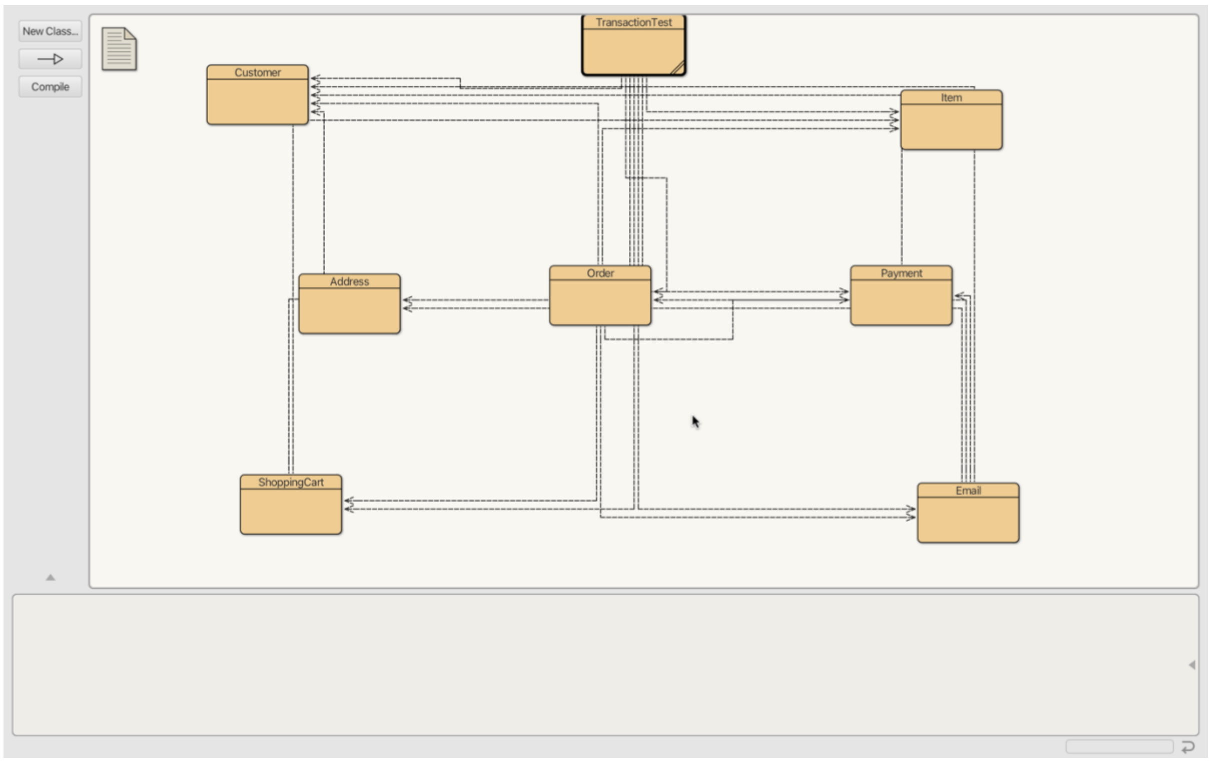
OOP Assignment 2

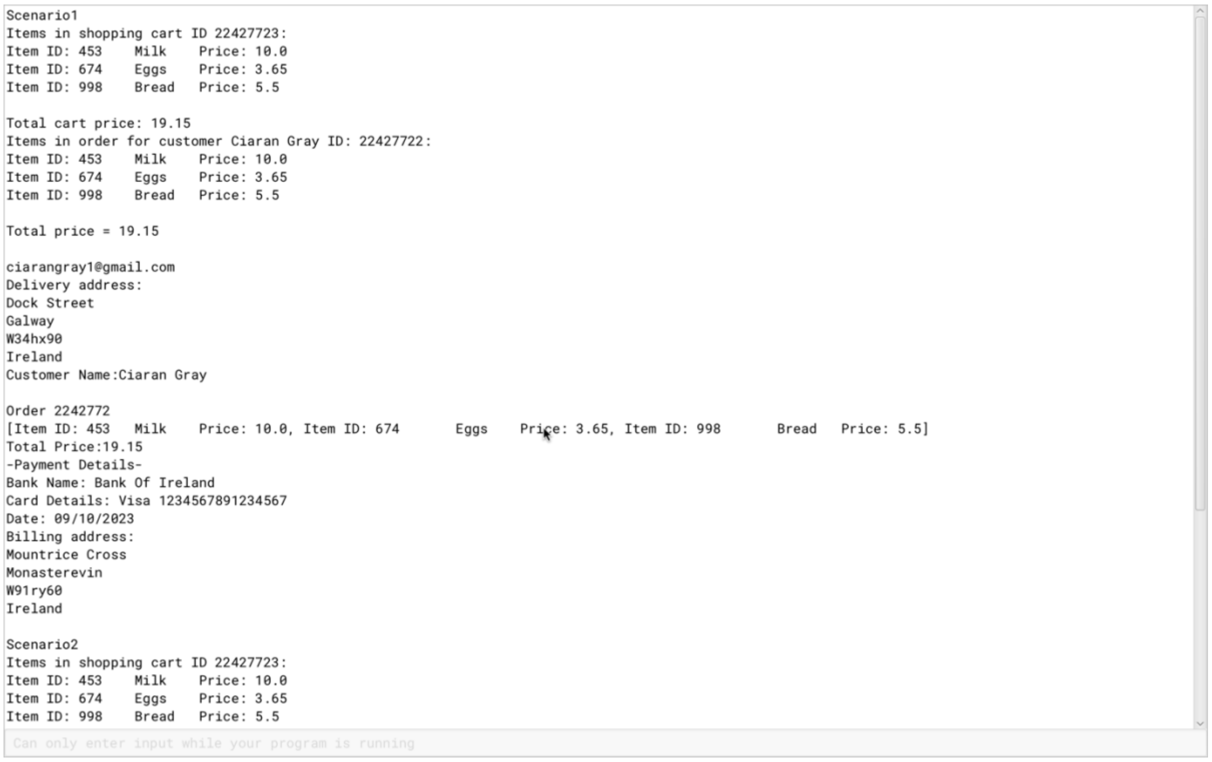
1.

For this assignment, I will use the following classes-
Customer,
Shopping cart,
Item,
Order,
Payment,
Address,
Email
And a testing class to perform the transactions.
First, I will create a customer class object which will have the first name, last name, email address and customer id fields, and an address class object, which will have the street, city, zip code and country fields. Then, I will create 3 items in the items class which will have the fields of item name, id, and price, and create a shopping cart object which will take in the 3 items and add them to the shopping cart array list by using the built in java add method. The order class will then take the items from the shopping cart and add them to the order, the order class will contain customer, shopping cart and payment constructors, when the order is created. I will then add a payment object with the class fields of card type, card number, bank and date, and confirm the payment details are correct in the payment class. If the payment details are right, the payment will then be validated using a Boolean method in the order class and once the order is finalised, the order class will call the printemail method in the email class that will print out information from the customer class, address class, payment class and order class, using the class constructors which will be initialised in the email class.

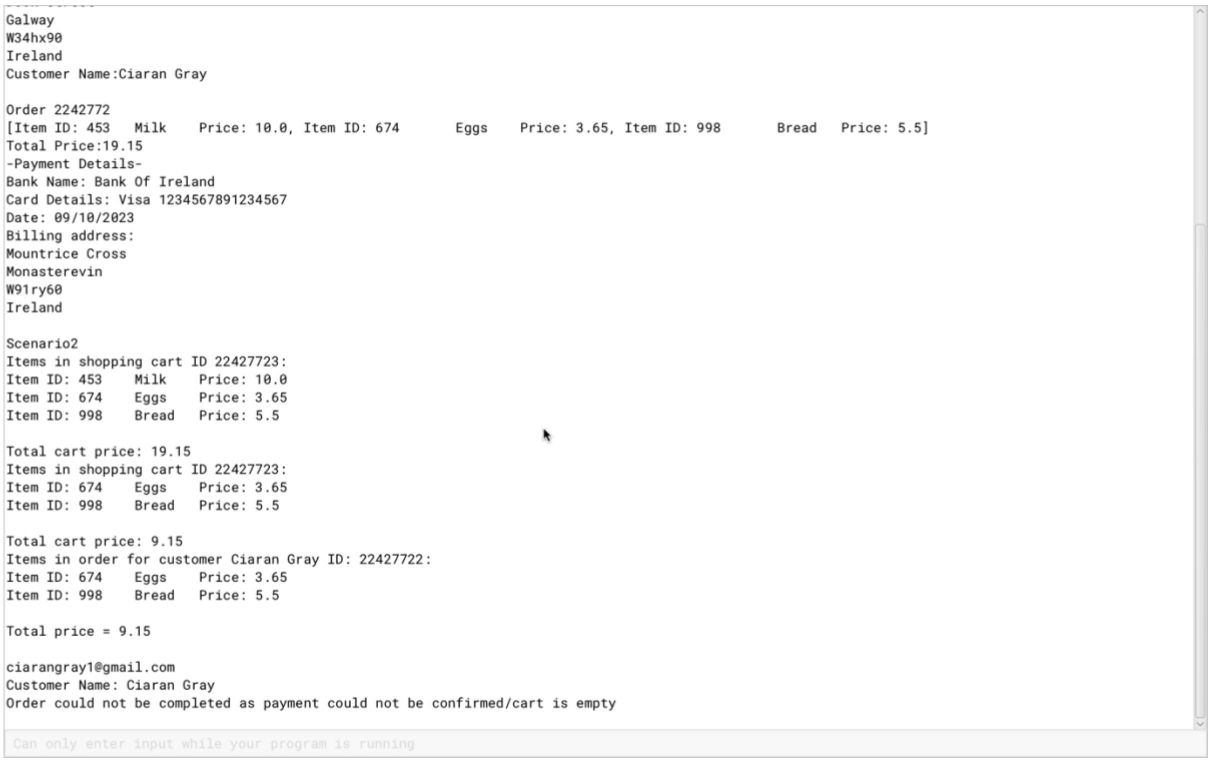
2.

```
public class TransactionTest
{
        public static void main(String[] args){
        TransactionTest test = new TransactionTest();
        test.scenario1();
        test.scenario2();
    }
    //this is the first scenario
    public void scenario1() {
        System.out.println("Scenario1");
        Customer customer = new Customer("Ciaran", "Gray", "ciarangray1@gmail.com",
22427722);

        //making a customer and billing and delivery address objects here
        Address daddress = new Address("Dock Street", "Galway", "W34hx90", "Ireland");
        Address baddress = new Address("Mountrice Cross", "Monasterevin", "W91ry60",
"Ireland");
        //the three item objects
        Item item1 = new Item("Milk", 10.00, 453);
        Item item2 = new Item("Eggs", 3.65, 674);
        Item item3 = new Item("Bread", 5.50, 998);

        //creating a shopping cart object while passing the customer object as a parameter and
adding the three items to the shopping cart
        ShoppingCart cart = new ShoppingCart(customer);
        cart.addToCart(item1);
        cart.addToCart(item2);
        cart.addToCart(item3);


        cart.displayCart();
        //creaing the order with customer sand shopping cart as the parameters
        Order order = new Order(customer, cart);
        order.addToOrder(cart);
        order.displayOrder();
        //email and payment objects
        Payment payment = new Payment(customer, "Visa", "1234567891234567",
"09/10/2023", "Bank Of Ireland");
        Email email = new Email(customer, order, payment, daddress, baddress);
        //if the payment is validated here in the order class, it then gets the email class to send
the email with all the info
        order.validateOrder(payment);
        order.confirmOrder(email);
    }
    //second scenario
    public void scenario2() {
        System.out.println("\nScenario2");
        //making a customer and billing and delivery address objects here
        Customer customer = new Customer("Ciaran", "Gray", "ciarangray1@gmail.com",
22427722);

        Address daddress = new Address("Dock Street", "Galway", "W34hx90", "Ireland");
        Address baddress = new Address("Mountrice", "Monasterevin", "W91ry60", "Ireland");
        //the three item objects
        Item item1 = new Item("Milk", 10.00, 453);
        Item item2 = new Item("Eggs", 3.65, 674);
        Item item3 = new Item("Bread", 5.50, 998);

        //creating a shopping cart object while passing the customer object as a parameter and
adding the three items to the shopping cart
        ShoppingCart cart = new ShoppingCart(customer);
        cart.addToCart(item1);
        cart.addToCart(item2);
        cart.addToCart(item3);

        cart.displayCart();
        //removing the first item from the cart
        cart.removeFromCart(item1);

        cart.displayCart();
        //creaing the order with customer sand shopping cart as the parameters
        Order order = new Order(customer, cart);
        order.addToOrder(cart);
        order.displayOrder();
        //email and payment objects
        Payment payment = new Payment(customer, "Viso", "1234567", "09/10/2023", "Bank
Of Ireland");
        Email email = new Email(customer, order, payment, daddress, baddress);
        //the payment won't be validated here as the payment details are incorrect, so
therefore an email of regret will be sent instead
        order.validateOrder(payment);
        order.confirmOrder(email);
    }

}


public class Customer
{
    private String fname;
    private String lname;
    private String email;
    private long custID;


    public Customer(String fname, String lname, String email, long custID)
    {
        this.fname = fname;
        this.lname = lname;
        this.email = email;
        this.custID = custID;
    }
    //getter methods for name, iD, and email
    public String getEmail() {
        return email;
    }

    public String getName() {
        return fname + " " + lname;
    }

    public long getID() {
        return custID;
    }

}


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

/**
 * ciaran gray
 */
public class ShoppingCart
{
    public ArrayList<Item> cartItems; //array that holds all the shopping cart items
    private boolean isLocked;
    private long cartID;
    private Customer customer;
    private float totalPrice;

    public ShoppingCart(Customer customer)
    {
        this.cartItems = new ArrayList<>();
        this.isLocked = false;
        this.cartID = customer.getID() + 1; //the cart id is linked to the customer ID
        this.totalPrice = totalPrice;
    }
    //getter and setter methods here

    public long getcartID() {
        return cartID;
    }

    public float getTotalPrice() {
        return totalPrice;
    }

    public void setTotalPrice() {
        this.totalPrice = totalPrice;
    }
    //if the cart isn't locked, it adds all the items that are passed through to the cart array and
adds up the total price of the cart by accessing the getprice method in the item class
    public void addToCart(Item item) {
        if (isLocked == false) {
            cartItems.add(item);
            totalPrice += item.getPrice();
        }
        else {
            System.out.println("Shopping Cart is Closed.");
        }
    }
    //works the same as the addToCart method but invertedly
        public void removeFromCart(Item item) {
        if (isLocked == false) {
            cartItems.remove(item);
            totalPrice -= item.getPrice();
        } else {
            System.out.println("Shopping Cart is Closed.");
        }
    }
    //locks cart by setting the boolean thats required to be false to perform actions on the
cart to true
    public void lockCart() {
        isLocked = true;
    }
    //clears the cart array and sets the total price to 0, also unlocks the cart
    public void clearCart() {
            cartItems.clear();
            totalPrice = 0;
            isLocked = false;
    }
    //prints the contents of the cart by looping through the items in the cart and calling the
toString method in the items class for each item in the cart
        public void displayCart() {
        System.out.println("Items in shopping cart ID " + cartID + ": ");
        for (Item item : cartItems) {
            System.out.println(item.toString());
        }
        System.out.println("\nTotal cart price: " + totalPrice);
    }

}


public class Item
{
    // instance variables
    private String name;
    private double price;
    private long itemID;
    public Item(String itemName, double price, long id)
    {
        this.name = itemName;
        this.price = price;
        this.itemID = id;
    }
    //some getter and setter methods
    public void setPrice(double price) {
        this.price = price;
    }


    public double getPrice()
    {
        return price;
    }
    //string that contains the item details
    @Override
        public String toString() {
        String iDetails = "Item ID: " + itemID + "\t" + name + "\tPrice: " + price;
        return iDetails;
    }

}


import java.util.ArrayList;
import java.util.List;
/**
 * ciaran gray
 */
public class Order
{
    // instance variables - replace the example below with your own
    private List<Item> orderItems;
    private float orderTotal;
    private Customer customer;
    private Payment payment;
    private ShoppingCart cart;
    private Email email;
    private boolean paymentConfirmed = false;
    private long orderNum;

    /**
     * Constructor for objects of class Order
     */
    public Order(Customer customer, ShoppingCart cart)
    {
        this.customer = customer;
        this.cart = cart;
        this.orderItems = new ArrayList<>();
        this.payment = payment;
        this.email = email;
        this.orderTotal = orderTotal;
        this.paymentConfirmed = false;
        this.orderNum = (customer.getID() / 10);
    }
    // loops through each item in the shopping cart array and adds the items individually into
the order items array and also adds up th total price from the cart to a new variable for the
order total, and calls the clear cart function at the end
    public void addToOrder(ShoppingCart cart)
    {
        for (Item item : cart.cartItems) {
            orderItems.add(item);
            orderTotal = cart.getTotalPrice();
        }
        cart.clearCart();
    }

    //displays the order by using the customer class for the customer's name and id and by
looping through the items in the order and calling the toString method in the items class for
each item in the order
        public void displayOrder() {
        System.out.println("Items in order for customer " + customer.getName() + " ID: "
+customer.getID() + ": ");
        for (Item item : orderItems) {
            System.out.println(item.toString());
        }
        System.out.println("\nTotal price = " + orderTotal);
    }
    //summary containing all the details of the order including the order, payment, order
number and order total price
    public String orderSummary(Payment payment) {
        return "\nOrder " + orderNum + "\n" + orderItems + "\nTotal Price:" + orderTotal +
payment.paymentDetails();
    }
    // method that checks if both the card type and card number are correct, if they are then
it sets a boolean to true which allows the email of the order summary to be set
    public void validateOrder(Payment payment) {
        if(payment.validateCardType() != "Invalid Card Type" &&
payment.validateCardNumber() != "Invalid Card Number") {
            paymentConfirmed = true;
        }
        else {
            paymentConfirmed = false;
        }
    }
    //if the order has been validated and the order actually contains at least 1 item, the oder
summary email is sent, or else an email of regret is sent
    public void confirmOrder(Email email) {
        if(paymentConfirmed == true && orderItems.size() != 0) {
            email.emailOrderSummary();
        }
        else {
            email.emailOfRegret();
        }
    }
}


public class Address
{
    private String street;
    private String city;
    private String zip;
    private String country;

    public Address(String l1, String l2, String l3, String country)
    {
        this.street = l1;
        this.city = l2;
        this.zip = l3;
        this.country = country;
    }
    //getter and setter methods
    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String City) {
        this.city = city;
    }
    public String getZip() {
        return zip;
    }

    public void setZip(String zip) {
        this.zip = zip;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }
    // String that contains all the address details together
    public String getAddress()
    {
        return "\n" + street + "\n" + city + "\n" + zip + "\n" + country;
    }
}

public class Payment
{
    private Customer customer;
    private String cardType;
    private String cardNumber;
    private String date;
    private String bankName;

    public Payment(Customer customer, String cardType, String cardNumber, String date,
String bankName)
    {
        this.customer = customer;
        this.cardType = cardType;
        this.cardNumber = cardNumber;
        this.date = date;
        this.bankName = bankName;
    }

    // method for validating the card type, should be Masercard or Visa
    public String validateCardType() {
        if (this.cardType.equals("MasterCard") || cardType.equals("Visa")) {
            return cardType;
        }
        else
        {
            return "Invalid Card Type";
        }
    }

    //Validating the card number by checking if it contains the right ammount of digits
    public String validateCardNumber() {
        if (this.cardNumber.length() == 16) {
            return cardNumber;
        } else {
            return "Invalid Card Number";
        }
    }

    //getter methods

    public String getCardType() {
        return cardType;
    }

    public String getCardNumber() {
        return cardNumber;
    }

    public String getDate() {
        return date;
    }

    public String getBankName() {
        return bankName;
    }
    //string that contains all the payment details
    public String paymentDetails() {
        return "\n-Payment Details-\nBank Name: " + bankName + "\nCard Details: " +
cardType + " " + cardNumber + "\nDate: " + date;
    }
}


public class Email
{
    private Customer customer;
    private String custEmailaddress;
    private String ordersum;
    private Order order;
    private Address daddress;
    private Address baddress;
    private Payment payment;
    //constructors
    public Email(Customer customer, Order order, Payment payment, Address daddress,
Address baddress)
    {
        this.customer = customer;
        this.payment = payment;
        this.daddress = daddress;
        this.baddress = baddress;
        this.custEmailaddress = customer.getEmail();//gets the customers email from the email
class
        this.ordersum = order.orderSummary(payment);//transfers the order summary from
the order class to email class and stores as a string variable

    }
    //getter and setters
    public String getOrderSum() {
        return ordersum;
    }

    public void setOrderSum() {
        this.ordersum = ordersum;
    }


    public void emailOrderSummary()
    {
        System.out.println("\n" + custEmailaddress + "\nDelivery address:" +
daddress.getAddress() + "\nCustomer Name:" + customer.getName() + "\n" + ordersum +
"\nBilling address: " + baddress.getAddress());
    }
    //both of these emails print out either an order summary or an email of regret depending
on which method is called in the order class
    public void emailOfRegret() {
        System.out.println("\n" + custEmailaddress + "\nCustomer Name: " +
customer.getName() + "\nOrder could not be completed as payment could not be
confirmed/cart is empty");
    }
}
```





```
Scenario1
Items in shopping cart ID 22427723:
Item ID: 453    Milk    Price: 10.0
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5

Total cart price: 19.15
Items in order for customer Ciaran Gray ID: 22427722:
Item ID: 453    Milk    Price: 10.0
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5

Total price = 19.15

ciarangray1@gmail.com
Delivery address:
Dock Street
Galway
W34hx90
Ireland
Customer Name:Ciaran Gray

Order 2242772
[Item ID: 453   Milk    Price: 10.0, Item ID: 674       Eggs    Price: 3.65, Item ID: 998       Bread   Price: 5.5]
Total Price:19.15
-Payment Details-
Bank Name: Bank Of Ireland
Card Details: Visa 1234567891234567
Date: 09/10/2023
Billing address:
Mountrice Cross
Monasterevin
W91ry60
Ireland

Scenario2
Items in shopping cart ID 22427723:
Item ID: 453    Milk    Price: 10.0
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5
Can only enter input while your program is running
```



```
Galway
W34hx90
Ireland
Customer Name:Ciaran Gray

Order 2242772
[Item ID: 453   Milk    Price: 10.0, Item ID: 674       Eggs    Price: 3.65, Item ID: 998       Bread   Price: 5.5]
Total Price:19.15
-Payment Details-
Bank Name: Bank Of Ireland
Card Details: Visa 1234567891234567
Date: 09/10/2023
Billing address:
Mountrice Cross
Monasterevin
W91ry60
Ireland

Scenario2
Items in shopping cart ID 22427723:
Item ID: 453    Milk    Price: 10.0
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5

Total cart price: 19.15
Items in shopping cart ID 22427723:
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5

Total cart price: 9.15
Items in order for customer Ciaran Gray ID: 22427722:
Item ID: 674    Eggs    Price: 3.65
Item ID: 998    Bread   Price: 5.5

Total price = 9.15

ciarangray1@gmail.com
Customer Name: Ciaran Gray
Order could not be completed as payment could not be confirmed/cart is empty
```

3. the code runs as expected here, I had different visions for how the methods would be separated around the classes but ultimately found that this version of the code was most effective as I could make it and runs as I expect it to.