JAVA -Junit

Prompt

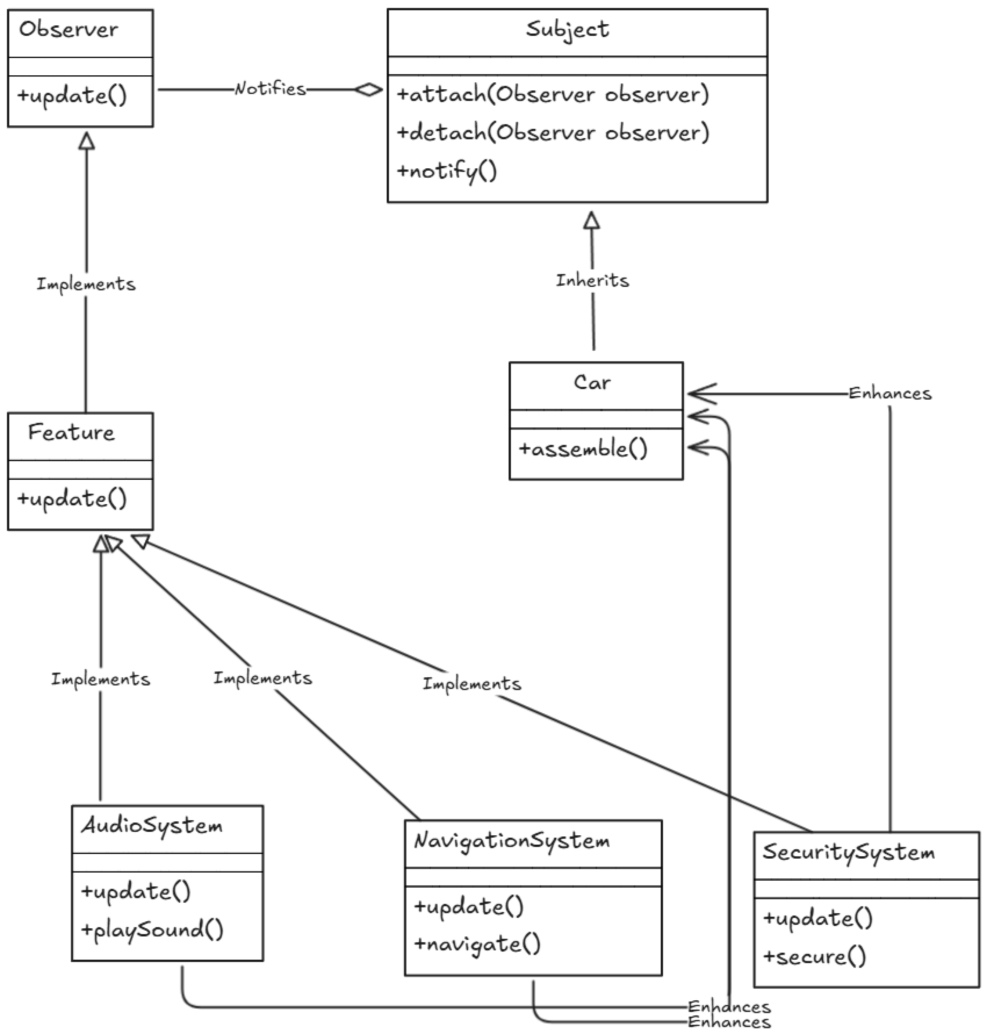
-Observer

Use the Observer pattern to write code in Java programming language and write JUnit to test the given code that have 100% statement coverage from the requirements below.

Use Observer pattern to design automotive system development by adding features to cars, such as audio system, navigation system or security systems without changing the main structure of the car.

Diagram

Observer



Observer

รอบที่1

AudioSystem

```
// AudioSystem.java
public class AudioSystem implements Observer {
    private String featureUpdate;

    @Override
    public void update(String featureUpdate) {
        this.featureUpdate = "Audio System updated with: " + featureUpdate;
        display();
    }

    public void display() {
        System.out.println(featureUpdate);
    }
}
```

AutomotiveSystemTest

```
// AutomotiveSystemTest.java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class AutomotiveSystemTest {

    @Test
    public void testObserverPattern() {
        Subject car = new Subject();

        AudioSystem audio = new AudioSystem();
        NavigationSystem navigation = new NavigationSystem();
        SecuritySystem security = new SecuritySystem();

        // Add observers
        car.addObserver(audio);
        car.addObserver(navigation);
        car.addObserver(security);

        // Test initial update
        car.setFeatureUpdate("Bluetooth connectivity");
        assertEquals("Audio System updated with: Bluetooth connectivity", audio.getFeatureUpdate());
        assertEquals("Navigation System updated with: Bluetooth connectivity",
navigation.getFeatureUpdate());
        assertEquals("Security System updated with: Bluetooth connectivity", security.getFeatureUpdate());

        // Remove an observer and update again
        car.removeObserver(security);
        car.setFeatureUpdate("GPS tracking");
        assertEquals("Audio System updated with: GPS tracking", audio.getFeatureUpdate());
        assertEquals("Navigation System updated with: GPS tracking", navigation.getFeatureUpdate());
        assertNull(security.getFeatureUpdate(), "Security System should not receive updates after being
removed");
    }
}
```

NavigationSystem

```
// NavigationSystem.java
public class NavigationSystem implements Observer {
    private String featureUpdate;

    @Override
    public void update(String featureUpdate) {
        this.featureUpdate = "Navigation System updated with: " + featureUpdate;
        display();
    }

    public void display() {
        System.out.println(featureUpdate);
    }
}
```

Observer

```
// Observer.java
public interface Observer {
    void update(String featureUpdate);
}
```

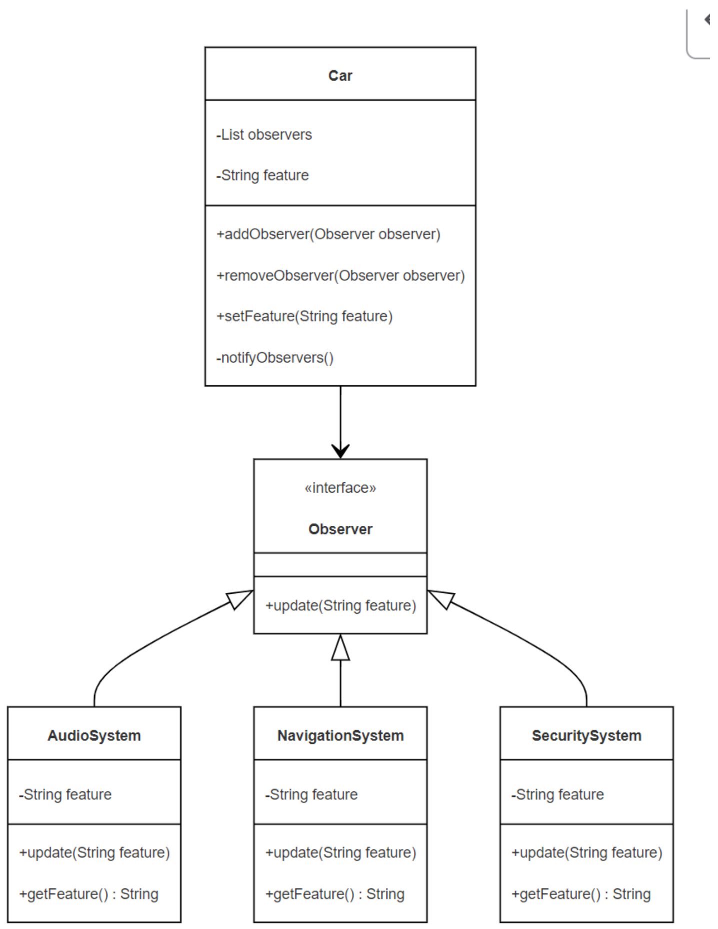
SecuritySystem

```
// SecuritySystem.java
public class SecuritySystem implements Observer {
    private String featureUpdate;

    @Override
    public void update(String featureUpdate) {
        this.featureUpdate = "Security System updated with: " + featureUpdate;
        display();
    }

    public void display() {
        System.out.println(featureUpdate);
    }
}
```

Subject

```
// Subject.java
import java.util.ArrayList;
import java.util.List;

public class Subject {
    private List<Observer> observers = new ArrayList<>();
    private String featureUpdate;

    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void removeObserver(Observer observer) {
        observers.remove(observer);
    }

    public void notifyObservers() {
        for (Observer observer : observers) {
            observer.update(featureUpdate);
        }
    }

    public void setFeatureUpdate(String featureUpdate) {
        this.featureUpdate = featureUpdate;
        notifyObservers();
    }
}
```

Diagram From Code:



โค้ดที่แก้ไข:

```
AudioSystem.java (add)
public String getFeatureUpdate() {
        return this.featureUpdate;
    }

NavigationSystem.java (add)
public String getFeatureUpdate() {
        return this.featureUpdate;
    }

SecuritySystem.java (add)
public String getFeatureUpdate() {
        return this.featureUpdate;
    }

subject.java (add final in front of List)
private List<Observer> observers = new ArrayList<>(); --> private final List<Observer> observers = new
ArrayList<>();
```