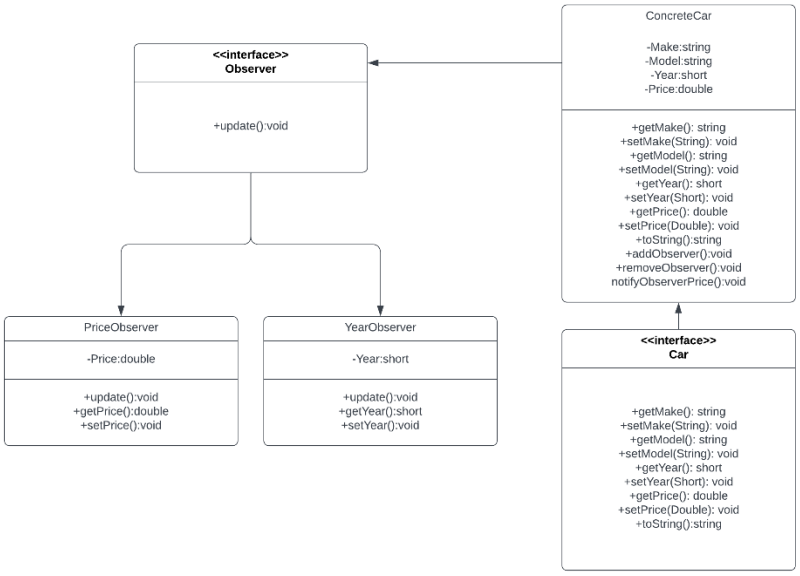
Anthony Ferrara

Observer Pattern

Team 4: Used Car Dealership

UML Diagram



Code

Car.java

```
package main;
public interface Car
{
    public String getMake();
    public void setMake(String make);
    public String getModel();
    public void setModel(String model);
    public short getYear();
    public void setYear(short year);
    public double getPrice();
    public void setPrice(double price);
    public String toString();
}
```

ConcreteCar.java

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

public class ConcreteCar implements Car
{
    private String Make;
    private String Model;
    private short Year;
    private double Price;
    private List<Observer> observers = new ArrayList<>();


    public ConcreteCar(String make, String model, short year, double price)
    {
        Make = make;
        Model = model;
        Year = year;
        Price = price;
    }

    @Override
    public String getMake()
    {
        return this.Make;
    }

    @Override
    public void setMake(String make)
    {
        this.Make = make;

    }

    @Override
    public String getModel()
    {
        return this.Model;
    }

    @Override
    public void setModel(String model)
    {
        this.Model = model;
    }

    @Override
    public short getYear()
    {
        return this.Year;
    }

    @Override
    public void setYear(short year)
    {
```

```
        this.Year = year;
    }

    @Override
    public double getPrice()
    {
        return this.Price;
    }

    @Override
    public void setPrice(double price)
    {
        this.Price = price;
    }

    public String toString()
    {
        return "Make: "+Make+"\nModel: "+Model+"\nYear: "+Year+"\nPrice:
"+Price;
    }
    public void addObserver(Observer observer) {
        this.observers.add(observer);
    }

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

    public void notifyObserversPrice(double price) {
        this.Price = price;
        for (Observer observer : this.observers) {
            observer.update(this.Price);
        }
    }
    public void notifyObserversYear(short year) {
        this.Year = year;
        for (Observer observer : this.observers) {
             observer.update(this.Year);
            }
        }
    }
```

Observer.java

```
package main;

public interface Observer {
    public void update(Object o);
}
```

PriceObserver.java

```
package main;

public class PriceObserver implements Observer{
    private double Price;
    @Override
    public void update(Object Price) {
        this.setPrice((Double) Price);
```

```
    }
    public double getPrice() {
        return this.Price;
    }
    public void setPrice(double price)
    {
        this.Price = price;
    }
}
```

YearObserver.java

```
package main;

public class YearObserver implements Observer{
    private short Year;
    @Override
    public void update(Object Year) {
        this.setYear((short) Year);
    }
    public double getYear() {
        return this.Year;
    }
    public void setYear(short year)
    {
        this.Year = year;
    }
}
```

Unit Tests

CarTests.java

```
import main.*;
import org.junit.jupiter.api.*;

public class CarTests {
    @Test
    public void doesReturnPrice(){
        ConcreteCar subject = new ConcreteCar("Ford", "Focus", (short) 2020,
25000);
        PriceObserver observer = new PriceObserver();
        subject.addObserver(observer);
        subject.notifyObserversPrice(23000);
        Assertions.assertEquals(23000, observer.getPrice());
    }
    @Test
    public void doesReturnYear(){
        ConcreteCar subject = new ConcreteCar("Ford", "Focus", (short) 2020,
25000);
        YearObserver observer = new YearObserver();
        subject.addObserver(observer);
        subject.notifyObserversYear((short) 2021);
        Assertions.assertEquals(2021, observer.getYear());

    }
}
```

Component Test

Create two observer classes, one that observes the car's year of production, and one that observes the car's price. When these values change, they are passed to the observers via the ConcreteCar class. Add an instance of either observer to the array list located within the ConcreteCar class. Then, use the getter methods located within each observer class, (priceObserver and yearObserver) to retrieve the information.