UDACITY

Return to Classroom

UdaSecurity

REVIEW

HISTORY

Unable to review

Key parts of your submission were the same as another student's submission or an online source. Please resubmit after you address the issue noted below by the reviewer.

POTENTIAL SOURCE URL: https://github.com/Sanjay93/udasecurityparent | https://github.com/Eduguimar/Udasecurity

Good Morning 🌞

I've `checked` your `code` and I could see that you've `violated` the:

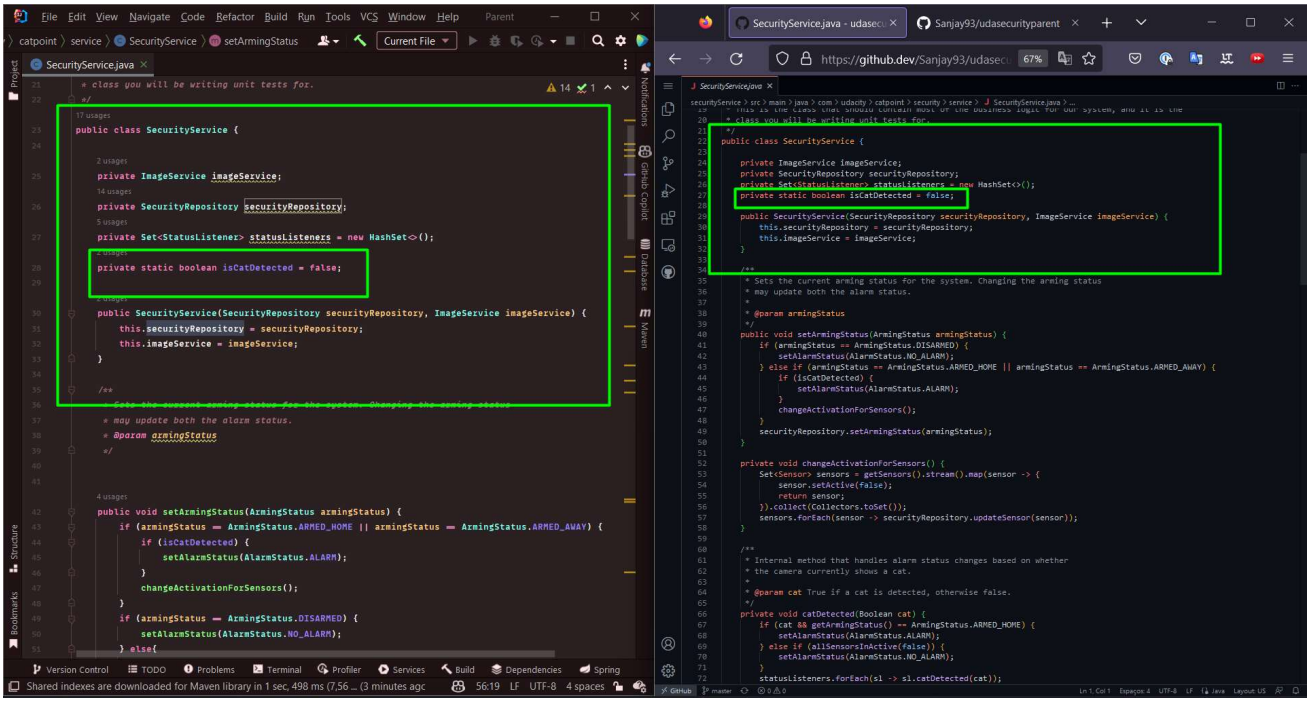
- Udacity's no-plagiarism honor code

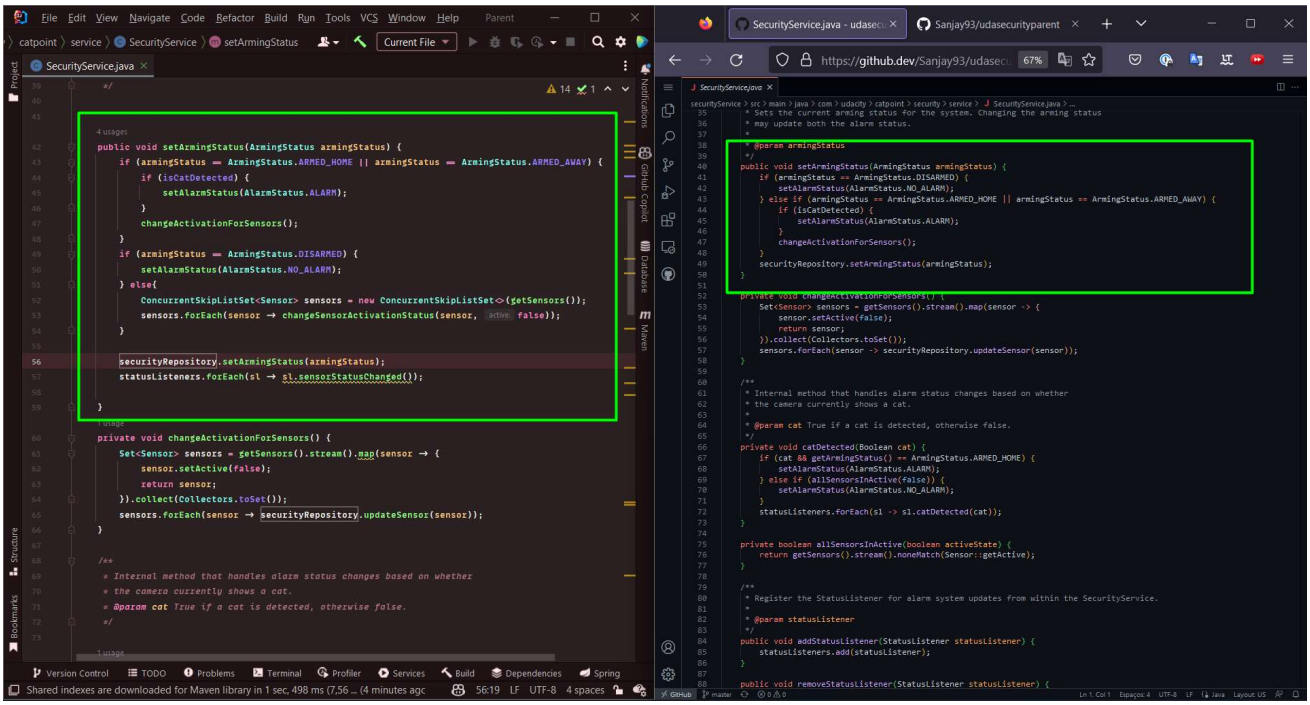
More specifically the items:

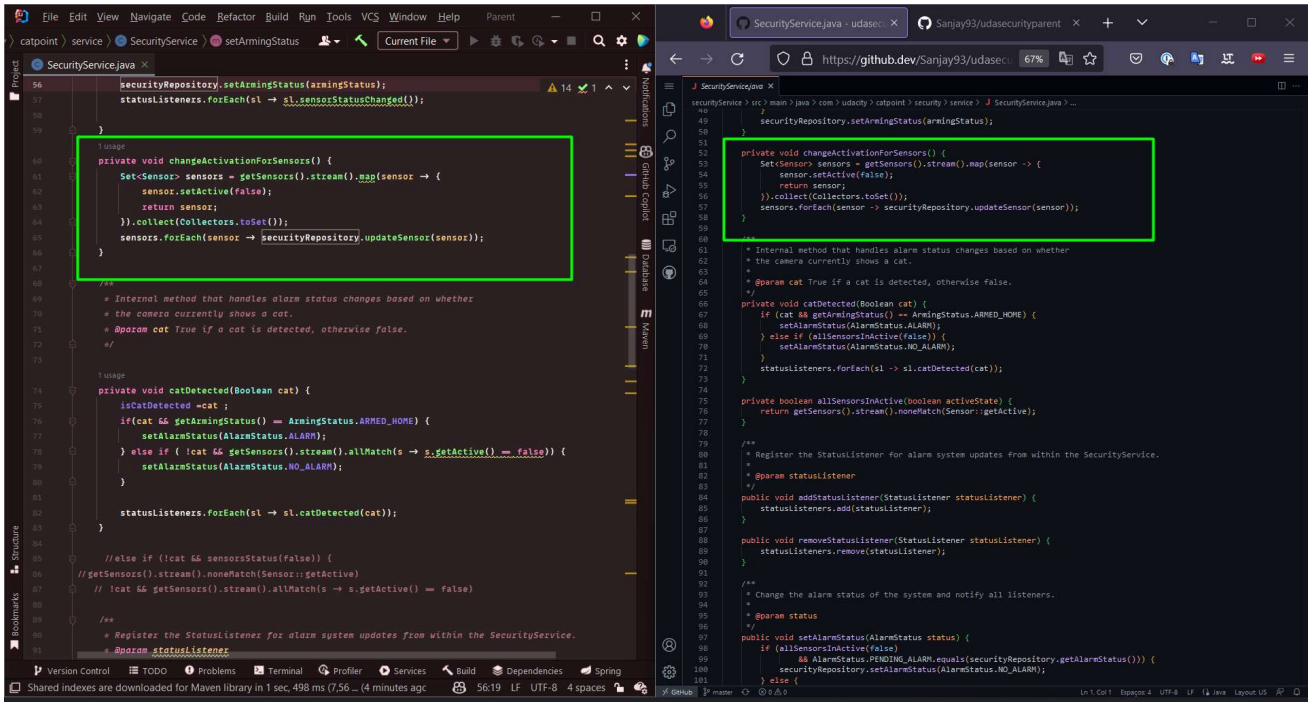
- Copying someone's code exactly, in whole or part (verbatim)
- Copying someone's code, in whole or part, and making changes (e.g. changing variable or function names, comments, order of function definition)
- Combining code copied verbatim from multiple sources.

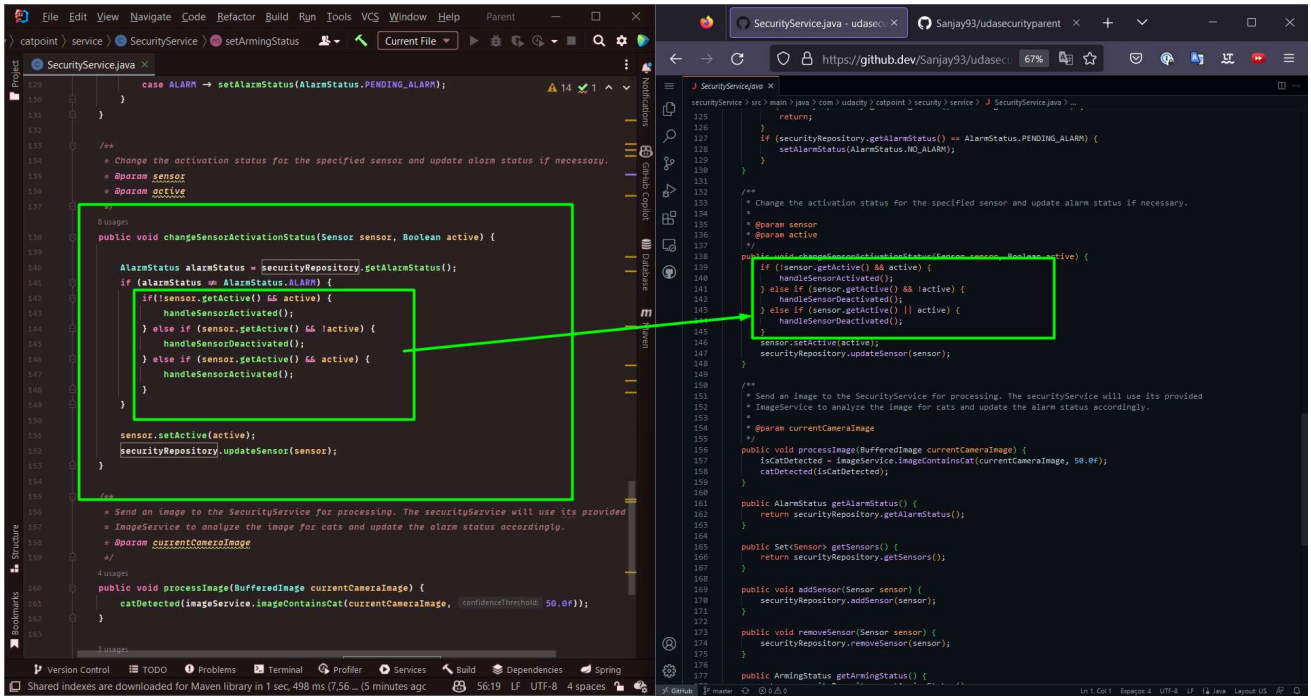
This can be seen in the `screenshots` right below:

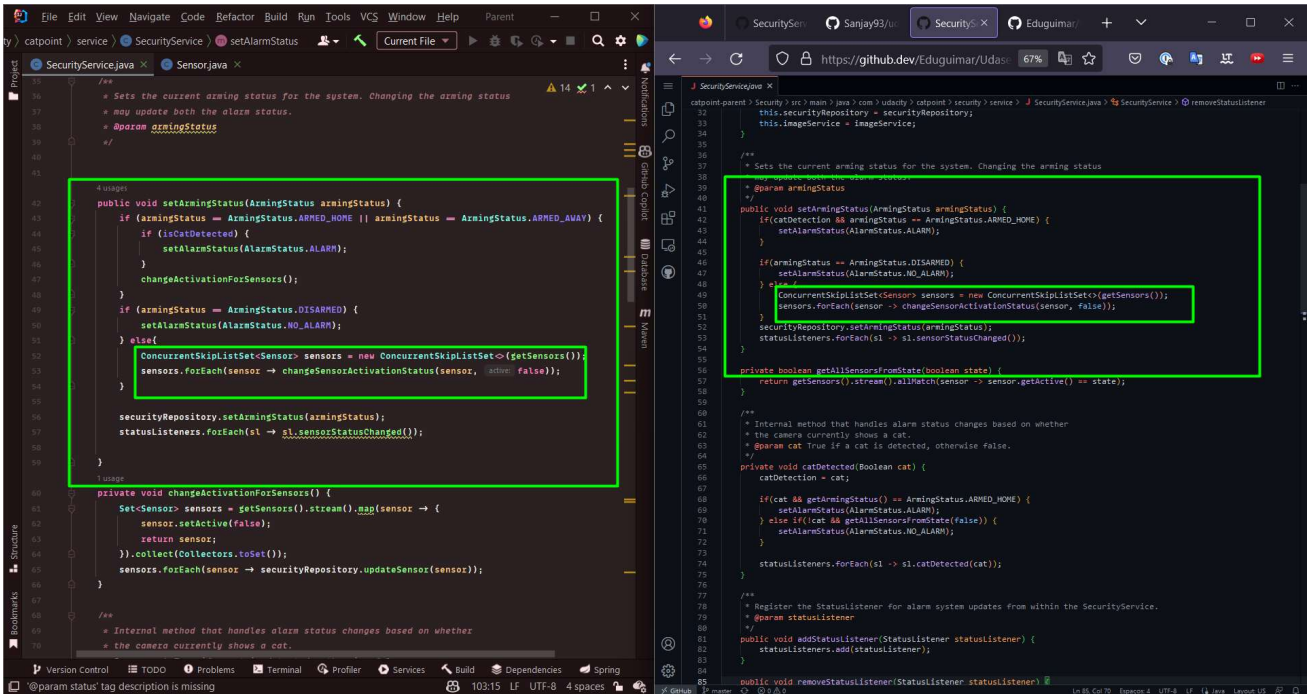
SCREENSHOTS - 🏞️



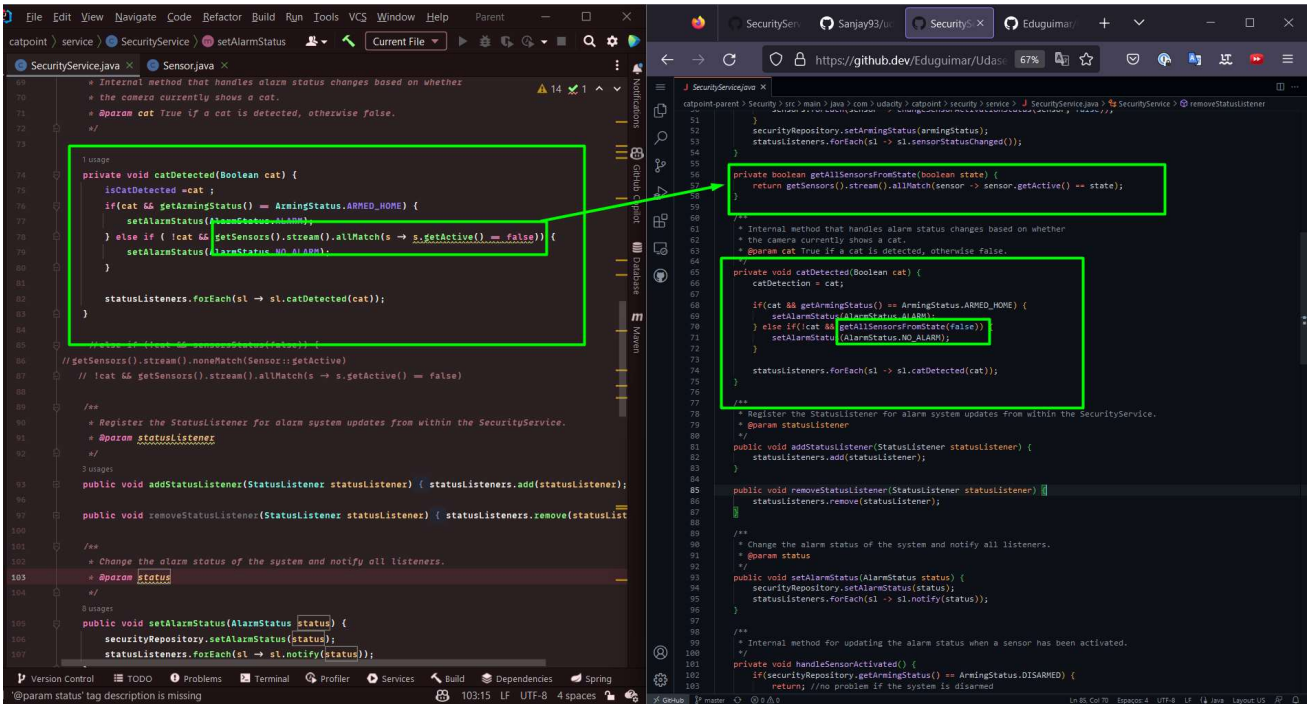




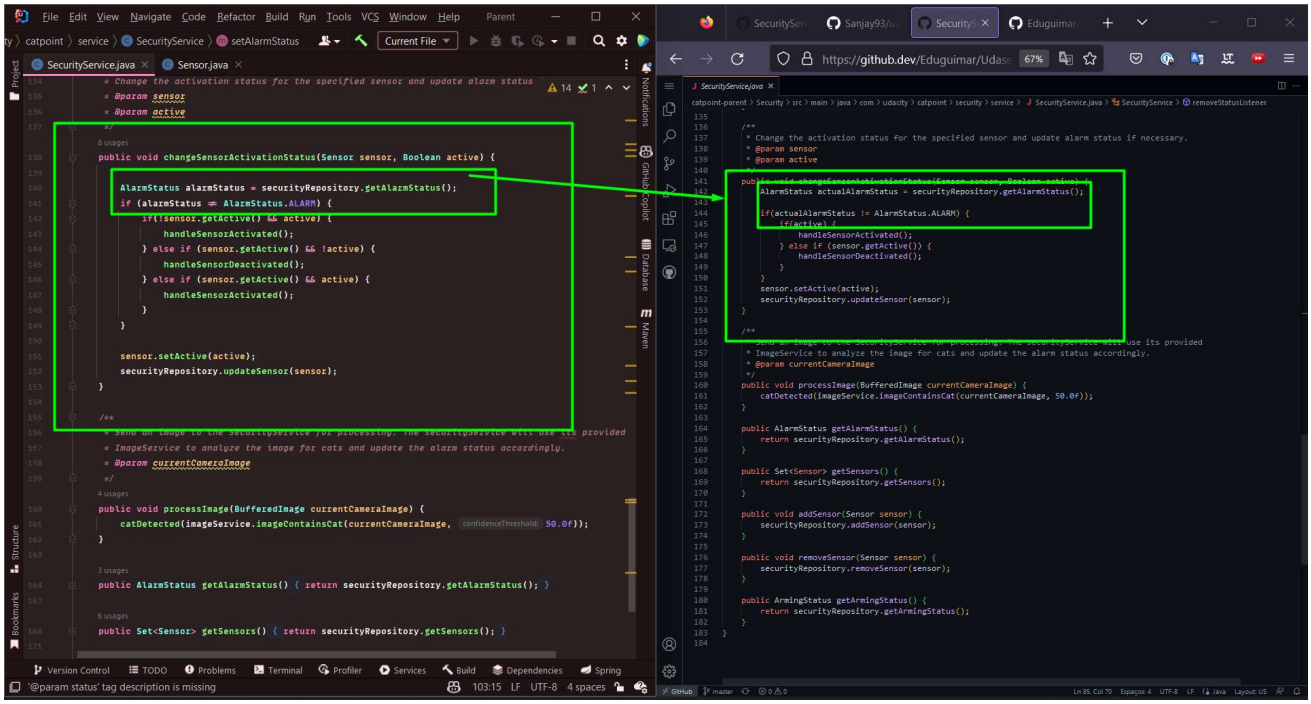




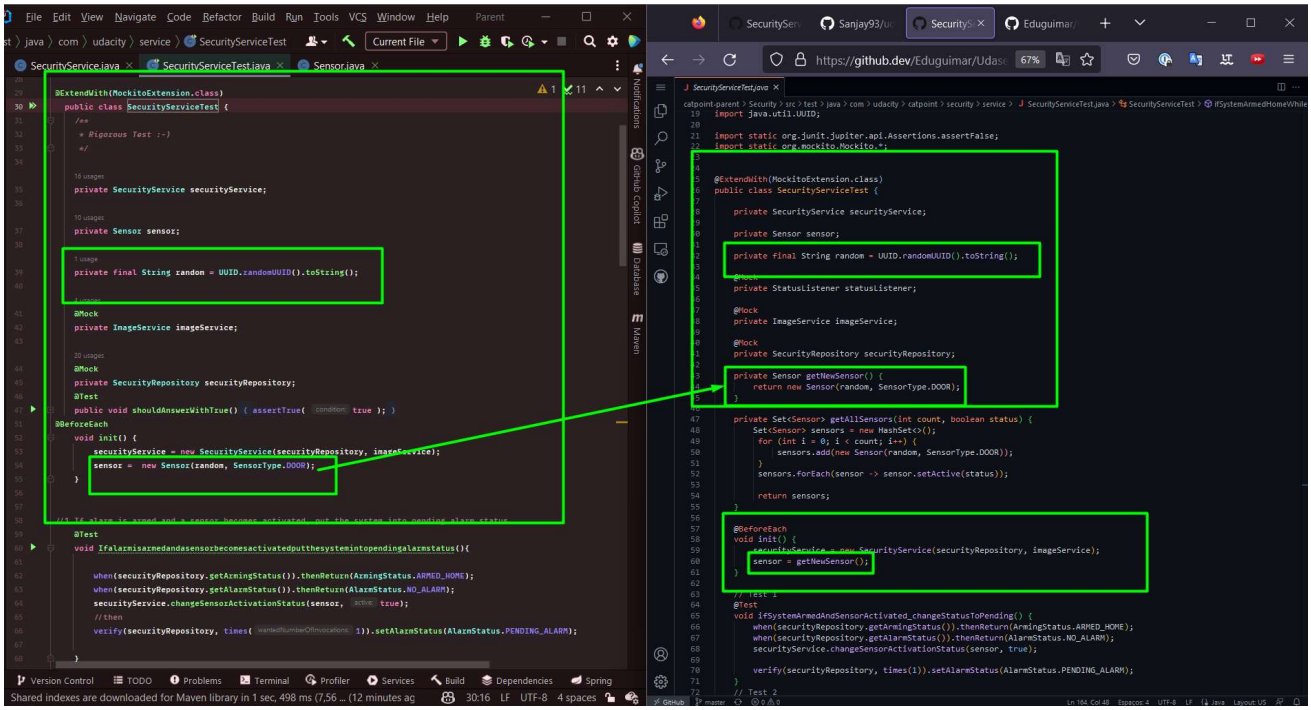
```
/**
 * Sets the current arming status for the system. Changing the arming status
 * may update both the alarm status.
 * @param armingStatus
 */

4 usages
public void setArmingStatus(ArmingStatus armingStatus) {
    if (armingStatus == ArmingStatus.ARMED_HOME || armingStatus == ArmingStatus.ARMED_AWAY) {
        if (isCatDetected) {
            setAlarmStatus(AlarmStatus.ALARM);
        }
        changeActivationForSensors();
    }
    if (armingStatus == ArmingStatus.DISARMED) {
        setAlarmStatus(AlarmStatus.NO_ALARM);
    } else{
        ConcurrentSkipListSet<Sensor> sensors = new ConcurrentSkipListSet<>(getSensors());
        sensors.forEach(sensor -> changeSensorActivationStatus(sensor, active: false));
    }

    securityRepository.setArmingStatus(armingStatus);
    statusListeners.forEach(sl -> sl.sensorStatusChanged());

}
1 usage
private void changeActivationForSensors() {
    Set<Sensor> sensors = getSensors().stream().map(sensor -> {
        sensor.setActive(false);
        return sensor;
    }).collect(Collectors.toSet());
    sensors.forEach(sensor -> securityRepository.updateSensor(sensor));
}

/**
 * Internal method that handles alarm status changes based on whether
 * the camera currently shows a cat.
```



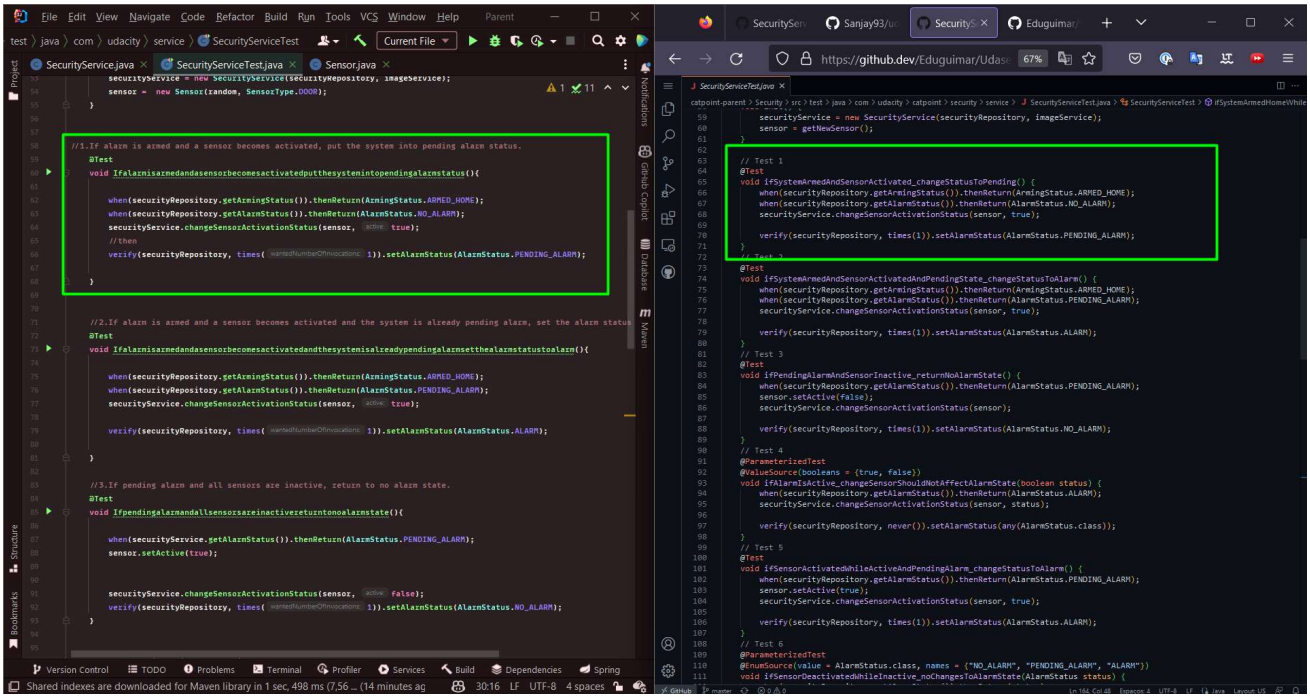
```
/**
 * Internal method that handles alarm status changes based on whether
 * the camera currently shows a cat.
 * @param cat True if a cat is detected, otherwise false.
 */

1 usage
private void catDetected(Boolean cat) {
    isCatDetected =cat ;
    if(cat && getArmingStatus() == ArmingStatus.ARMED_HOME) {
        setAlarmStatus(AlarmStatus.ALARM);
    } else if ( !cat && getSensors().stream().allMatch(s -> s.getActive() == false)) {
        setAlarmStatus(AlarmStatus.NO_ALARM);
    }

    statusListeners.forEach(sl -> sl.catDetected(cat));
}

// else if (!cat && sensorsStatus(false)) {
//getSensors().stream().noneMatch(Sensor::getActive)
// !cat && getSensors().stream().allMatch(s -> s.getActive() == false)

/**
 * Register the StatusListener for alarm system updates from within the SecurityService.
 * @param statusListener
 */
3 usages
public void addStatusListener(StatusListener statusListener) { statusListeners.add(statusListener);

public void removeStatusListener(StatusListener statusListener) { statusListeners.remove(statusList

/**
 * Change the alarm status of the system and notify all listeners.
 * @param status
 */
8 usages
public void setAlarmStatus(AlarmStatus status) {
    securityRepository.setAlarmStatus(status);
    statusListeners.forEach(sl -> sl.notify(status));
```

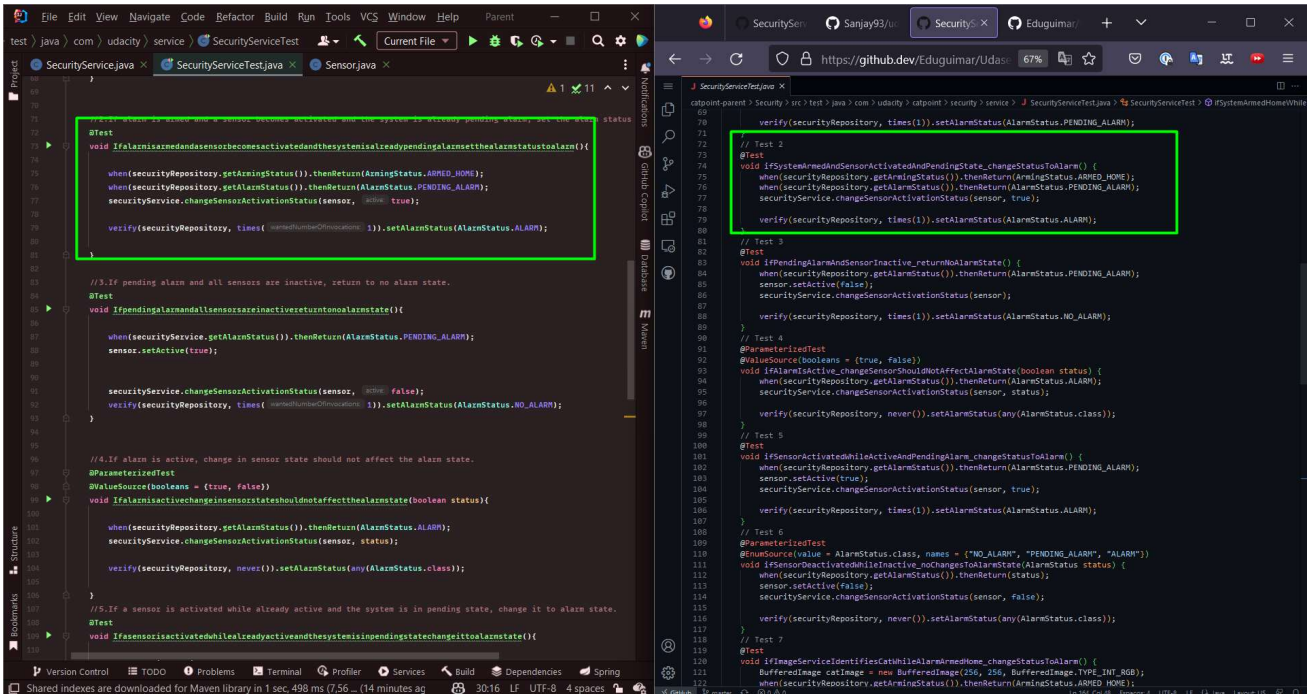


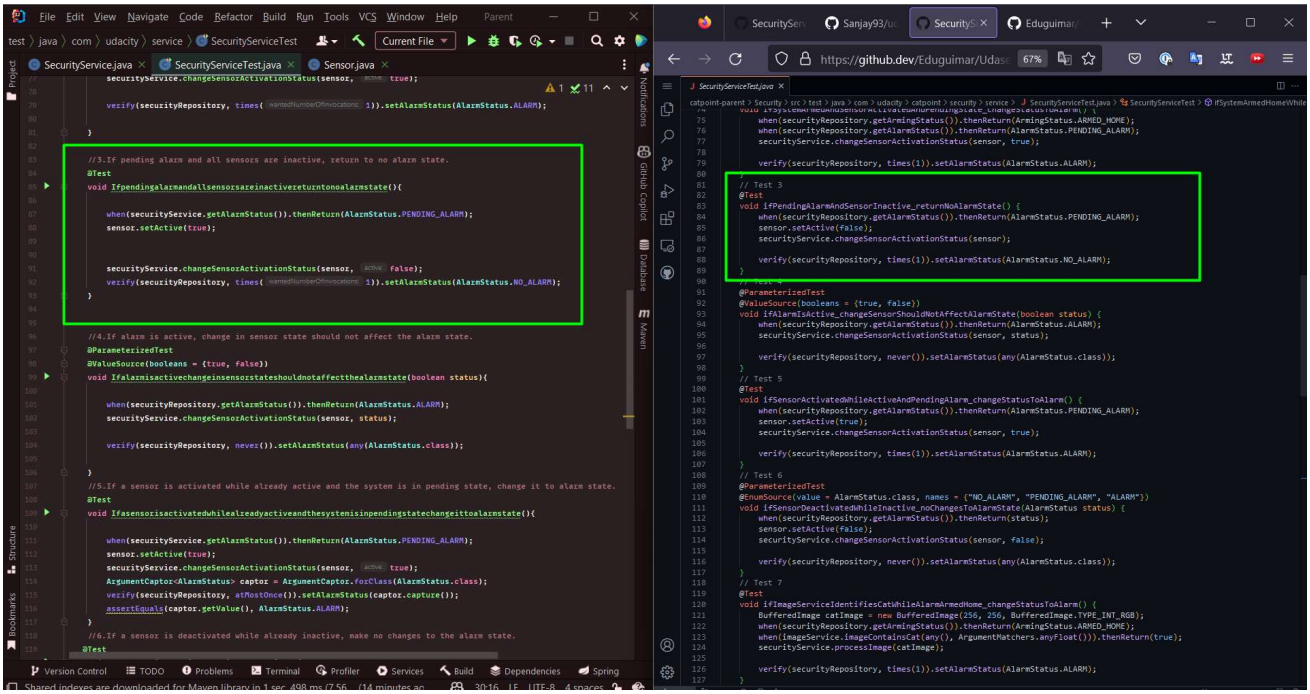
```java
/**
 * Change the activation status for the specified sensor and update alarm status if necessary.
 * @param sensor
 * @param active
 */
public void changeSensorActivationStatus(Sensor sensor, Boolean active) {
    AlarmStatus alarmStatus = securityRepository.getAlarmStatus();
    if (alarmStatus == AlarmStatus.ALARM) {
        if(!sensor.getActive() && active) {
            handleSensorActivated();
        } else if (sensor.getActive() && !active) {
            handleSensorDeactivated();
        } else if (sensor.getActive() && active) {
            handleSensorActivated();
        }
    }

    sensor.setActive(active);
    securityRepository.updateSensor(sensor);
}

/**
 * Send an image to the SecurityService for processing. The SecurityService will use its provided
 * ImageService to analyze the image for cats and update the alarm status accordingly.
 * @param currentCameraImage
 */
public void processImage(BufferedImage currentCameraImage) {
    catDetected(imageService.imageContainsCat(currentCameraImage, 50.0f));
}

public AlarmStatus getAlarmStatus() { return securityRepository.getAlarmStatus(); }

public Set<Sensor> getSensors() { return securityRepository.getSensors(); }
```

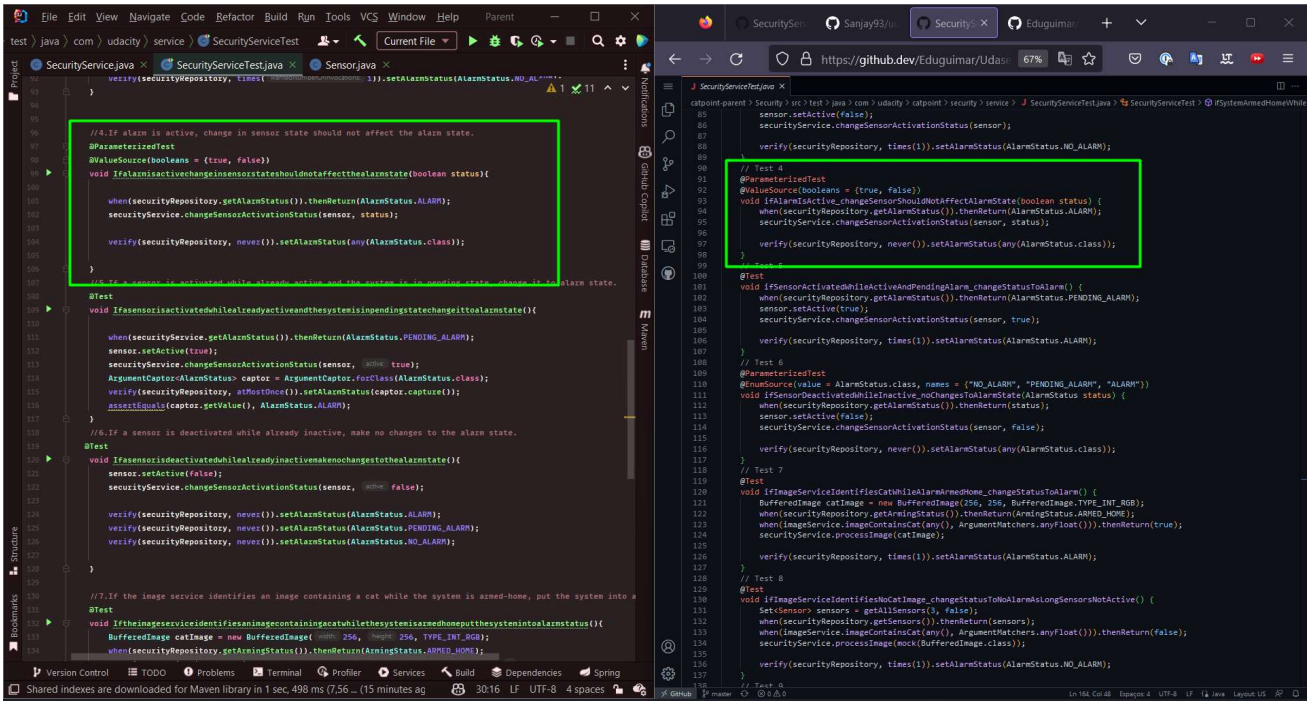


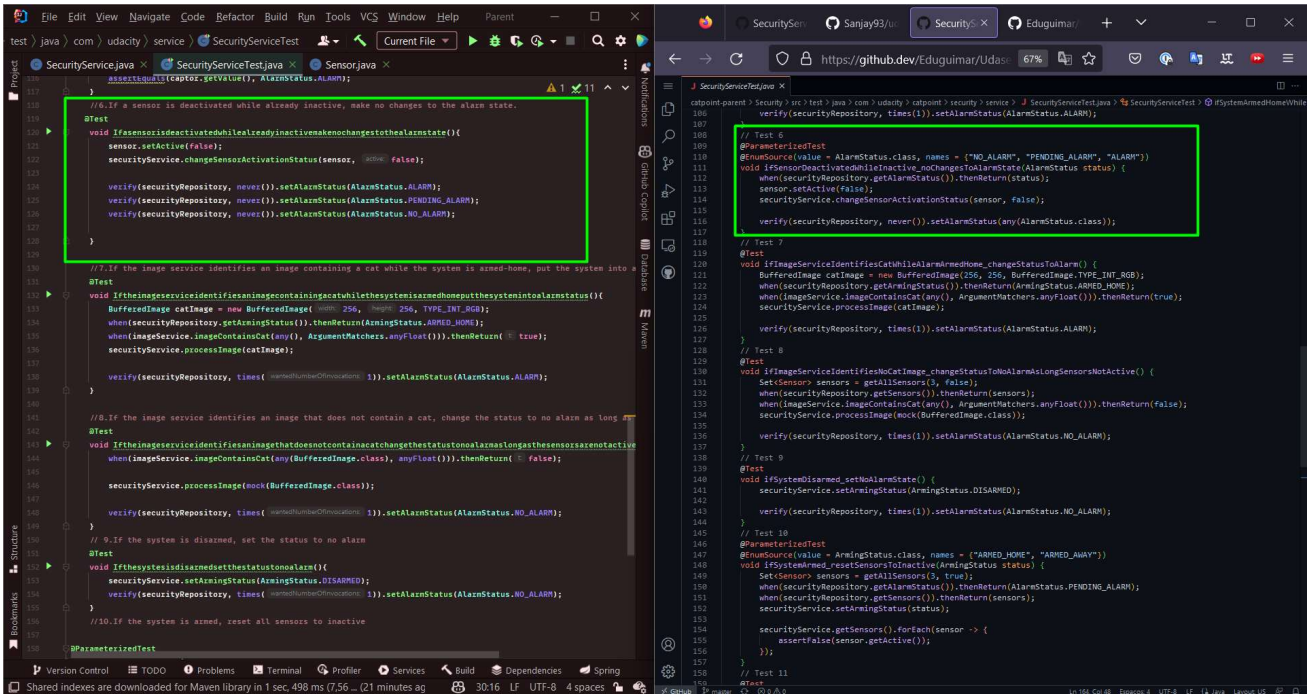
```java
@ExtendWith(MockitoExtension.class)
public class SecurityServiceTest {
    /**
     * Rigorous Test :-)
     */

    private SecurityService securityService;

    private Sensor sensor;

    private final String random = UUID.randomUUID().toString();

    @Mock
    private ImageService imageService;

    @Mock
    private SecurityRepository securityRepository;
    @Test
    public void shouldAnswerWithTrue() { assertTrue( true ); }
@BeforeEach
    void init() {
        securityService = new SecurityService(securityRepository, imageService);
        sensor = new Sensor(random, SensorType.DOOR);
    }

// If alarm is armed and a sensor becomes activated, put the system into pending alarm status.
    @Test
    void IfalarmisarmedandasensorbecomesactivatedputthesystemintopendingalarmstatusIf(){

        when(securityRepository.getArmingStatus()).thenReturn(ArmingStatus.ARMED_HOME);
        when(securityRepository.getAlarmStatus()).thenReturn(AlarmStatus.NO_ALARM);
        securityService.changeSensorActivationStatus(sensor, true);
        // then
        verify(securityRepository, times( 1)).setAlarmStatus(AlarmStatus.PENDING_ALARM);

    }
```

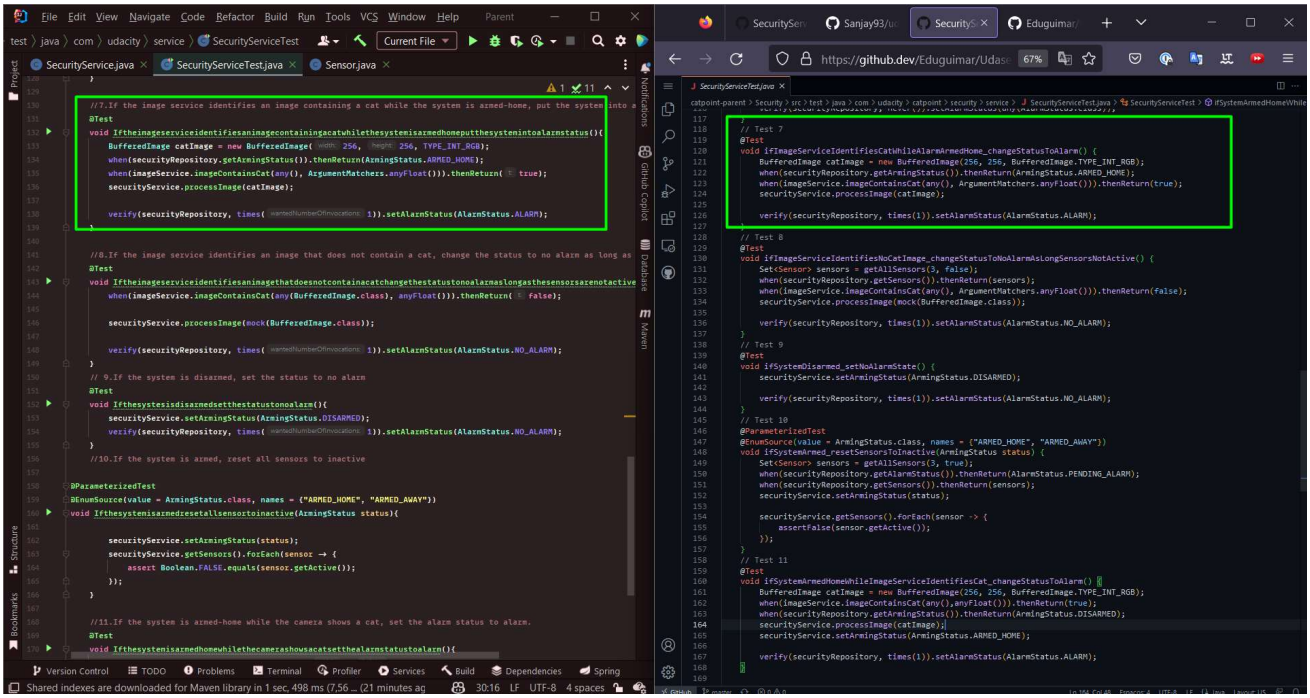


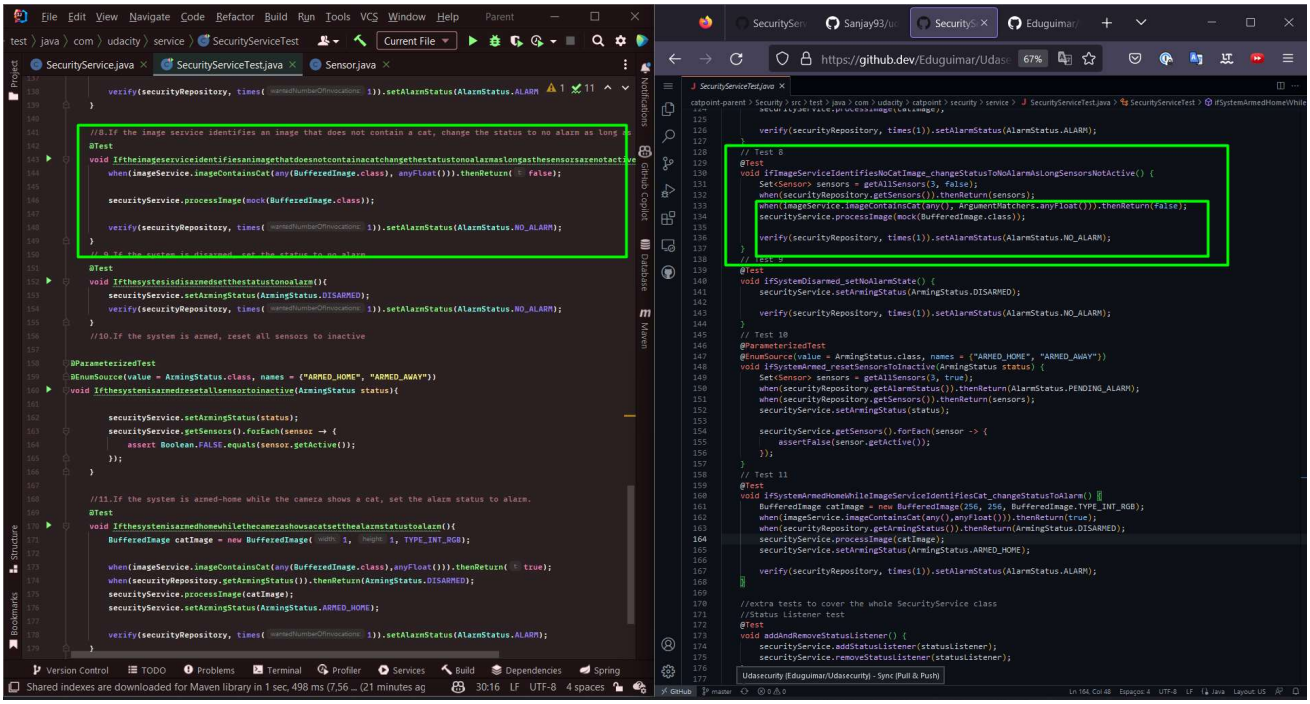


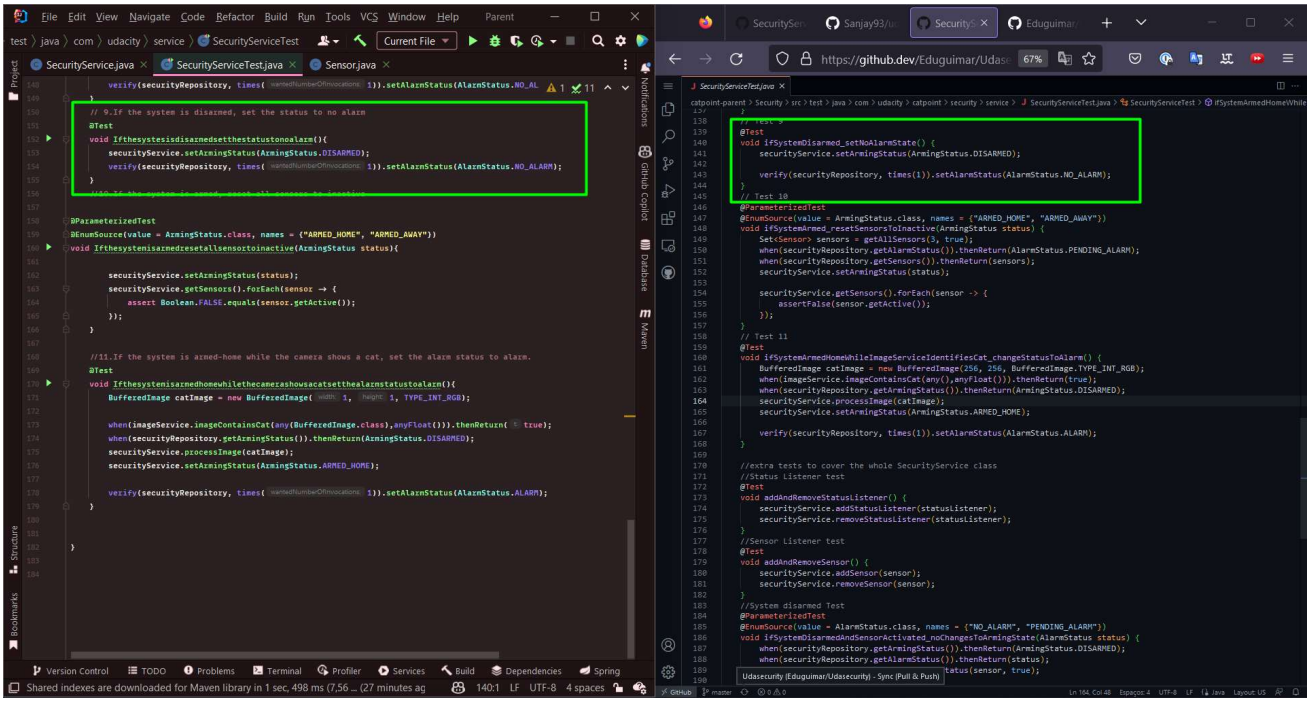


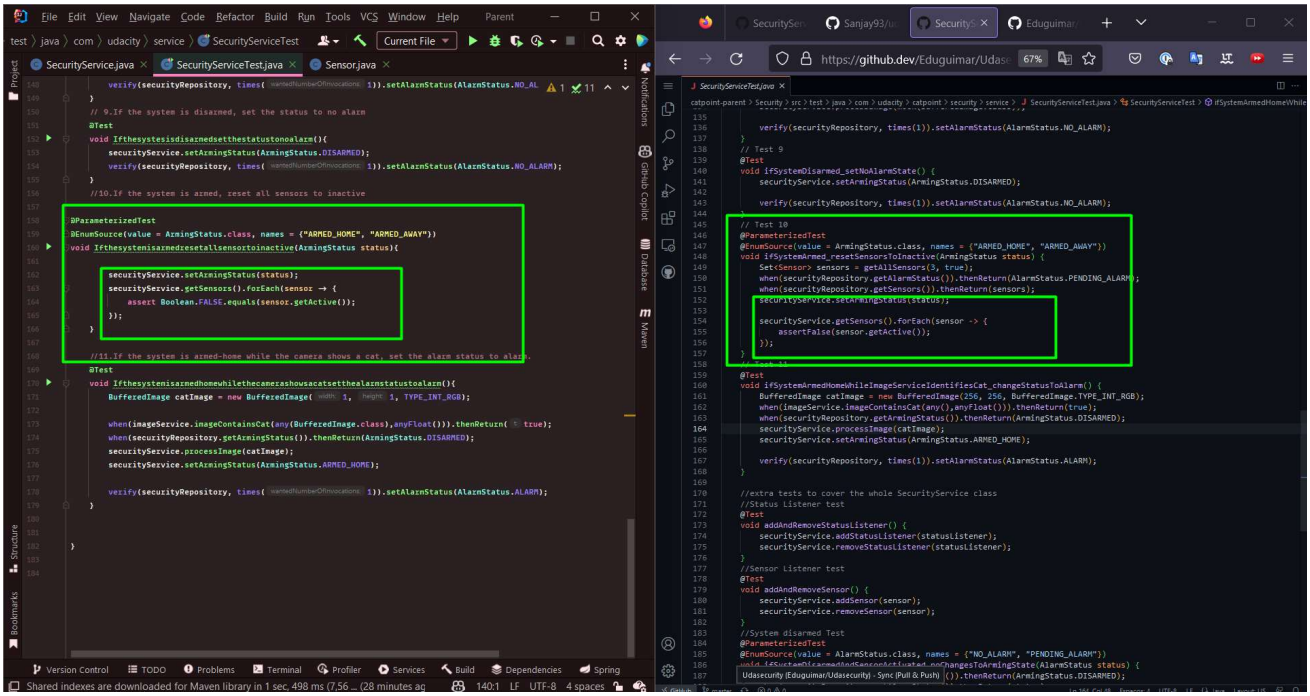


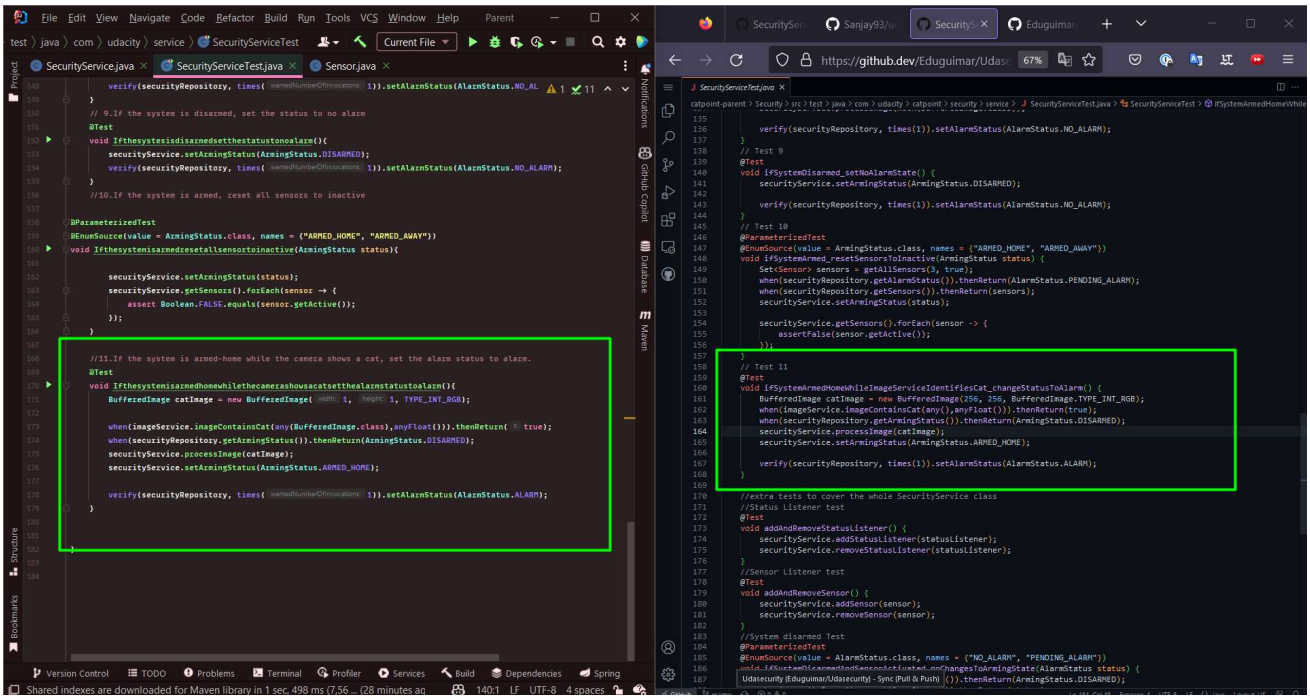












So, as can be seen above, you have basically plagiarized the `logic` and `code` of the classes:

- `SecurityService.java`
- `SecurityServiceTest.java`

From the repositories that were pointed to as sources.

Please, send an `original project`, so it can be `evaluated`

I wish you a great day

RESUBMIT

DOWNLOAD PROJECT FILES

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH