode 100644 src/main/java/tr/edu/medipol/yova/AdanaSecimApi.java
create mode 100644 src/test/java/tr/edu/medipol/yova/AdanaSecimApiTest.java

emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim (master)
$ git remote add origin https://github.com/chela98/yova20230611.git

emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim (master)
$ git push -u origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (18/18), 2.07 KiB | 2.07 MiB/s, done.
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/chela98/yova20230611.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

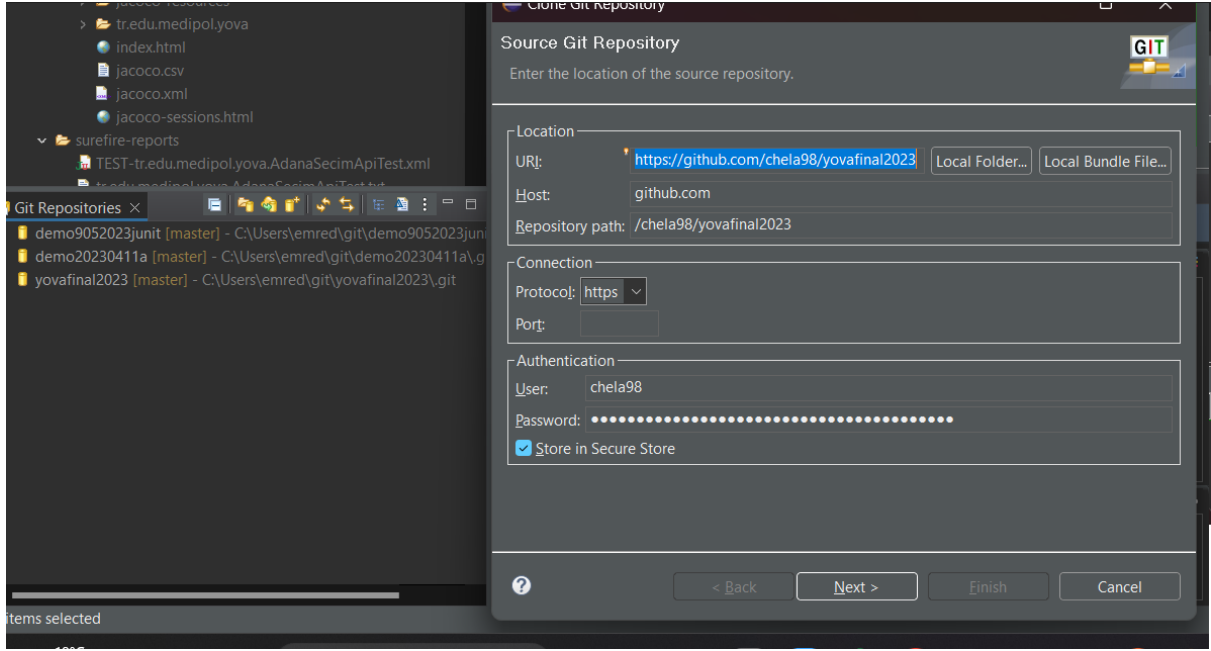
emred@emre MINGW64 ~/eclipse-workspace/AdanaSecim (master)
$ |
```

resim 4.2



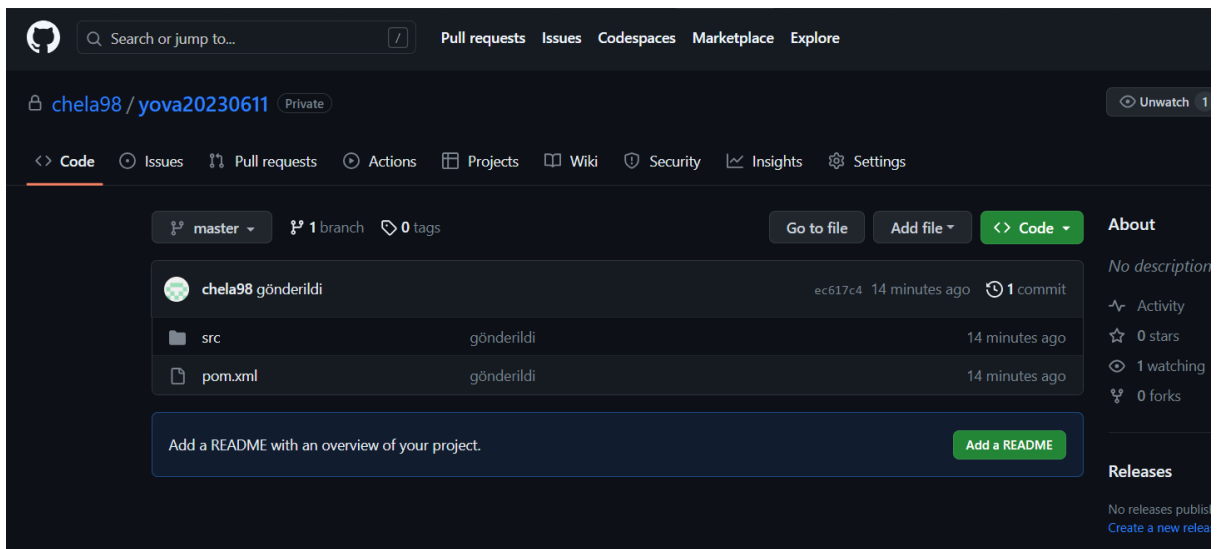
resim 4.3

Bu aşamadan sonra github actions kısmında hata alınması üzerine yovafinal2023 adında yeni bir repo oluşturup Eclipse üzerinden gönderim işlemi yapıldı.

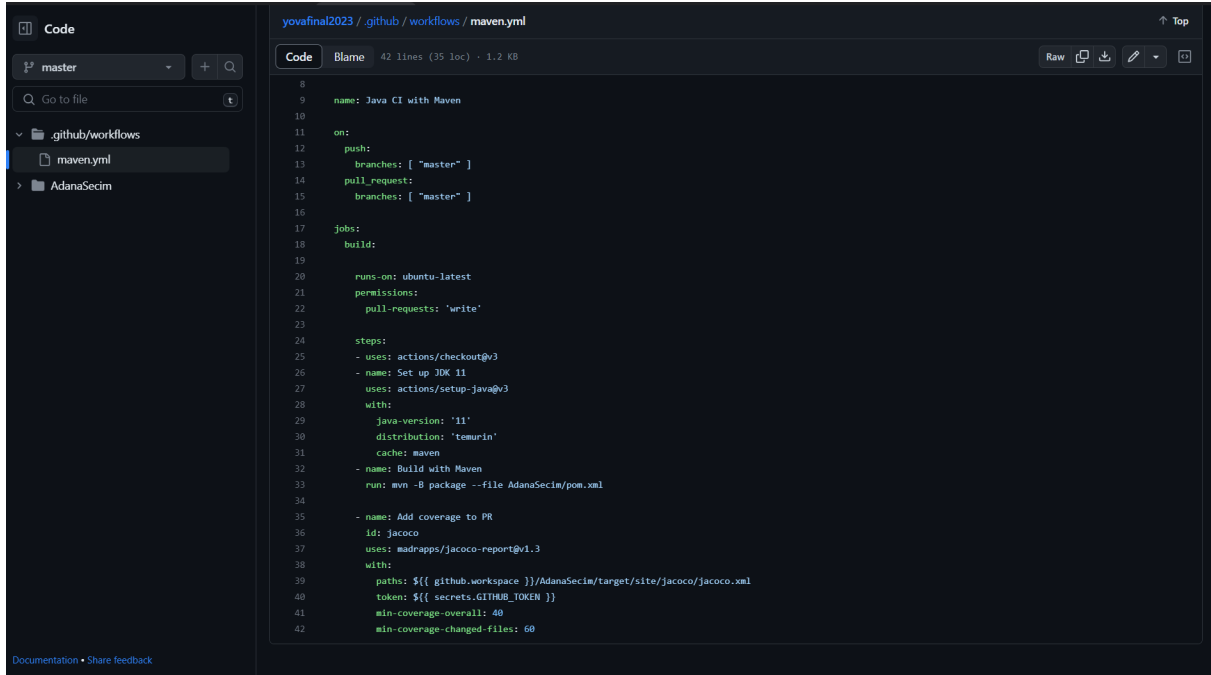


resim 5.1

Projenin GitHub deposuna gönderimden sonra otomatik olarak Java with Maven Actions ve PR (Pull Request) kapsama oranı hesaplamasını gerçekleştirmek için bir dizi işlemler gerçekleştirdik. Bu işlemlerin amacı yazılımın kalitesini ve güvenilirliğini artırmak, test kapsamını izlemek ve geliştirmek için üzere testleri otomatik yürütmektir. Bu bağlamda ilk etapta actions kısmından Java with Maven aracının config ayarlarını düzeltmek oldu.



resim 6.1



resim 6.2

```
- name: Build with Maven

run: mvn -B package --file AdanaSecim/pom.xml

- name: Add coverage to PR

id: jacoco

uses: madrapps/jacoco-report@v1.3

with:

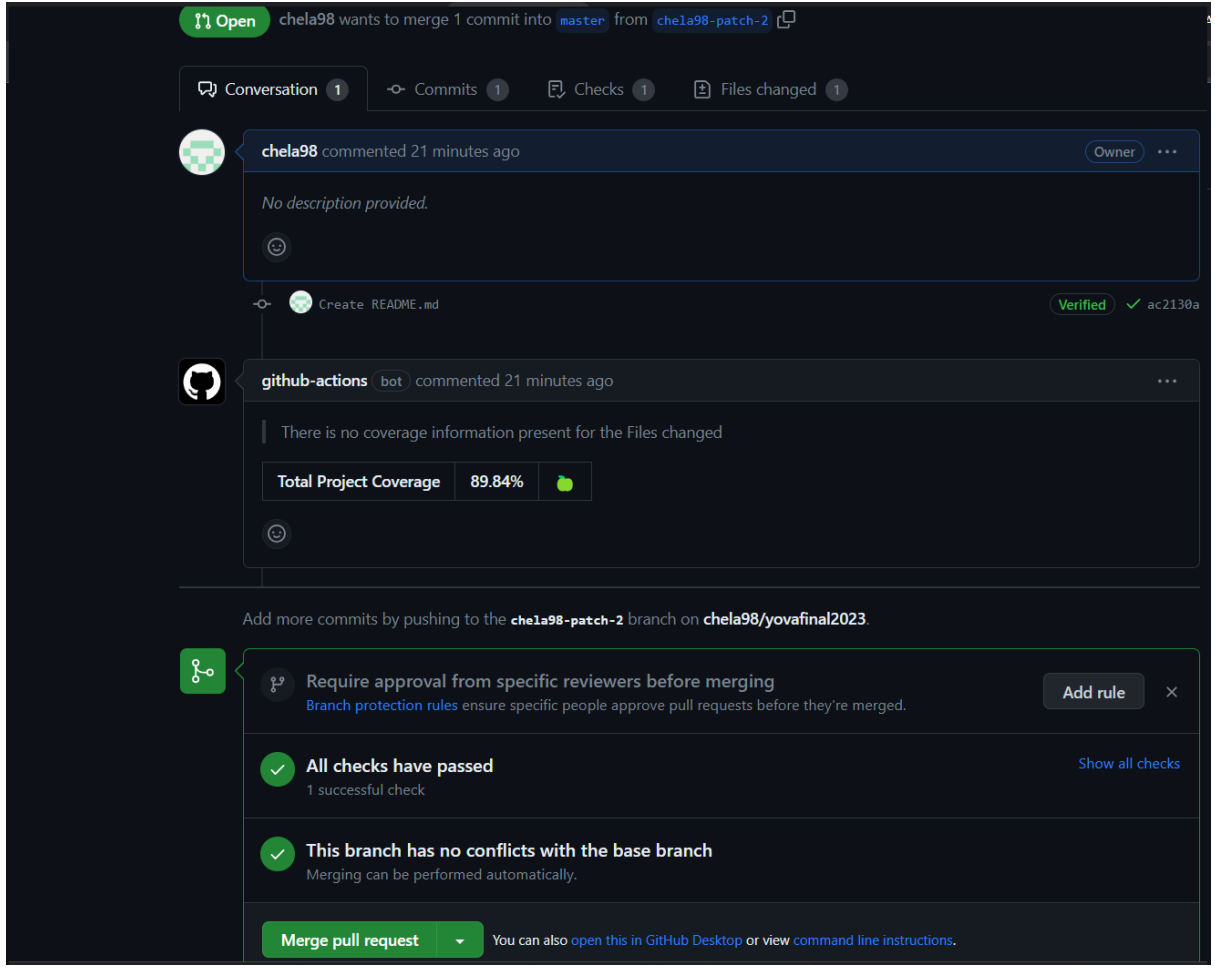
paths: ${{ github.workspace }}/AdanaSecim/target/site/jacoco/jacoco.xml

token: ${{ secrets.GITHUB_TOKEN }}

min-coverage-overall: 40

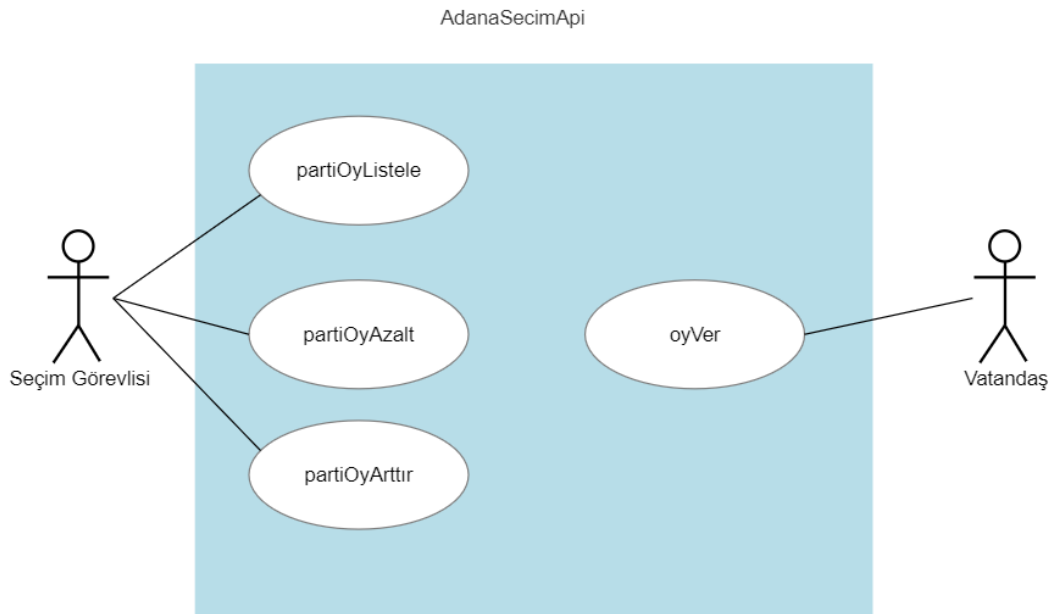
min-coverage-changed-files: 60
```

Gerekli ayarlamalar yapıldıktan sonra repoya README.md adında bir dosya oluşturuldu ve actions ve PR kapsama oranlarının doğru çalışıp çalışmadığı kontrol edildi.(resim 6.3)

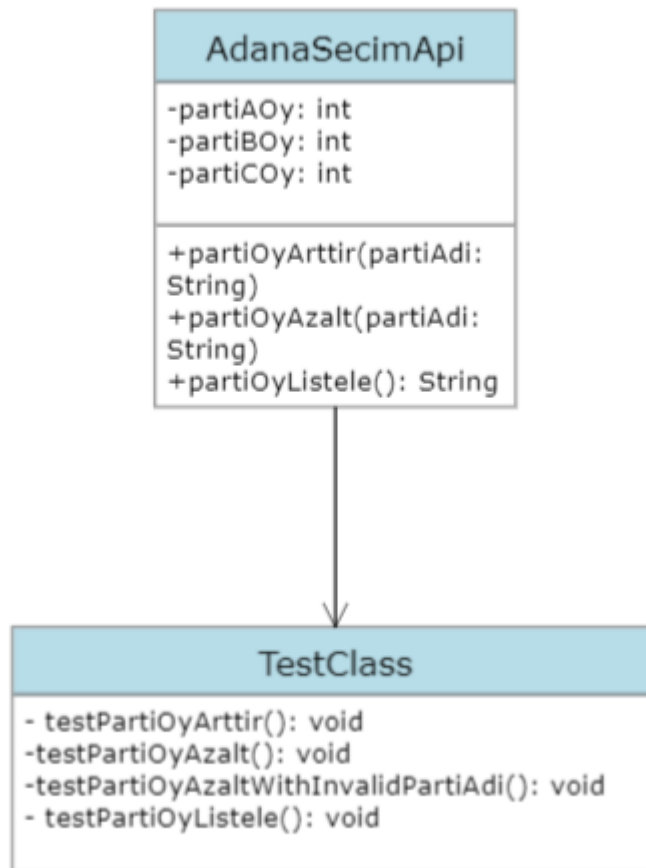


resim 6.3

USE CASE DİYAGRAM



UML SINIF DİYAGRAMI



<https://github.com/chela98/yovafinal2023>