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

// Base User class
class User {
    private String firstName;
    private String lastName;
    private String phoneNumber;

    public User(String firstName, String lastName, String phoneNumber) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.phoneNumber = phoneNumber;
    }

    // Getters and setters
    public String getFirstName() { return firstName; }
    public String getLastName() { return lastName; }
    public String getPhoneNumber() { return phoneNumber; }
}

// Fruit class
class Fruit {
    private String name;
    private String variety;
    private double totalQuantityKg;
    private double pricePerKg;

    public Fruit(String name, String variety, double totalQuantityKg, double pricePerKg) {
        this.name = name;
        this.variety = variety;
        this.totalQuantityKg = totalQuantityKg;
        this.pricePerKg = pricePerKg;
    }

    // Getters and setters
    public String getName() { return name; }
    public String getVariety() { return variety; }
    public double getTotalQuantityKg() { return totalQuantityKg; }
    public double getPricePerKg() { return pricePerKg; }
    public void setTotalQuantityKg(double totalQuantityKg) { this.totalQuantityKg = totalQuantityKg; }
}

// Seller class
class Seller extends User {
    private String address;
    private List<Fruit> fruits;

    public Seller(String firstName, String lastName, String phoneNumber, String address) {
        super(firstName, lastName, phoneNumber);
        this.address = address;
        this.fruits = new ArrayList<>();
    }

    public String getAddress() { return address; }
    public List<Fruit> getFruits() { return fruits; }
}

// Order class
class Order {
    private List<Fruit> fruits;
    private List<Double> quantityInKg;
    private Buyer buyer;
    private Seller seller;

    public Order(Buyer buyer, Seller seller) {
        this.buyer = buyer;
        this.seller = seller;
        this.fruits = new ArrayList<>();
        this.quantityInKg = new ArrayList<>();
    }

    public List<Fruit> getFruits() { return fruits; }
    public List<Double> getQuantityInKg() { return quantityInKg; }
}

// Buyer class
class Buyer extends User {
    private String address;
    private double rewards;
    private Order order;

    public Buyer(String firstName, String lastName, String phoneNumber, String address) {
        super(firstName, lastName, phoneNumber);
        this.address = address;
        this.rewards = 0.0;
    }

    public String getAddress() { return address; }
    public double getRewards() { return rewards; }
    public void setRewards(double rewards) { this.rewards = rewards; }

    public Order placeOrder(Seller seller) throws NoFruitsException {
        System.out.println("Available fruits from seller " + seller.getFirstName() + " " + seller.getLastName() + ":");

        for (Fruit fruit : seller.getFruits()) {
            System.out.println("Name: " + fruit.getName() + ", Variety: " + fruit.getVariety() + ", Price: " + fruit.getPricePerKg());
        }

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the name of the fruit to buy: ");
        String fruitName = scanner.nextLine();
        System.out.print("Enter quantity in kg: ");
        double quantity = scanner.nextDouble();

        Fruit selectedFruit = null;
        for (Fruit fruit : seller.getFruits()) {
            if (fruit.getName().equalsIgnoreCase(fruitName)) {
                selectedFruit = fruit;
                break;
            }
        }

        if (selectedFruit == null || selectedFruit.getTotalQuantityKg() < quantity) {
            throw new NoFruitsException("Selected fruit is not available in the required quantity.");
        }

        this.order = new Order(this, seller);
        order.getFruits().add(selectedFruit);
        order.getQuantityInKg().add(quantity);

        return order;
    }
}

// RoutePlanner interface
interface RoutePlanner {
    String calculateRoute(String startAddress, String endAddress);
}

// Delivery class
class Delivery {
    private RoutePlanner routePlanner;

    public Delivery(RoutePlanner routePlanner) {
        this.routePlanner = routePlanner;
    }

    public void deliver(Order order) {
        String route = routePlanner.calculateRoute(order.buyer.getAddress(), order.seller.getAddress());
        System.out.println("Route for delivery: " + route);

        double totalCost = 0.0;
        for (int i = 0; i < order.getFruits().size(); i++) {
            Fruit fruit = order.getFruits().get(i);
            double quantity = order.getQuantityInKg().get(i);
            totalCost += fruit.getPricePerKg() * quantity;
            fruit.setTotalQuantityKg(fruit.getTotalQuantityKg() - quantity);
        }

        double reward = totalCost * 0.01;
        order.buyer.setRewards(order.buyer.getRewards() + reward);

        System.out.println("Order delivered successfully! Buyer reward increased by: " + reward);
    }
}

// Custom NoFruitsException
class NoFruitsException extends Exception {
    public NoFruitsException(String message) {
        super(message);
    }
}
```