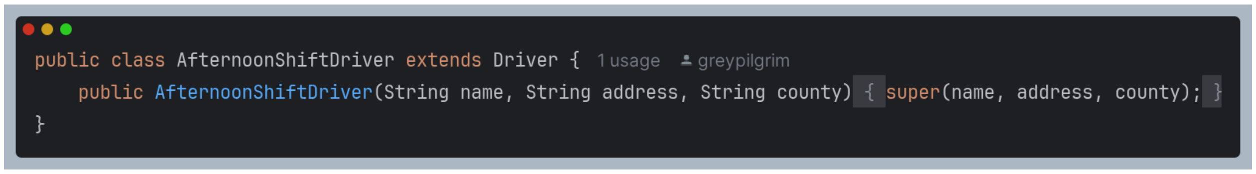
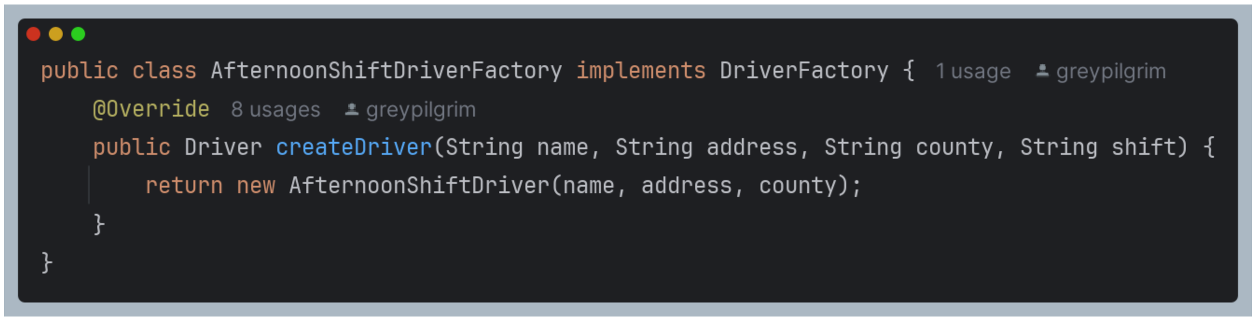
Repo: https://github.com/greypilgrim/CS_4800_FinalProject.git

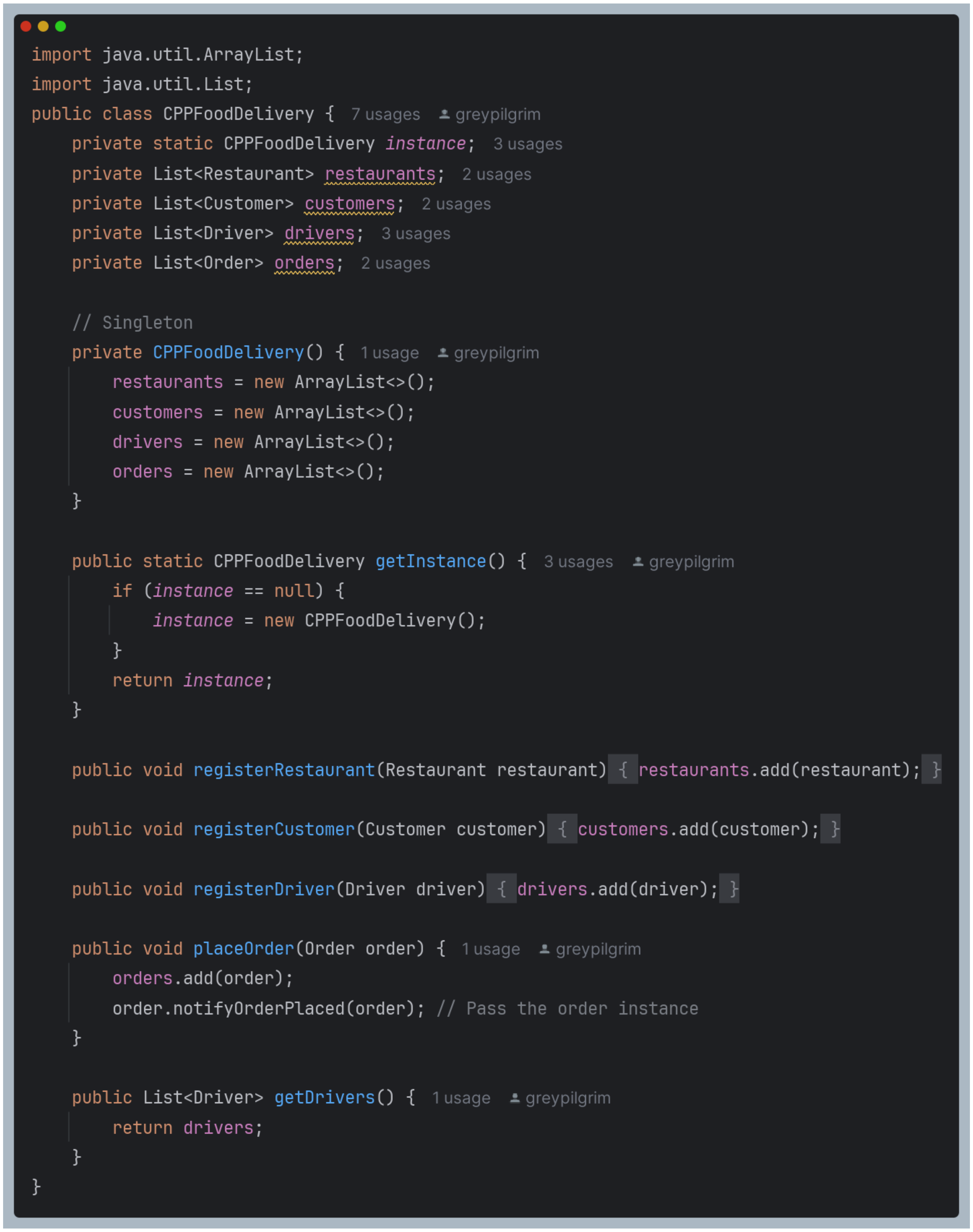
PART 1:



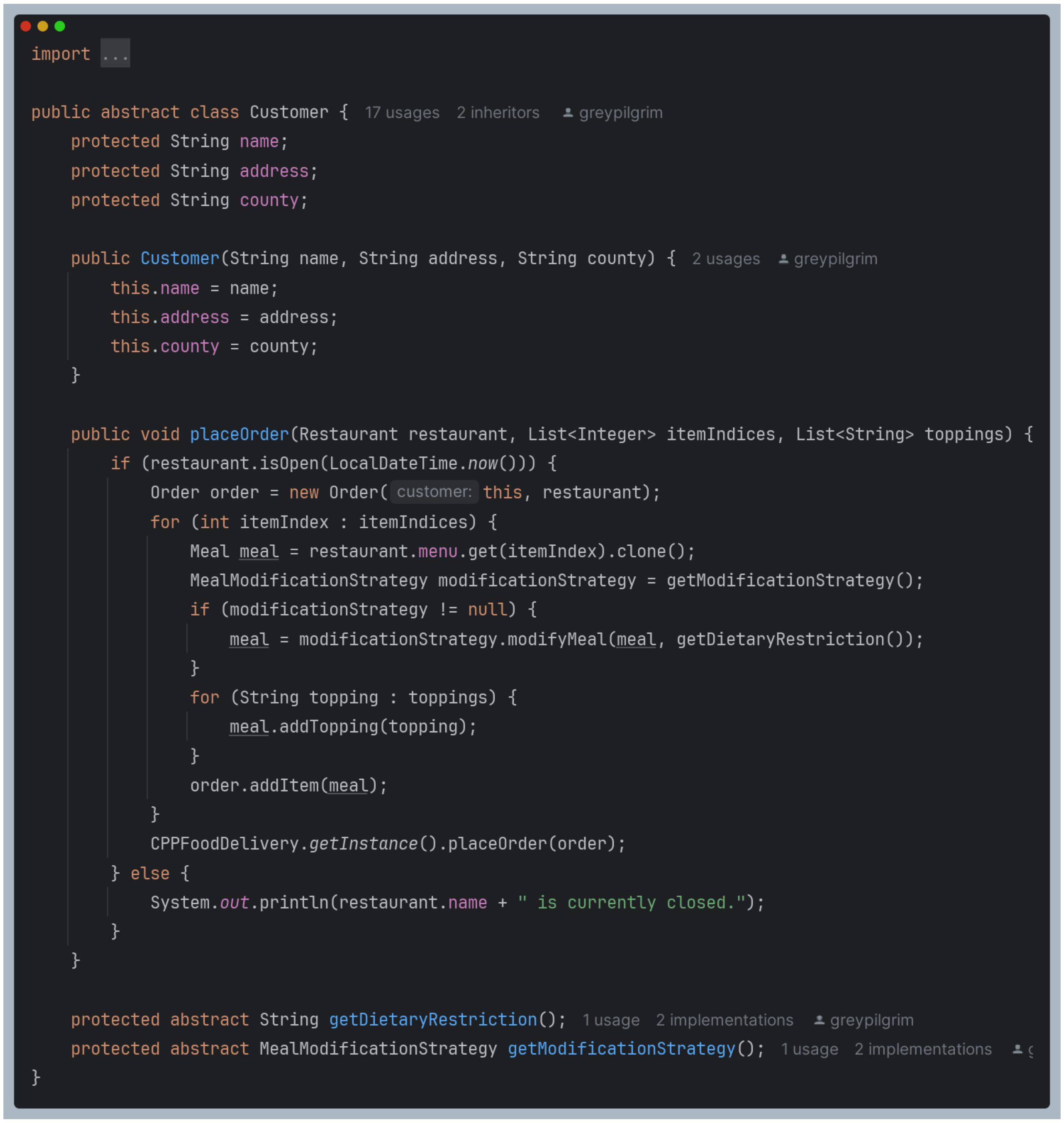
```java
public class AfternoonShiftDriver extends Driver {  1 usage  greypilgrim
    public AfternoonShiftDriver(String name, String address, String county) { super(name, address, county); }
}
```



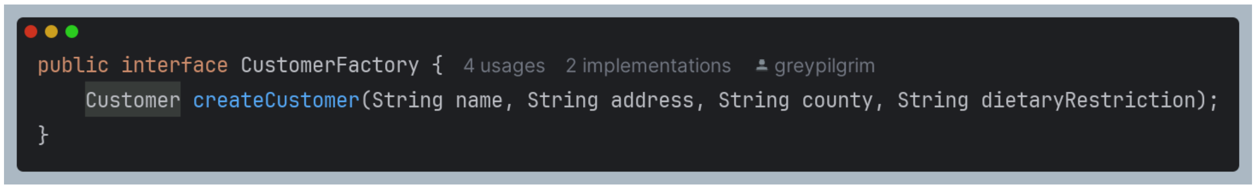
```java
public class AfternoonShiftDriverFactory implements DriverFactory {  1 usage  greypilgrim
    @Override  8 usages  greypilgrim
    public Driver createDriver(String name, String address, String county, String shift) {
        return new AfternoonShiftDriver(name, address, county);
    }
}
```



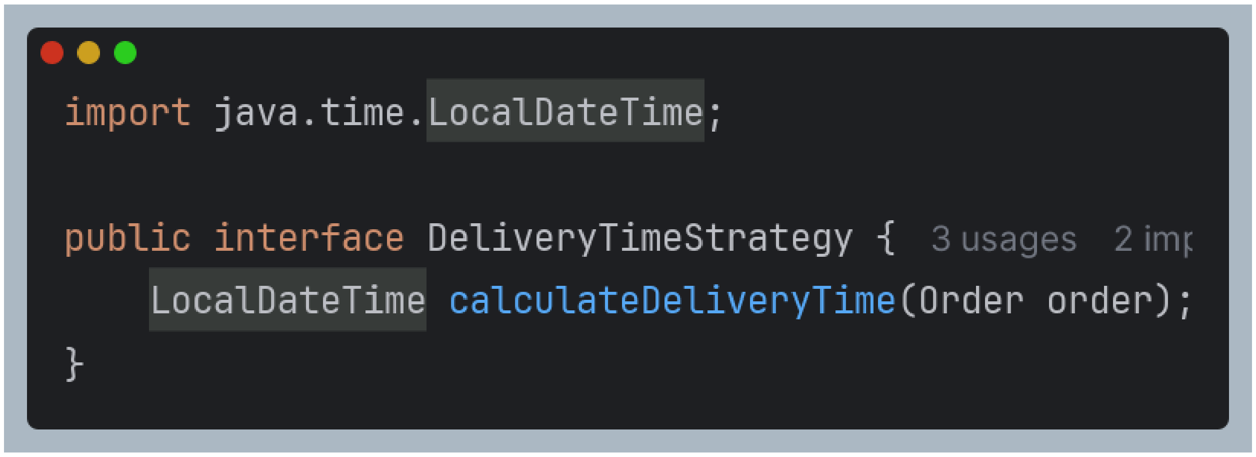
```java
import java.util.ArrayList;
import java.util.List;
public class CPPFoodDelivery {  7 usages   greypilgrim
    private static CPPFoodDelivery instance;  3 usages
    private List<Restaurant> restaurants;  2 usages
    private List<Customer> customers;  2 usages
    private List<Driver> drivers;  3 usages
    private List<Order> orders;  2 usages

    // Singleton
    private CPPFoodDelivery() {  1 usage   greypilgrim
        restaurants = new ArrayList<>();
        customers = new ArrayList<>();
        drivers = new ArrayList<>();
        orders = new ArrayList<>();
    }

    public static CPPFoodDelivery getInstance() {  3 usages   greypilgrim
        if (instance == null) {
            instance = new CPPFoodDelivery();
        }
        return instance;
    }

    public void registerRestaurant(Restaurant restaurant) { restaurants.add(restaurant); }

    public void registerCustomer(Customer customer) { customers.add(customer); }

    public void registerDriver(Driver driver) { drivers.add(driver); }

    public void placeOrder(Order order) {  1 usage   greypilgrim
        orders.add(order);
        order.notifyOrderPlaced(order); // Pass the order instance
    }

    public List<Driver> getDrivers() {  1 usage   greypilgrim
        return drivers;
    }
}
```



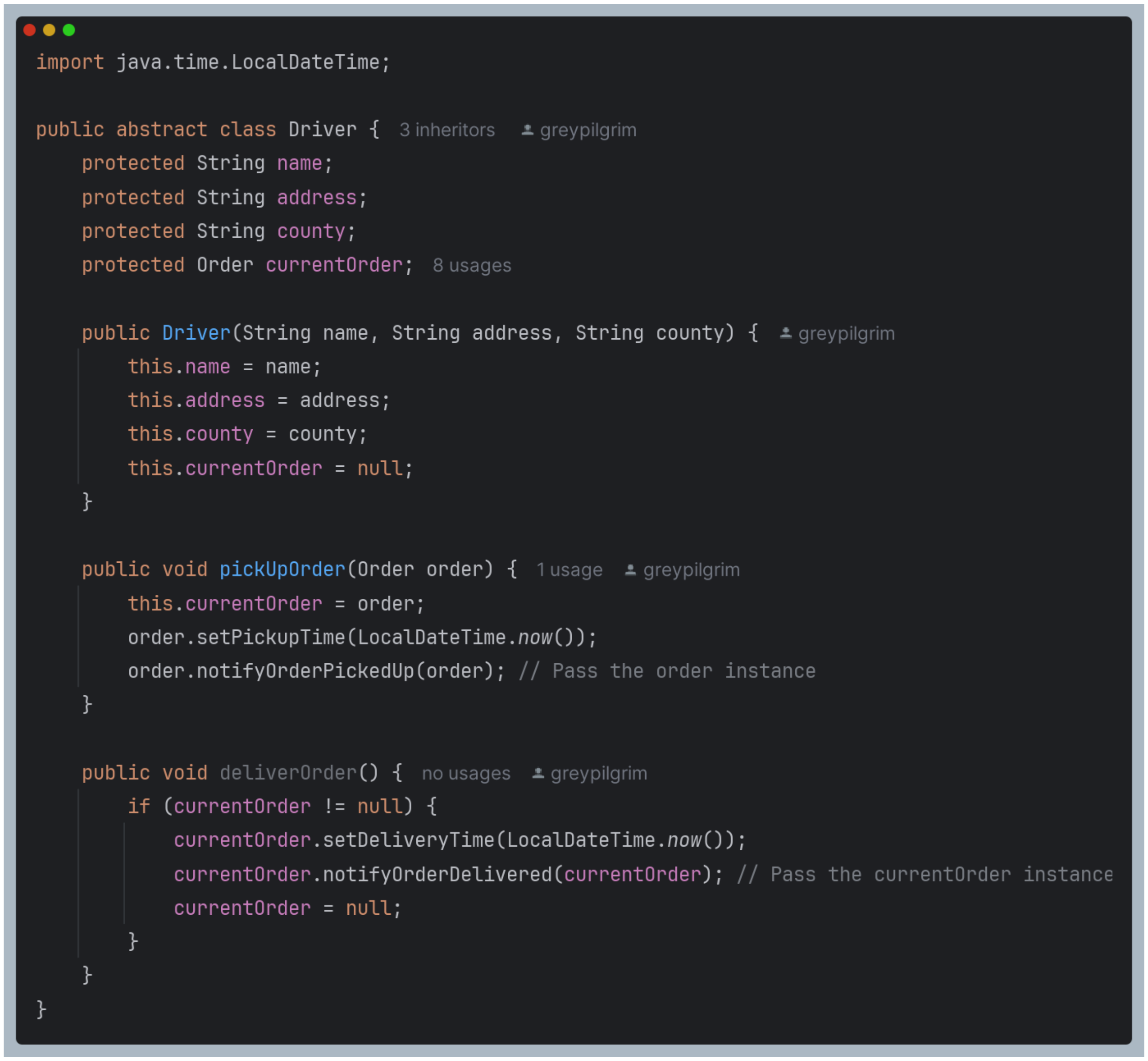
```java
import ...

public abstract class Customer {  17 usages   2 inheritors   greypilgrim
    protected String name;
    protected String address;
    protected String county;

    public Customer(String name, String address, String county) {  2 usages   greypilgrim
        this.name = name;
        this.address = address;
        this.county = county;
    }

    public void placeOrder(Restaurant restaurant, List<Integer> itemIndices, List<String> toppings) {
        if (restaurant.isOpen(LocalDateTime.now())) {
            Order order = new Order( customer: this, restaurant);
            for (int itemIndex : itemIndices) {
                Meal meal = restaurant.menu.get(itemIndex).clone();
                MealModificationStrategy modificationStrategy = getModificationStrategy();
                if (modificationStrategy != null) {
                    meal = modificationStrategy.modifyMeal(meal, getDietaryRestriction());
                }
                for (String topping : toppings) {
                    meal.addTopping(topping);
                }
                order.addItem(meal);
            }
            CPPFoodDelivery.getInstance().placeOrder(order);
        } else {
            System.out.println(restaurant.name + " is currently closed.");
        }
    }

    protected abstract String getDietaryRestriction();  1 usage   2 implementations   greypilgrim
    protected abstract MealModificationStrategy getModificationStrategy();  1 usage   2 implementations
}
```



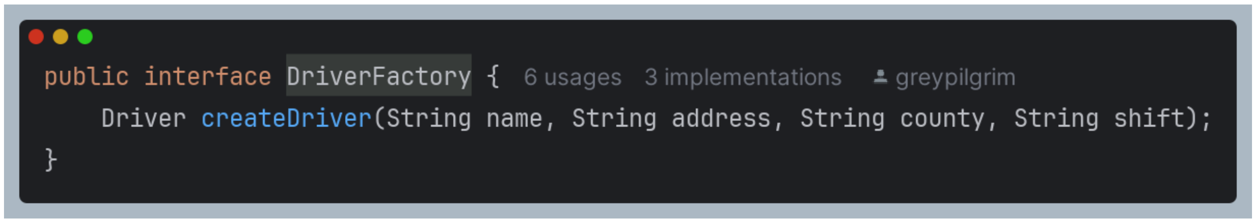
```java
public interface CustomerFactory {  4 usages   2 implementations   greypilgrim
    Customer createCustomer(String name, String address, String county, String dietaryRestriction);
}
```



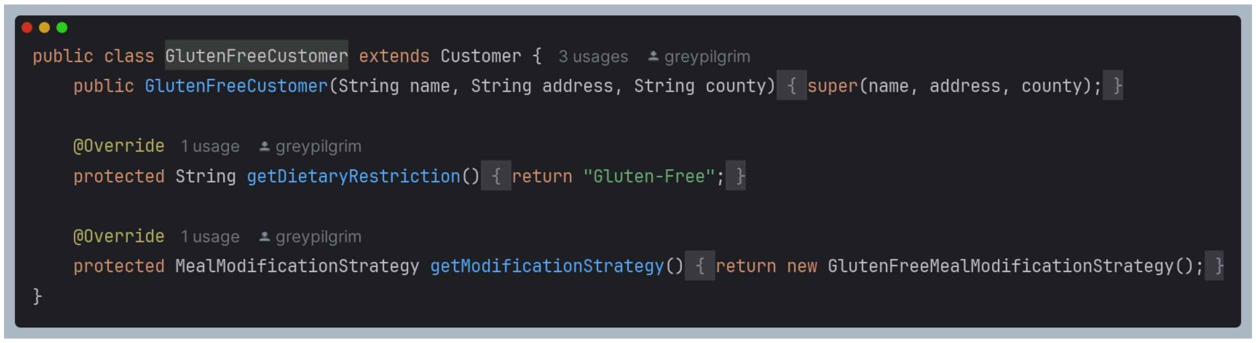
```
import java.time.LocalDateTime;

public interface DeliveryTimeStrategy {  3 usages   2 imp
    LocalDateTime calculateDeliveryTime(Order order);
}
```



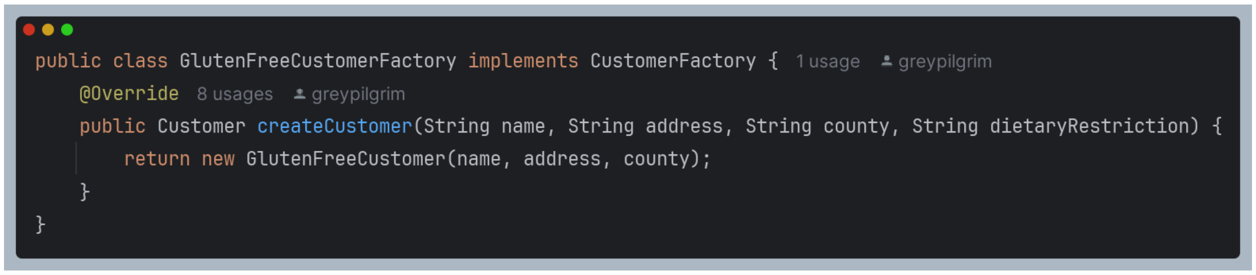
```
import java.time.LocalDateTime;

public abstract class Driver {  3 inheritors   greypilgrim
    protected String name;
    protected String address;
    protected String county;
    protected Order currentOrder;  8 usages

    public Driver(String name, String address, String county) {  greypilgrim
        this.name = name;
        this.address = address;
        this.county = county;
        this.currentOrder = null;
    }

    public void pickUpOrder(Order order) {  1 usage   greypilgrim
        this.currentOrder = order;
        order.setPickupTime(LocalDateTime.now());
        order.notifyOrderPickedUp(order); // Pass the order instance
    }

    public void deliverOrder() {  no usages   greypilgrim
        if (currentOrder != null) {
            currentOrder.setDeliveryTime(LocalDateTime.now());
            currentOrder.notifyOrderDelivered(currentOrder); // Pass the currentOrder instance
            currentOrder = null;
        }
    }
}
```



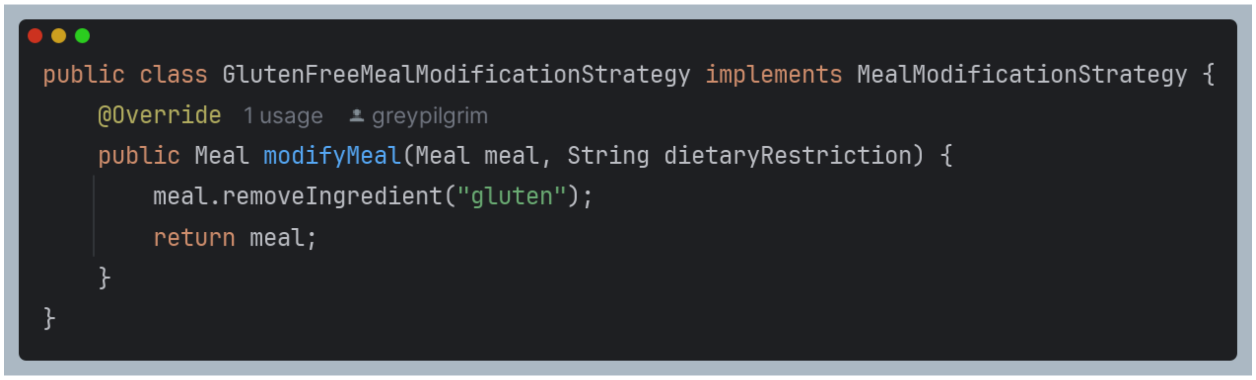
```
public interface DriverFactory {  6 usages  3 implementations  greypilgrim
    Driver createDriver(String name, String address, String county, String shift);
}
```



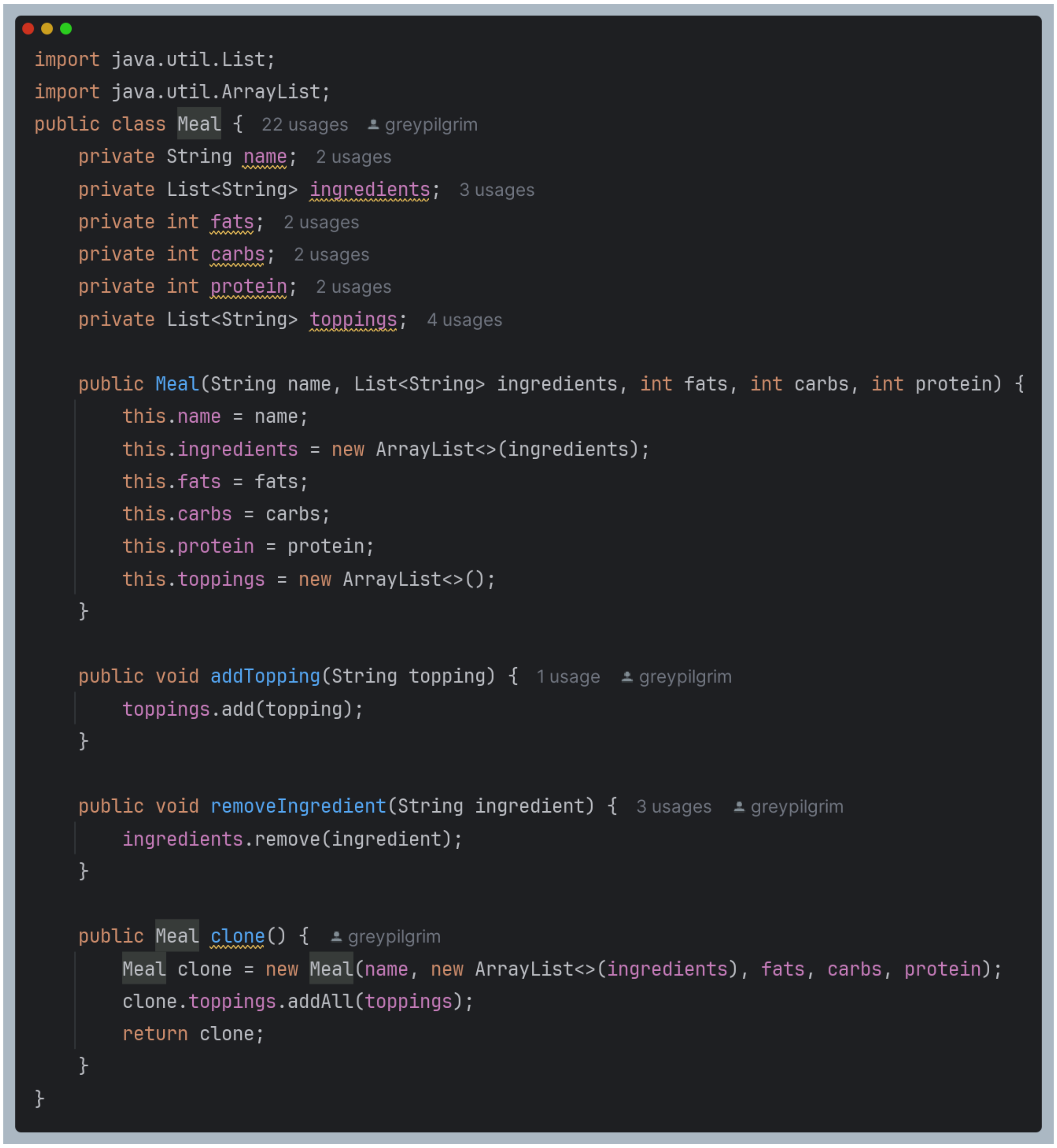
```
public class GlutenFreeCustomer extends Customer {  3 usages  greypilgrim
    public GlutenFreeCustomer(String name, String address, String county) { super(name, address, county); }

    @Override  1 usage  greypilgrim
    protected String getDietaryRestriction() { return "Gluten-Free"; }

    @Override  1 usage  greypilgrim
    protected MealModificationStrategy getModificationStrategy() { return new GlutenFreeMealModificationStrategy(); }
}
```



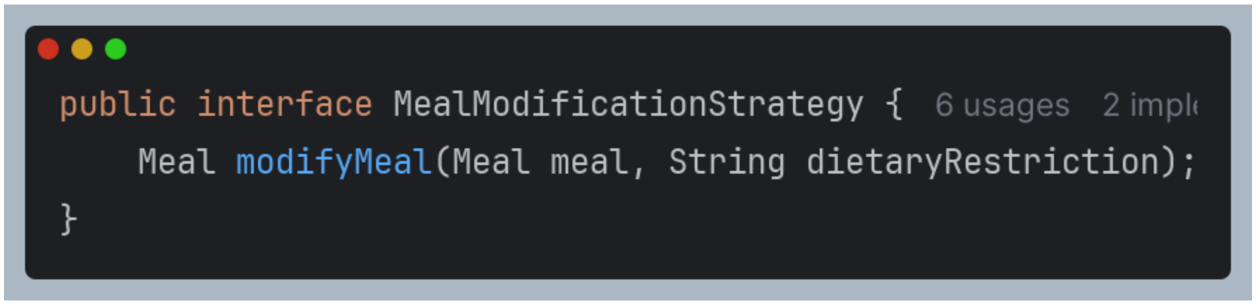
```
public class GlutenFreeCustomerFactory implements CustomerFactory {  1 usage  greypilgrim
    @Override  8 usages  greypilgrim
    public Customer createCustomer(String name, String address, String county, String dietaryRestriction) {
        return new GlutenFreeCustomer(name, address, county);
    }
}
```



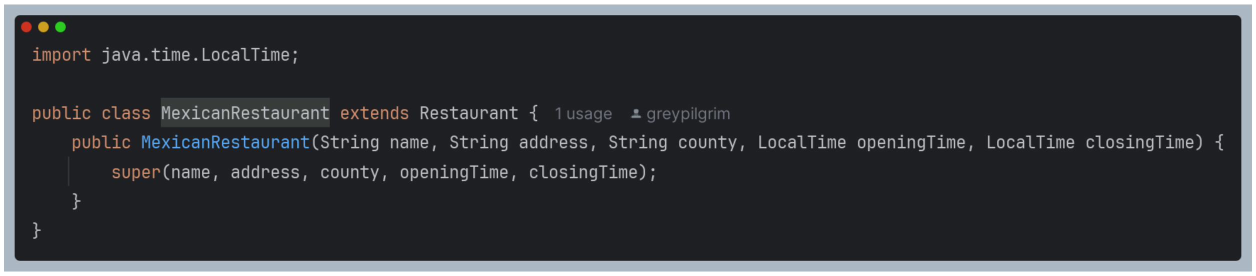
```
public class GlutenFreeMealModificationStrategy implements MealModificationStrategy {
    @Override  1 usage  greypilgrim
    public Meal modifyMeal(Meal meal, String dietaryRestriction) {
        meal.removeIngredient("gluten");
        return meal;
    }
}
```



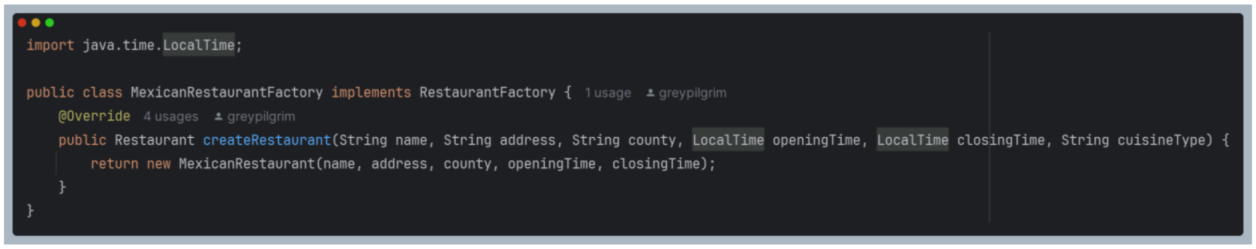
```java
import java.util.List;
import java.util.ArrayList;
public class Meal {
    private String name;
    private List<String> ingredients;
    private int fats;
    private int carbs;
    private int protein;
    private List<String> toppings;

    public Meal(String name, List<String> ingredients, int fats, int carbs, int protein) {
        this.name = name;
        this.ingredients = new ArrayList<>(ingredients);
        this.fats = fats;
        this.carbs = carbs;
        this.protein = protein;
        this.toppings = new ArrayList<>();
    }

    public void addTopping(String topping) {
        toppings.add(topping);
    }

    public void removeIngredient(String ingredient) {
        ingredients.remove(ingredient);
    }

    public Meal clone() {
        Meal clone = new Meal(name, new ArrayList<>(ingredients), fats, carbs, protein);
        clone.toppings.addAll(toppings);
        return clone;
    }
}
```



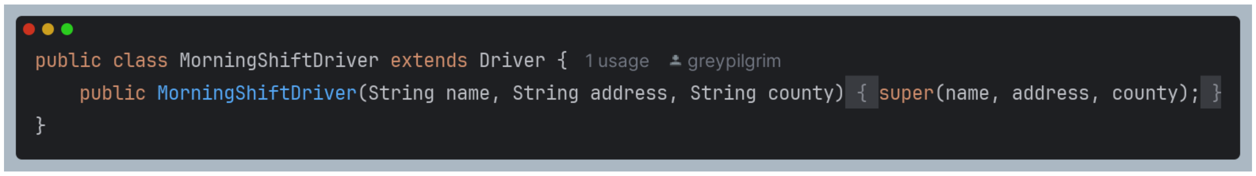
```java
public interface MealModificationStrategy {
    Meal modifyMeal(Meal meal, String dietaryRestriction);
}
```



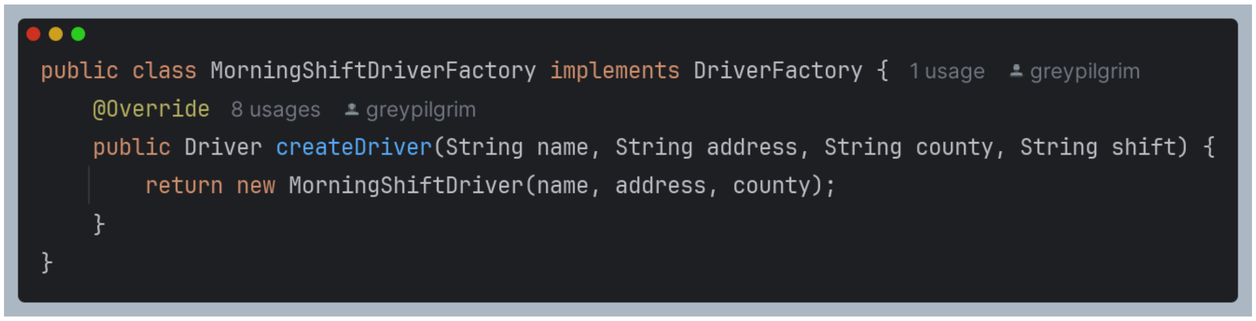
```java
import java.time.LocalTime;

public class MexicanRestaurant extends Restaurant {  1 usage  greypilgrim
    public MexicanRestaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime) {
        super(name, address, county, openingTime, closingTime);
    }
}
```



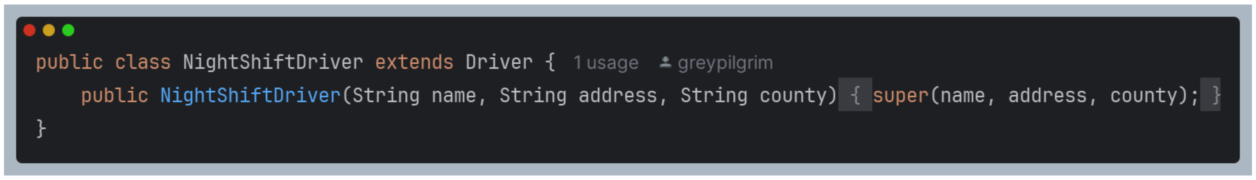
```java
import java.time.LocalTime;

public class MexicanRestaurantFactory implements RestaurantFactory {  1 usage  greypilgrim
    @Override  4 usages  greypilgrim
    public Restaurant createRestaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime, String cuisineType) {
        return new MexicanRestaurant(name, address, county, openingTime, closingTime);
    }
}
```



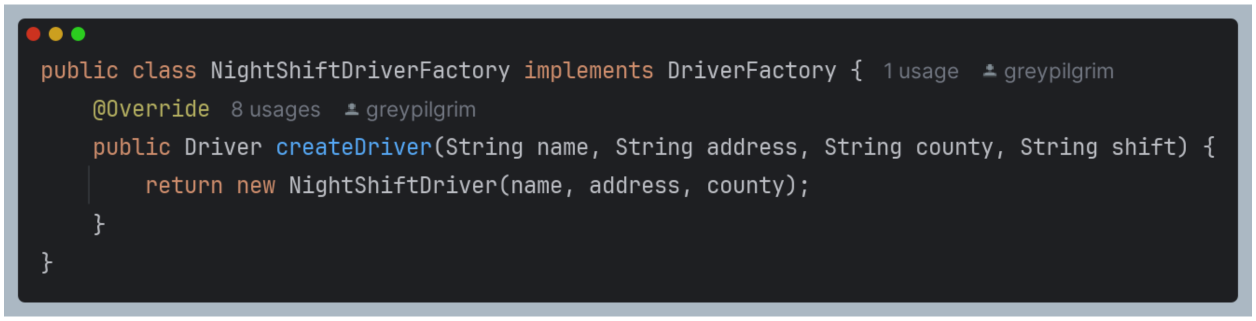
```java
public class MorningShiftDriver extends Driver {  1 usage  greypilgrim
    public MorningShiftDriver(String name, String address, String county) { super(name, address, county); }
}
```



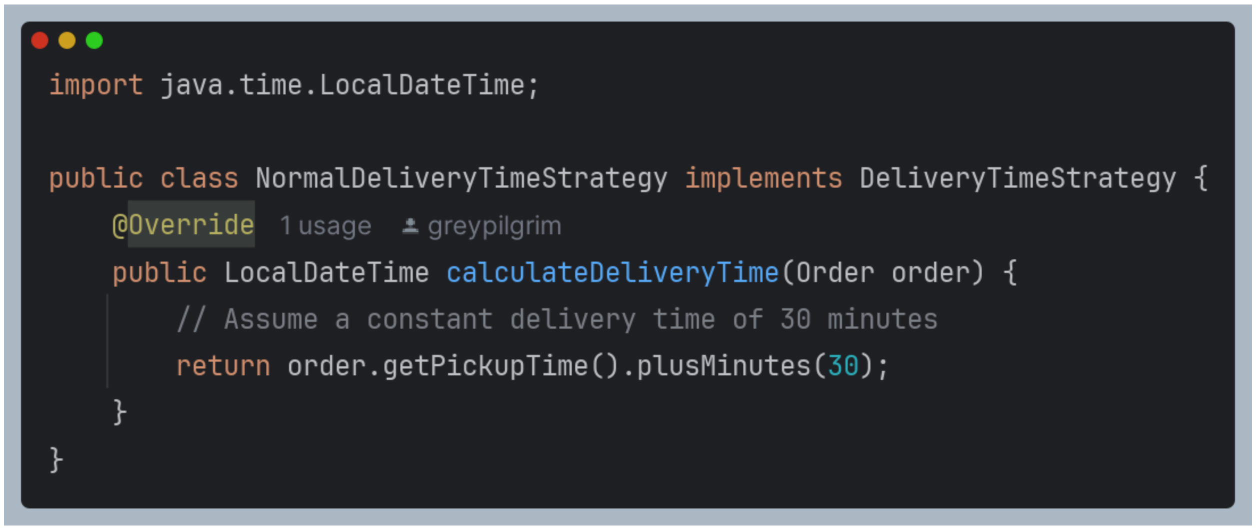
```java
public class MorningShiftDriverFactory implements DriverFactory {  1 usage  greypilgrim
    @Override  8 usages  greypilgrim
    public Driver createDriver(String name, String address, String county, String shift) {
        return new MorningShiftDriver(name, address, county);
    }
}
```

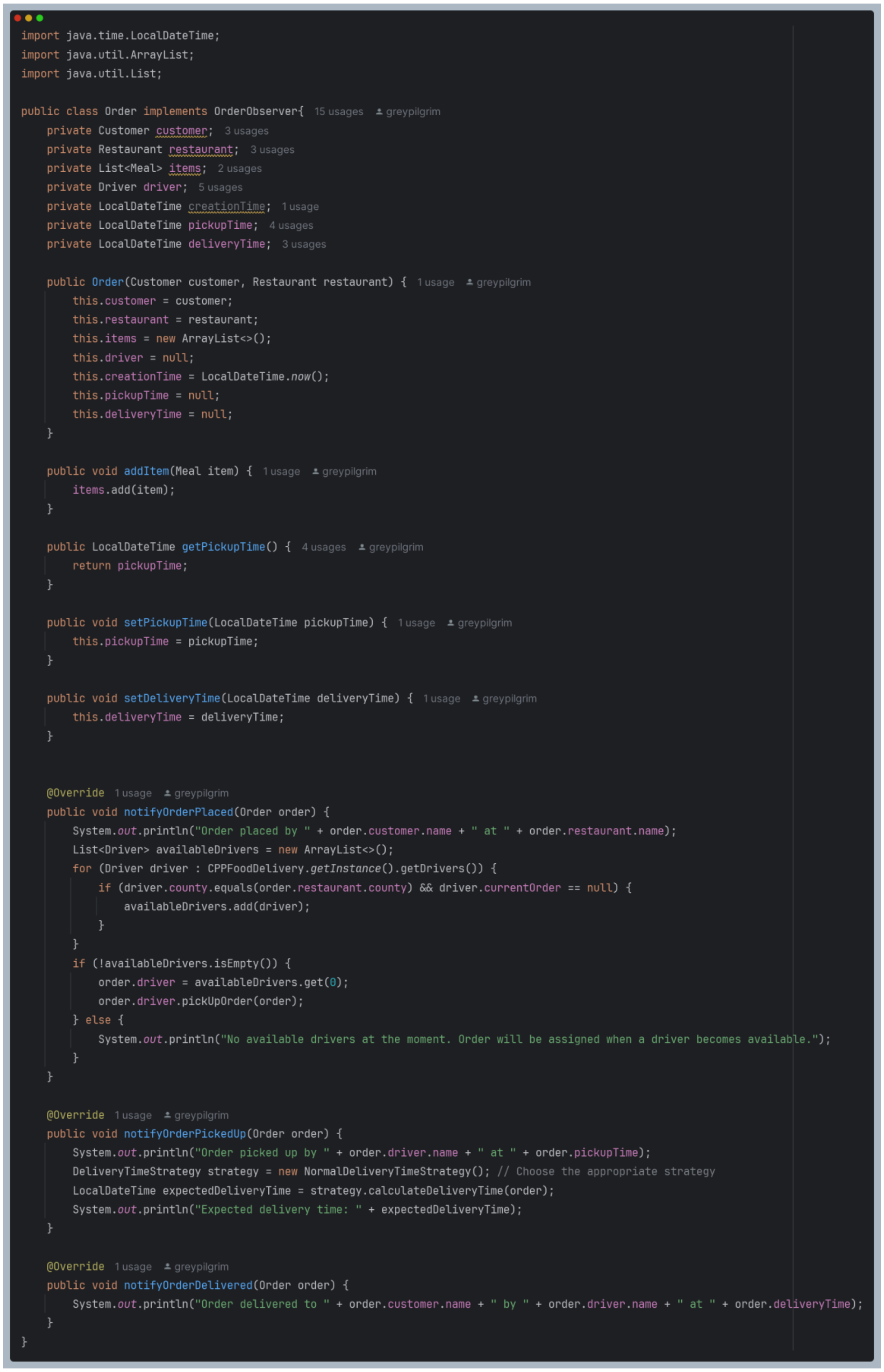


```java
public class NightShiftDriver extends Driver {  1 usage  greypilgrim
    public NightShiftDriver(String name, String address, String county) { super(name, address, county); }
}
```

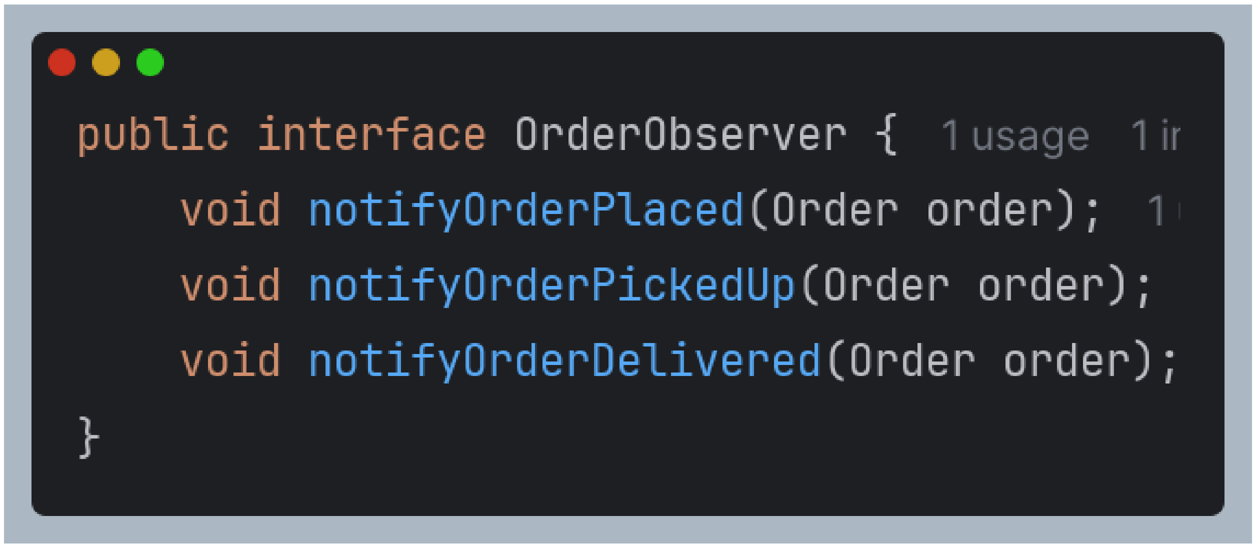


```java
public class NightShiftDriverFactory implements DriverFactory {  1 usage  greypilgrim
    @Override  8 usages  greypilgrim
    public Driver createDriver(String name, String address, String county, String shift) {
        return new NightShiftDriver(name, address, county);
    }
}
```

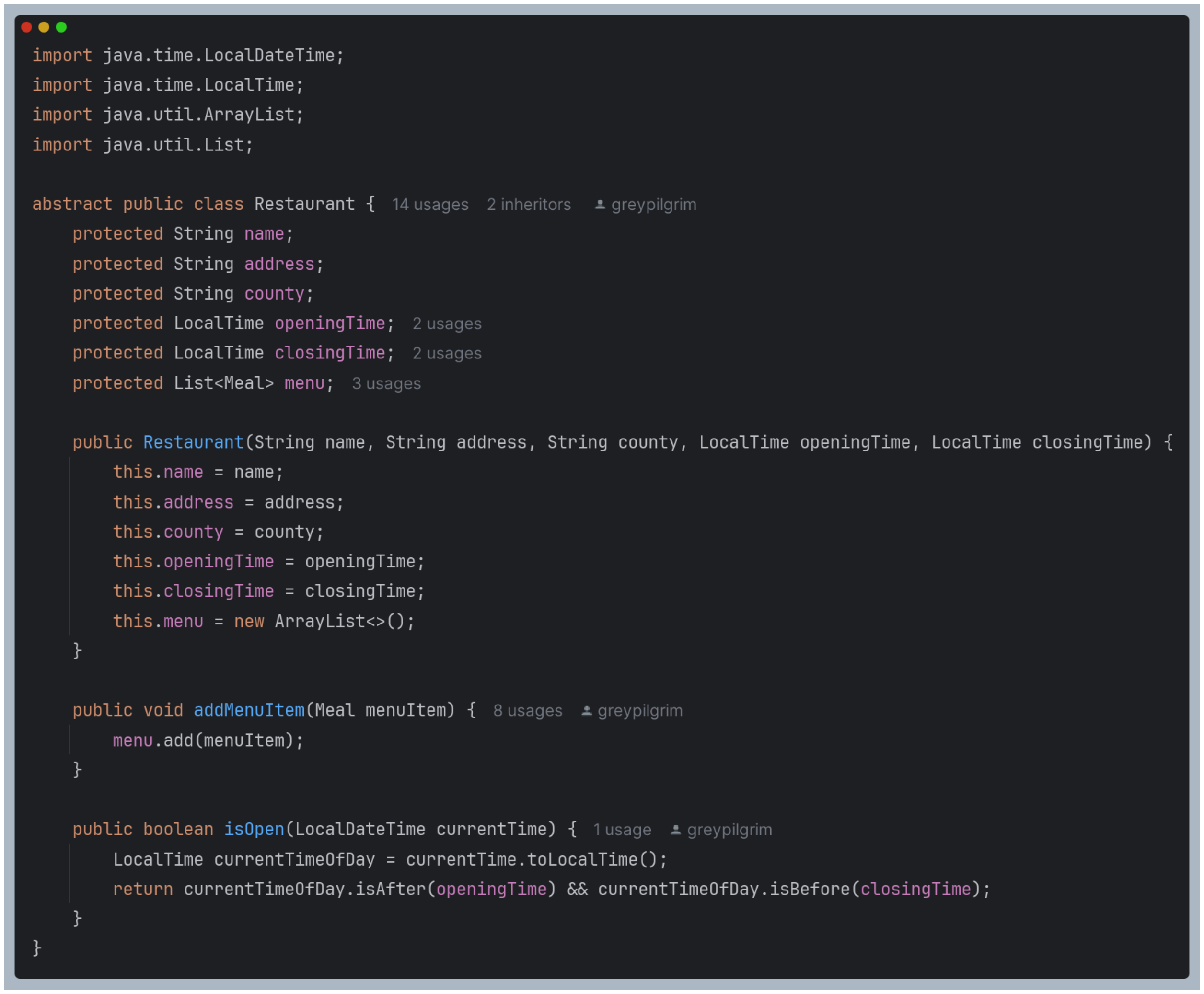

```java
import java.time.LocalDateTime;

public class NormalDeliveryTimeStrategy implements DeliveryTimeStrategy {
    @Override  1 usage  greypilgrim
    public LocalDateTime calculateDeliveryTime(Order order) {
        // Assume a constant delivery time of 30 minutes
        return order.getPickupTime().plusMinutes(30);
    }
}
```



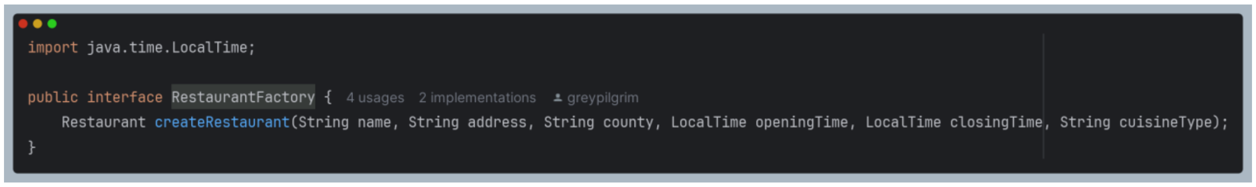
```java
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class Order implements OrderObserver{  15 usages  ≗ greypilgrim
    private Customer customer;  3 usages
    private Restaurant restaurant;  3 usages
    private List<Meal> items;  2 usages
    private Driver driver;  5 usages
    private LocalDateTime creationTime;  1 usage
    private LocalDateTime pickupTime;  4 usages
    private LocalDateTime deliveryTime;  3 usages

    public Order(Customer customer, Restaurant restaurant) {  1 usage  ≗ greypilgrim
        this.customer = customer;
        this.restaurant = restaurant;
        this.items = new ArrayList<>();
        this.driver = null;
        this.creationTime = LocalDateTime.now();
        this.pickupTime = null;
        this.deliveryTime = null;
    }

    public void addItem(Meal item) {  1 usage  ≗ greypilgrim
        items.add(item);
    }

    public LocalDateTime getPickupTime() {  4 usages  ≗ greypilgrim
        return pickupTime;
    }

    public void setPickupTime(LocalDateTime pickupTime) {  1 usage  ≗ greypilgrim
        this.pickupTime = pickupTime;
    }

    public void setDeliveryTime(LocalDateTime deliveryTime) {  1 usage  ≗ greypilgrim
        this.deliveryTime = deliveryTime;
    }


    @Override  1 usage  ≗ greypilgrim
    public void notifyOrderPlaced(Order order) {
        System.out.println("Order placed by " + order.customer.name + " at " + order.restaurant.name);
        List<Driver> availableDrivers = new ArrayList<>();
        for (Driver driver : CPPFoodDelivery.getInstance().getDrivers()) {
            if (driver.county.equals(order.restaurant.county) && driver.currentOrder == null) {
                availableDrivers.add(driver);
            }
        }
        if (!availableDrivers.isEmpty()) {
            order.driver = availableDrivers.get(0);
            order.driver.pickUpOrder(order);
        } else {
            System.out.println("No available drivers at the moment. Order will be assigned when a driver becomes available.");
        }
    }

    @Override  1 usage  ≗ greypilgrim
    public void notifyOrderPickedUp(Order order) {
        System.out.println("Order picked up by " + order.driver.name + " at " + order.pickupTime);
        DeliveryTimeStrategy strategy = new NormalDeliveryTimeStrategy(); // Choose the appropriate strategy
        LocalDateTime expectedDeliveryTime = strategy.calculateDeliveryTime(order);
        System.out.println("Expected delivery time: " + expectedDeliveryTime);
    }

    @Override  1 usage  ≗ greypilgrim
    public void notifyOrderDelivered(Order order) {
        System.out.println("Order delivered to " + order.customer.name + " by " + order.driver.name + " at " + order.deliveryTime);
    }
}
```



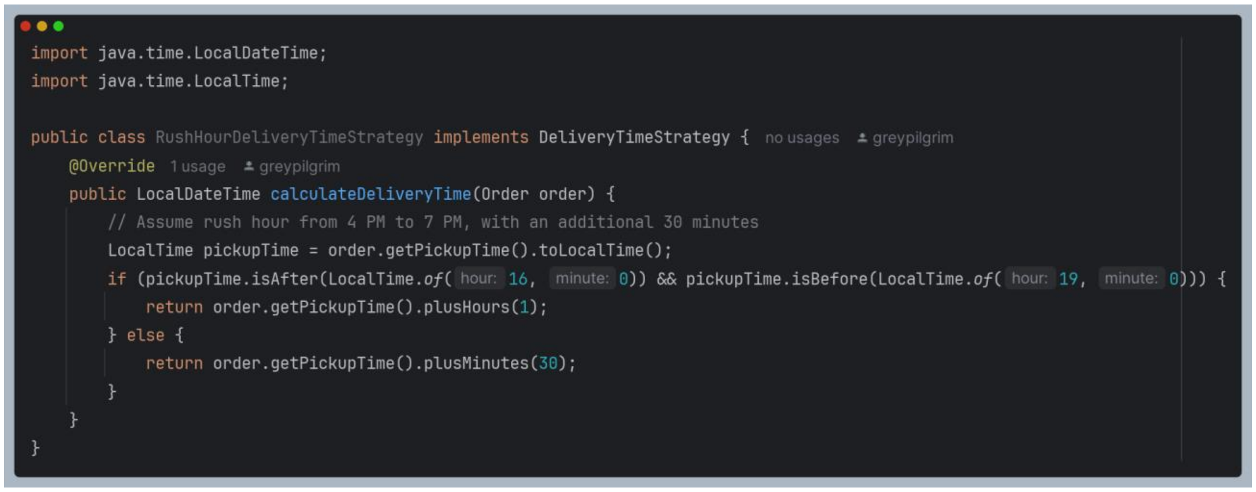
```
public interface OrderObserver {  1 usage   1 ir
    void notifyOrderPlaced(Order order);  1
    void notifyOrderPickedUp(Order order);
    void notifyOrderDelivered(Order order);
}
```



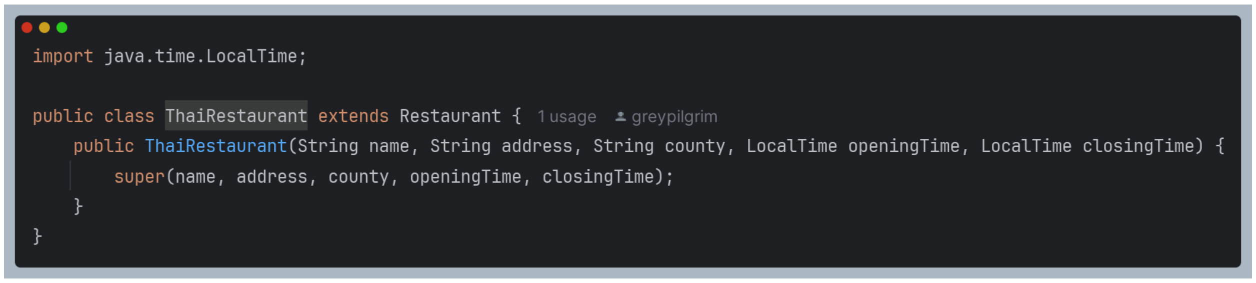
```
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;

abstract public class Restaurant {  14 usages   2 inheritors   greypilgrim
    protected String name;
    protected String address;
    protected String county;
    protected LocalTime openingTime;  2 usages
    protected LocalTime closingTime;  2 usages
    protected List<Meal> menu;  3 usages

    public Restaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime) {
        this.name = name;
        this.address = address;
        this.county = county;
        this.openingTime = openingTime;
        this.closingTime = closingTime;
        this.menu = new ArrayList<>();
    }

    public void addMenuItem(Meal menuItem) {  8 usages   greypilgrim
        menu.add(menuItem);
    }

    public boolean isOpen(LocalDateTime currentTime) {  1 usage   greypilgrim
        LocalTime currentTimeOfDay = currentTime.toLocalTime();
        return currentTimeOfDay.isAfter(openingTime) && currentTimeOfDay.isBefore(closingTime);
    }
}
```



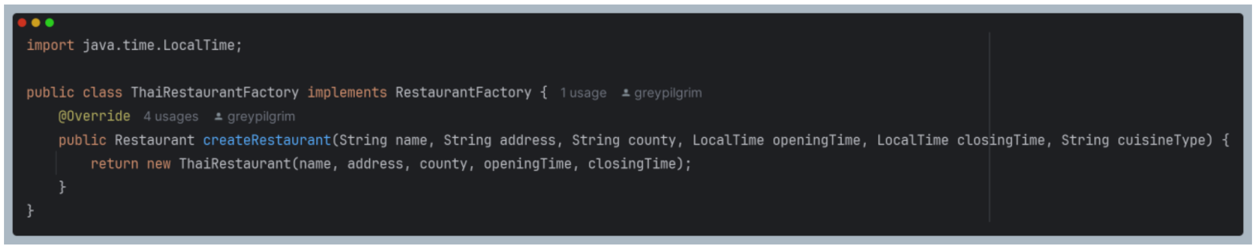
```
import java.time.LocalTime;

public interface RestaurantFactory {  4 usages  2 implementations  greypilgrim
    Restaurant createRestaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime, String cuisineType);
}
```



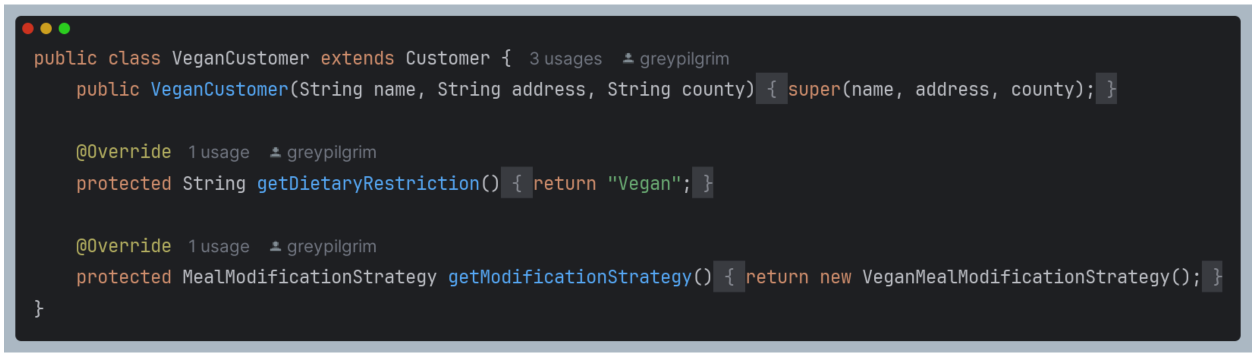
```
import java.time.LocalDateTime;
import java.time.LocalTime;

public class RushHourDeliveryTimeStrategy implements DeliveryTimeStrategy {  no usages  greypilgrim
    @Override  1 usage  greypilgrim
    public LocalDateTime calculateDeliveryTime(Order order) {
        // Assume rush hour from 4 PM to 7 PM, with an additional 30 minutes
        LocalTime pickupTime = order.getPickupTime().toLocalTime();
        if (pickupTime.isAfter(LocalTime.of( hour: 16,  minute: 0)) && pickupTime.isBefore(LocalTime.of( hour: 19,  minute: 0))) {
            return order.getPickupTime().plusHours(1);
        } else {
            return order.getPickupTime().plusMinutes(30);
        }
    }
}
```



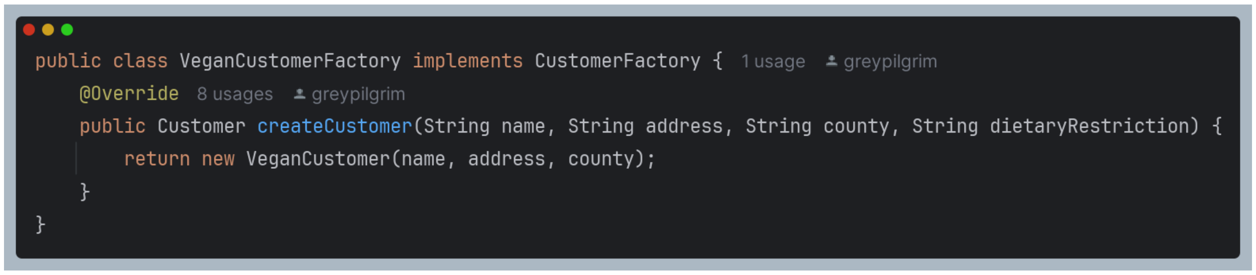
```
import java.time.LocalTime;

public class ThaiRestaurant extends Restaurant {  1 usage  greypilgrim
    public ThaiRestaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime) {
        super(name, address, county, openingTime, closingTime);
    }
}
```



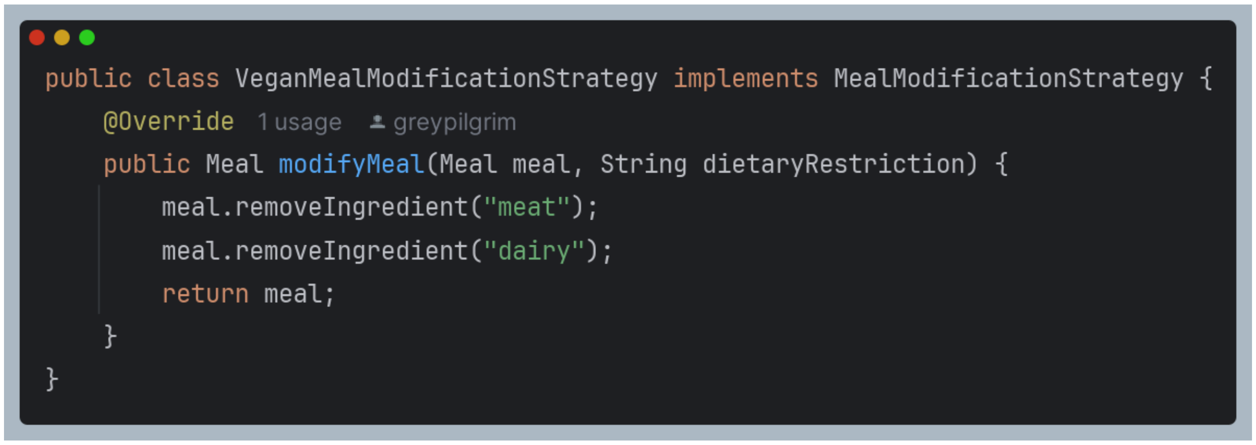
```
import java.time.LocalTime;

public class ThaiRestaurantFactory implements RestaurantFactory {  1 usage  greypilgrim
    @Override  4 usages  greypilgrim
    public Restaurant createRestaurant(String name, String address, String county, LocalTime openingTime, LocalTime closingTime, String cuisineType) {
        return new ThaiRestaurant(name, address, county, openingTime, closingTime);
    }
}
```



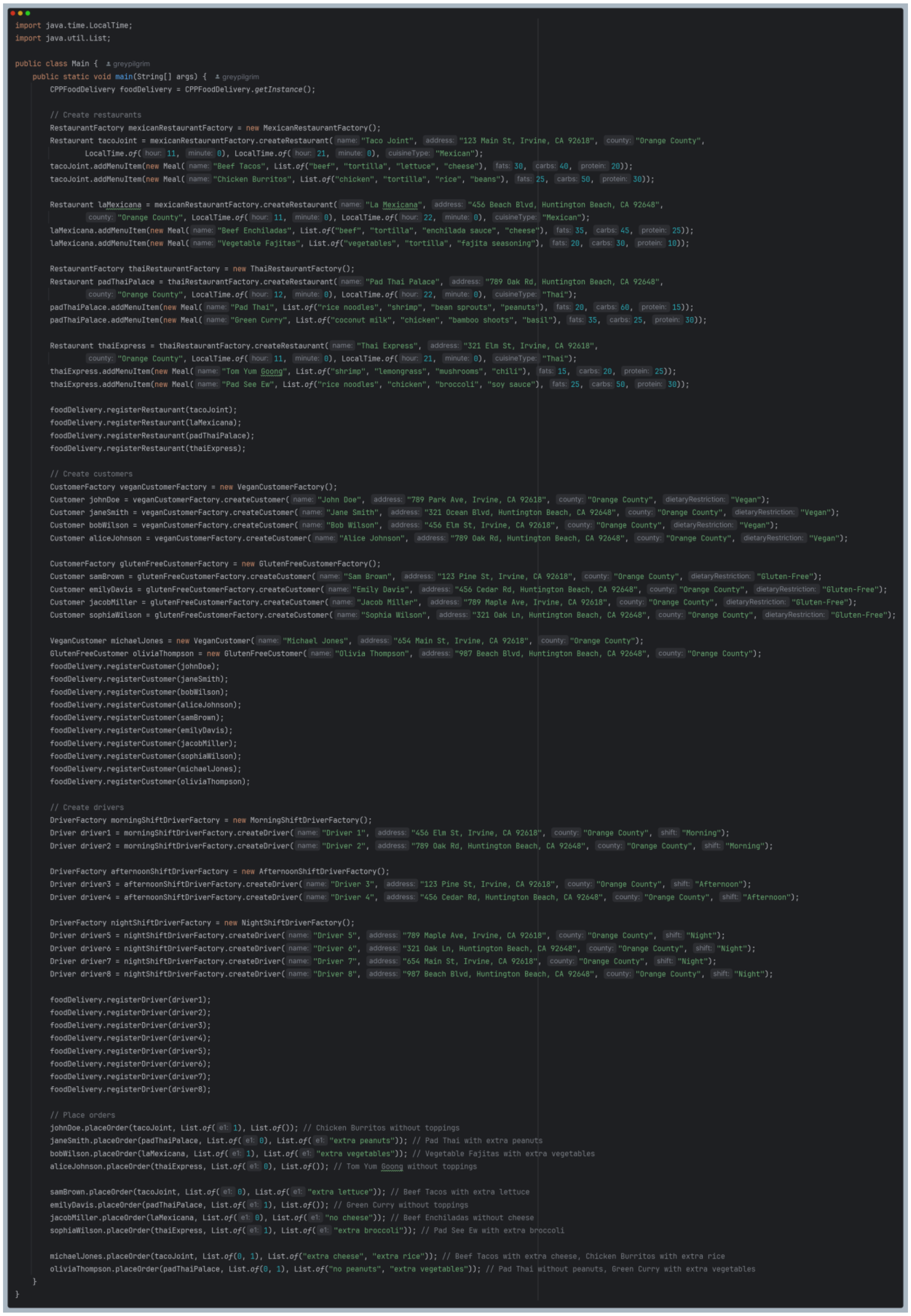
```
public class VeganCustomer extends Customer {  3 usages  greypilgrim
    public VeganCustomer(String name, String address, String county) { super(name, address, county); }

    @Override  1 usage  greypilgrim
    protected String getDietaryRestriction() { return "Vegan"; }

    @Override  1 usage  greypilgrim
    protected MealModificationStrategy getModificationStrategy() { return new VeganMealModificationStrategy(); }
}
```



```
public class VeganCustomerFactory implements CustomerFactory {  1 usage  greypilgrim
    @Override  8 usages  greypilgrim
    public Customer createCustomer(String name, String address, String county, String dietaryRestriction) {
        return new VeganCustomer(name, address, county);
    }
}
```

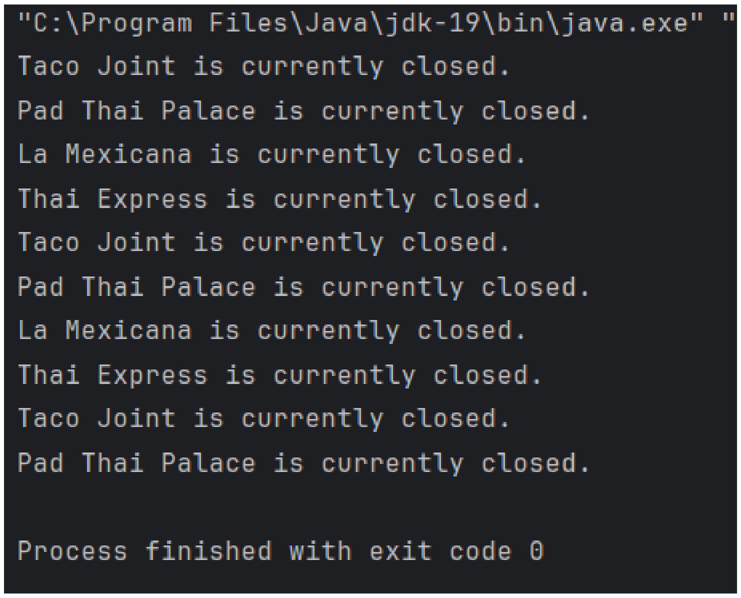


```
public class VeganMealModificationStrategy implements MealModificationStrategy {
    @Override  1 usage  greypilgrim
    public Meal modifyMeal(Meal meal, String dietaryRestriction) {
        meal.removeIngredient("meat");
        meal.removeIngredient("dairy");
        return meal;
    }
}
```



```java
import java.time.LocalTime;
import java.util.List;

public class Main {  greypilgrim
    public static void main(String[] args) {  greypilgrim
        CPPFoodDelivery foodDelivery = CPPFoodDelivery.getInstance();

        // Create restaurants
        RestaurantFactory mexicanRestaurantFactory = new MexicanRestaurantFactory();
        Restaurant tacoJoint = mexicanRestaurantFactory.createRestaurant( name: "Taco Joint",  address: "123 Main St, Irvine, CA 92618",  county: "Orange County",
                LocalTime.of( hour: 11,  minute: 0), LocalTime.of( hour: 21,  minute: 0),  cuisineType: "Mexican");
        tacoJoint.addMenuItem(new Meal( name: "Beef Tacos", List.of("beef", "tortilla", "lettuce", "cheese"),  fats: 30,  carbs: 40,  protein: 20));
        tacoJoint.addMenuItem(new Meal( name: "Chicken Burritos", List.of("chicken", "tortilla", "rice", "beans"),  fats: 25,  carbs: 50,  protein: 30));

        Restaurant laMexicana = mexicanRestaurantFactory.createRestaurant( name: "La Mexicana",  address: "456 Beach Blvd, Huntington Beach, CA 92648",
                 county: "Orange County", LocalTime.of( hour: 11,  minute: 0), LocalTime.of( hour: 22,  minute: 0),  cuisineType: "Mexican");
        laMexicana.addMenuItem(new Meal( name: "Beef Enchiladas", List.of("beef", "tortilla", "enchilada sauce", "cheese"),  fats: 35,  carbs: 45,  protein: 25));
        laMexicana.addMenuItem(new Meal( name: "Vegetable Fajitas", List.of("vegetables", "tortilla", "fajita seasoning"),  fats: 20,  carbs: 30,  protein: 10));

        RestaurantFactory thaiRestaurantFactory = new ThaiRestaurantFactory();
        Restaurant padThaiPalace = thaiRestaurantFactory.createRestaurant( name: "Pad Thai Palace",  address: "789 Oak Rd, Huntington Beach, CA 92648",
                 county: "Orange County", LocalTime.of( hour: 12,  minute: 0), LocalTime.of( hour: 22,  minute: 0),  cuisineType: "Thai");
        padThaiPalace.addMenuItem(new Meal( name: "Pad Thai", List.of("rice noodles", "shrimp", "bean sprouts", "peanuts"),  fats: 20,  carbs: 60,  protein: 15));
        padThaiPalace.addMenuItem(new Meal( name: "Green Curry", List.of("coconut milk", "chicken", "bamboo shoots", "basil"),  fats: 35,  carbs: 25,  protein: 30));

        Restaurant thaiExpress = thaiRestaurantFactory.createRestaurant( name: "Thai Express",  address: "321 Elm St, Irvine, CA 92618",
                 county: "Orange County", LocalTime.of( hour: 11,  minute: 0), LocalTime.of( hour: 21,  minute: 0),  cuisineType: "Thai");
        thaiExpress.addMenuItem(new Meal( name: "Tom Yum Goong", List.of("shrimp", "lemongrass", "mushrooms", "chili"),  fats: 15,  carbs: 20,  protein: 25));
        thaiExpress.addMenuItem(new Meal( name: "Pad See Ew", List.of("rice noodles", "chicken", "broccoli", "soy sauce"),  fats: 25,  carbs: 50,  protein: 30));

        foodDelivery.registerRestaurant(tacoJoint);
        foodDelivery.registerRestaurant(laMexicana);
        foodDelivery.registerRestaurant(padThaiPalace);
        foodDelivery.registerRestaurant(thaiExpress);

        // Create customers
        CustomerFactory veganCustomerFactory = new VeganCustomerFactory();
        Customer johnDoe = veganCustomerFactory.createCustomer( name: "John Doe",  address: "789 Park Ave, Irvine, CA 92618",  county: "Orange County",  dietaryRestriction: "Vegan");
        Customer janeSmith = veganCustomerFactory.createCustomer( name: "Jane Smith",  address: "321 Ocean Blvd, Huntington Beach, CA 92648",  county: "Orange County",  dietaryRestriction: "Vegan");
        Customer bobWilson = veganCustomerFactory.createCustomer( name: "Bob Wilson",  address: "456 Elm St, Irvine, CA 92618",  county: "Orange County",  dietaryRestriction: "Vegan");
        Customer aliceJohnson = veganCustomerFactory.createCustomer( name: "Alice Johnson",  address: "789 Oak Rd, Huntington Beach, CA 92648",  county: "Orange County",  dietaryRestriction: "Vegan");

        CustomerFactory glutenFreeCustomerFactory = new GlutenFreeCustomerFactory();
        Customer samBrown = glutenFreeCustomerFactory.createCustomer( name: "Sam Brown",  address: "123 Pine St, Irvine, CA 92618",  county: "Orange County",  dietaryRestriction: "Gluten-Free");
        Customer emilyDavis = glutenFreeCustomerFactory.createCustomer( name: "Emily Davis",  address: "456 Cedar Rd, Huntington Beach, CA 92648",  county: "Orange County",  dietaryRestriction: "Gluten-Free");
        Customer jacobMiller = glutenFreeCustomerFactory.createCustomer( name: "Jacob Miller",  address: "789 Maple Ave, Irvine, CA 92618",  county: "Orange County",  dietaryRestriction: "Gluten-Free");
        Customer sophiaWilson = glutenFreeCustomerFactory.createCustomer( name: "Sophia Wilson",  address: "321 Oak Ln, Huntington Beach, CA 92648",  county: "Orange County",  dietaryRestriction: "Gluten-Free");

        VeganCustomer michaelJones = new VeganCustomer( name: "Michael Jones",  address: "654 Main St, Irvine, CA 92618",  county: "Orange County");
        GlutenFreeCustomer oliviaThompson = new GlutenFreeCustomer( name: "Olivia Thompson",  address: "987 Beach Blvd, Huntington Beach, CA 92648",  county: "Orange County");
        foodDelivery.registerCustomer(johnDoe);
        foodDelivery.registerCustomer(janeSmith);
        foodDelivery.registerCustomer(bobWilson);
        foodDelivery.registerCustomer(aliceJohnson);
        foodDelivery.registerCustomer(samBrown);
        foodDelivery.registerCustomer(emilyDavis);
        foodDelivery.registerCustomer(jacobMiller);
        foodDelivery.registerCustomer(sophiaWilson);
        foodDelivery.registerCustomer(michaelJones);
        foodDelivery.registerCustomer(oliviaThompson);

        // Create drivers
        DriverFactory morningShiftDriverFactory = new MorningShiftDriverFactory();
        Driver driver1 = morningShiftDriverFactory.createDriver( name: "Driver 1",  address: "456 Elm St, Irvine, CA 92618",  county: "Orange County",  shift: "Morning");
        Driver driver2 = morningShiftDriverFactory.createDriver( name: "Driver 2",  address: "789 Oak Rd, Huntington Beach, CA 92648",  county: "Orange County",  shift: "Morning");

        DriverFactory afternoonShiftDriverFactory = new AfternoonShiftDriverFactory();
        Driver driver3 = afternoonShiftDriverFactory.createDriver( name: "Driver 3",  address: "123 Pine St, Irvine, CA 92618",  county: "Orange County",  shift: "Afternoon");
        Driver driver4 = afternoonShiftDriverFactory.createDriver( name: "Driver 4",  address: "456 Cedar Rd, Huntington Beach, CA 92648",  county: "Orange County",  shift: "Afternoon");

        DriverFactory nightShiftDriverFactory = new NightShiftDriverFactory();
        Driver driver5 = nightShiftDriverFactory.createDriver( name: "Driver 5",  address: "789 Maple Ave, Irvine, CA 92618",  county: "Orange County",  shift: "Night");
        Driver driver6 = nightShiftDriverFactory.createDriver( name: "Driver 6",  address: "321 Oak Ln, Huntington Beach, CA 92648",  county: "Orange County",  shift: "Night");
        Driver driver7 = nightShiftDriverFactory.createDriver( name: "Driver 7",  address: "654 Main St, Irvine, CA 92618",  county: "Orange County",  shift: "Night");
        Driver driver8 = nightShiftDriverFactory.createDriver( name: "Driver 8",  address: "987 Beach Blvd, Huntington Beach, CA 92648",  county: "Orange County",  shift: "Night");

        foodDelivery.registerDriver(driver1);
        foodDelivery.registerDriver(driver2);
        foodDelivery.registerDriver(driver3);
        foodDelivery.registerDriver(driver4);
        foodDelivery.registerDriver(driver5);
        foodDelivery.registerDriver(driver6);
        foodDelivery.registerDriver(driver7);
        foodDelivery.registerDriver(driver8);

        // Place orders
        johnDoe.placeOrder(tacoJoint, List.of( e1: 1), List.of()); // Chicken Burritos without toppings
        janeSmith.placeOrder(padThaiPalace, List.of( e1: 0), List.of( e1: "extra peanuts")); // Pad Thai with extra peanuts
        bobWilson.placeOrder(laMexicana, List.of( e1: 1), List.of( e1: "extra vegetables")); // Vegetable Fajitas with extra vegetables
        aliceJohnson.placeOrder(thaiExpress, List.of( e1: 0), List.of()); // Tom Yum Goong without toppings

        samBrown.placeOrder(tacoJoint, List.of( e1: 0), List.of( e1: "extra lettuce")); // Beef Tacos with extra lettuce
        emilyDavis.placeOrder(padThaiPalace, List.of( e1: 1), List.of()); // Green Curry without toppings
        jacobMiller.placeOrder(laMexicana, List.of( e1: 0), List.of( e1: "no cheese")); // Beef Enchiladas without cheese
        sophiaWilson.placeOrder(thaiExpress, List.of( e1: 1), List.of( e1: "extra broccoli")); // Pad See Ew with extra broccoli

        michaelJones.placeOrder(tacoJoint, List.of(0, 1), List.of("extra cheese", "extra rice")); // Beef Tacos with extra cheese, Chicken Burritos with extra rice
        oliviaThompson.placeOrder(padThaiPalace, List.of(0, 1), List.of("no peanuts", "extra vegetables")); // Pad Thai without peanuts, Green Curry with extra vegetables
    }
}
```

Output: (**NOTE: The output looks the way it does because of the local time being used. If this code is run during daylight hours(before 10pm), you will see different outputs**)



```
"C:\Program Files\Java\jdk-19\bin\java.exe" "
Taco Joint is currently closed.
Pad Thai Palace is currently closed.
La Mexicana is currently closed.
Thai Express is currently closed.
Taco Joint is currently closed.
Pad Thai Palace is currently closed.
La Mexicana is currently closed.
Thai Express is currently closed.
Taco Joint is currently closed.
Pad Thai Palace is currently closed.

Process finished with exit code 0
```