JAVA -Junit

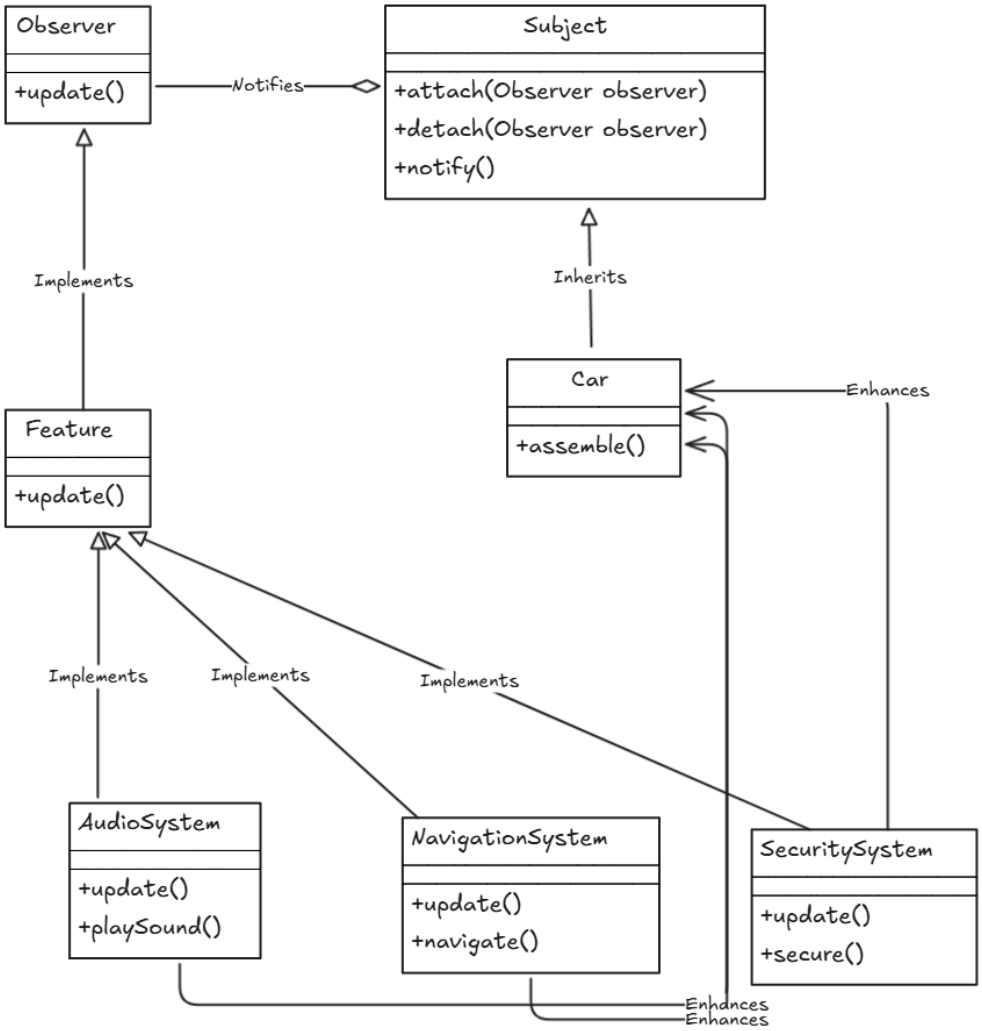
Prompt

-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

รอบที่3

AudioSystem

```
public class AudioSystem implements Observer {
    private String feature;

    @Override
    public void update(String feature) {
        this.feature = "Audio System: " + feature;
        System.out.println(this.feature);
    }

    public String getFeature() {
        return feature;
    }
}
```

AutomotiveSystemTest

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

public class CarTest {

    @Test
    public void testAddingFeatures() {
        Car car = new Car();

        AudioSystem audioSystem = new AudioSystem();
        NavigationSystem navigationSystem = new NavigationSystem();
        SecuritySystem securitySystem = new SecuritySystem();
```

```
    car.addObserver(audioSystem);
    car.addObserver(navigationSystem);
    car.addObserver(securitySystem);

    car.setFeature("Bluetooth enabled");

    assertEquals("Audio System: Bluetooth enabled", audioSystem.getFeature());
    assertEquals("Navigation System: Bluetooth enabled", navigationSystem.getFeature());
    assertEquals("Security System: Bluetooth enabled", securitySystem.getFeature());
}

@Test
public void testRemovingFeature() {
    Car car = new Car();

    AudioSystem audioSystem = new AudioSystem();
    NavigationSystem navigationSystem = new NavigationSystem();
    SecuritySystem securitySystem = new SecuritySystem();

    car.addObserver(audioSystem);
    car.addObserver(navigationSystem);
    car.addObserver(securitySystem);

    car.setFeature("GPS tracking enabled");

    assertEquals("Audio System: GPS tracking enabled", audioSystem.getFeature());
    assertEquals("Navigation System: GPS tracking enabled", navigationSystem.getFeature());
    assertEquals("Security System: GPS tracking enabled", securitySystem.getFeature());

    // Remove NavigationSystem and update again
    car.removeObserver(navigationSystem);
    car.setFeature("Anti-theft system enabled");

    assertEquals("Audio System: Anti-theft system enabled", audioSystem.getFeature());
```

```
        assertEquals("Security System: Anti-theft system enabled", securitySystem.getFeature());
        assertEquals("Navigation System: GPS tracking enabled", navigationSystem.getFeature());  // No update
after removal
    }
}
```

NavigationSystem

```
public class NavigationSystem implements Observer {
    private String feature;

    @Override
    public void update(String feature) {
        this.feature = "Navigation System: " + feature;
        System.out.println(this.feature);
    }

    public String getFeature() {
        return feature;
    }
}
```

Observer

```
public interface Observer {
    void update(String feature);
}
```

SecuritySystem

```
public class SecuritySystem implements Observer {
```

```
    private String feature;

    @Override
    public void update(String feature) {
        this.feature = "Security System: " + feature;
        System.out.println(this.feature);
    }

    public String getFeature() {
        return feature;
    }
}
```

Subject

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

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

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

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

    public void setFeature(String feature) {
        this.feature = feature;
        notifyObservers();
```

```
    }

    private void notifyObservers() {
        for (Observer observer : observers) {
            observer.update(feature);
        }
    }
}
```

Diagram From Code: