JAVA -Junit

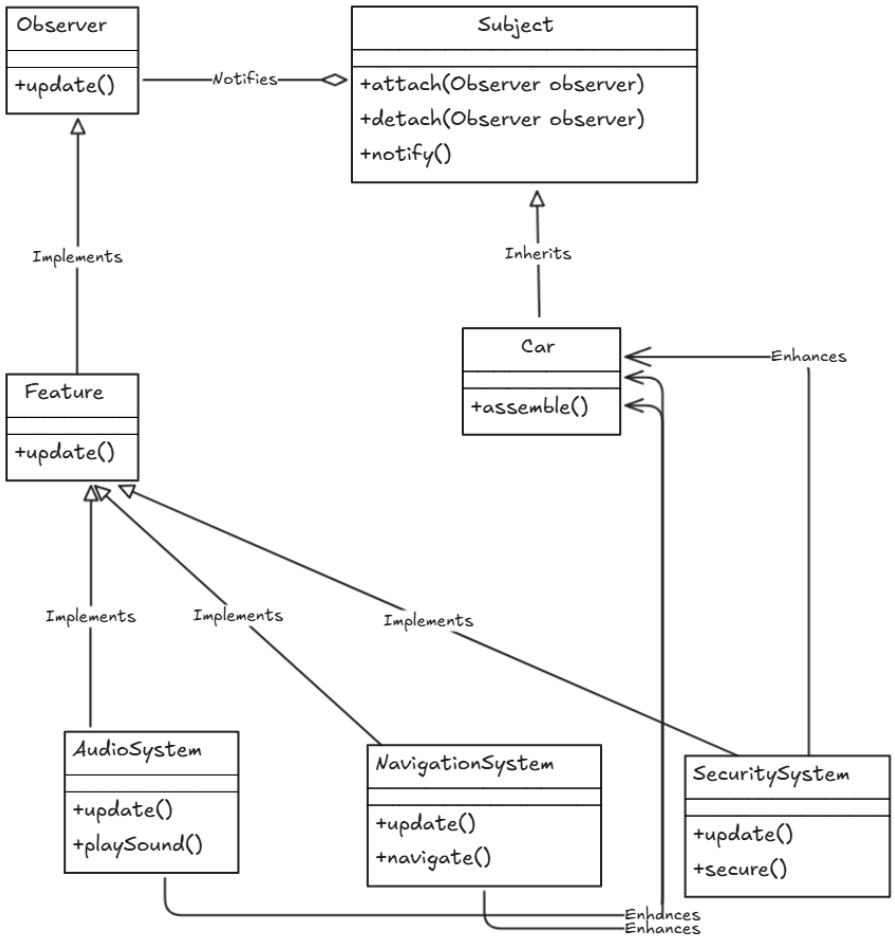
Prompt

-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

รอบที่2

AudioSystem

```
public class AudioSystem implements CarFeature {
    @Override
    public void update() {
        System.out.println("Audio System updated.");
    }
}
```

Car

```
import java.util.List;

public interface Car {
    void attach(CarFeature feature);
    void detach(CarFeature feature);
    void notifyFeatures();
}
```

CarFeature

```
public interface CarFeature {
    void update();
}
```

ConcreateCar

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

```
import java.util.List;

public class ConcreteCar implements Car {
    private List<CarFeature> features = new ArrayList<>();

    @Override
    public void attach(CarFeature feature) {
        features.add(feature);
    }

    @Override
    public void detach(CarFeature feature) {
        features.remove(feature);
    }

    @Override
    public void notifyFeatures() {
        for (CarFeature feature : features) {
            feature.update();
        }
    }
}
```

ConcreteCarTest

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

public class ConcreteCarTest {
    @Test
    public void testAttachAndNotify() {
        ConcreteCar car = new ConcreteCar();
        CarFeature audio = new AudioSystem();
```

```
        CarFeature navigation = new NavigationSystem();
        CarFeature security = new SecuritySystem();

        car.attach(audio);
        car.attach(navigation);
        car.attach(security);

        car.notifyFeatures();
    }

    @Test
    public void testDetach() {
        ConcreteCar car = new ConcreteCar();
        CarFeature audio = new AudioSystem();
        CarFeature navigation = new NavigationSystem();
        CarFeature security = new SecuritySystem();

        car.attach(audio);
        car.attach(navigation);
        car.attach(security);

        car.detach(navigation);

        car.notifyFeatures();
    }
}
```

NavigationSystem

```
public class NavigationSystem implements CarFeature {
    @Override
    public void update() {
        System.out.println("Navigation System updated.");
```

```
    }
}
```

SecuritySystem

```
public class SecuritySystem implements CarFeature {
    @Override
    public void update() {
        System.out.println("Security System updated.");
    }
}
```

Diagram From Code: