COEN 6312: E-Commerce System

Deliverable 3

Submitted To:

Prof. Abdelwahab Hamou-Lhadj

Submitted By:

Group 10

Hephzibah Pocharam - 40127128

Nikhil Verma - 40160264

Chirag Jhamb - 40169876

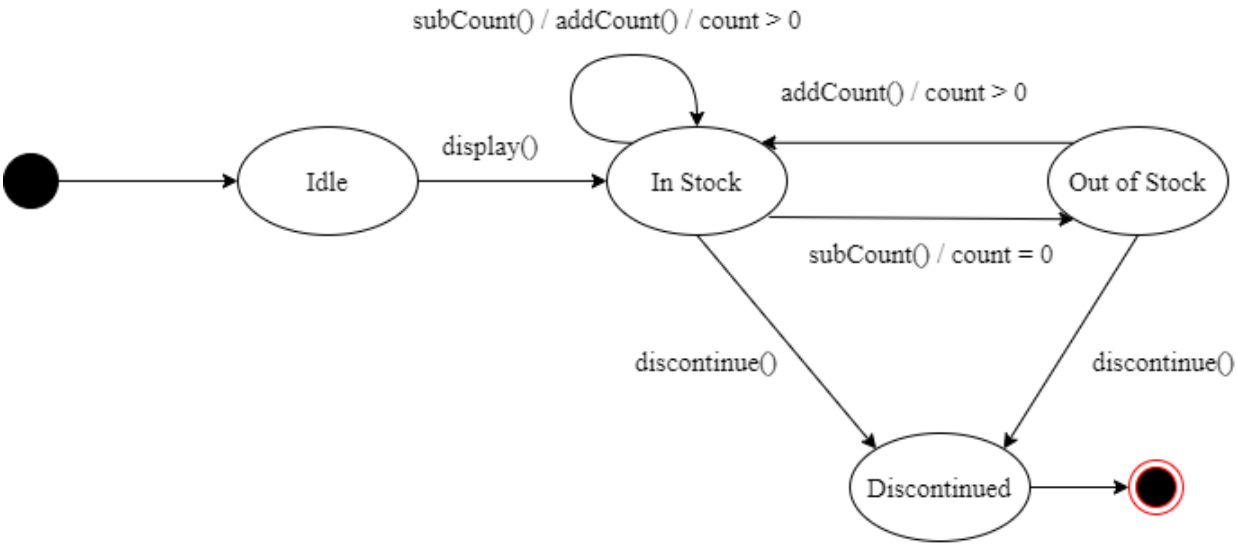
Divyaa Mahalakshmi Guruswamy- 40167923

Jayapriya Muthuramasamy - 40184587

Abdul Rahman Koleilat – 40086025

TABLE OF CONTENTS

PRODUCT CLASS STATE DIAGRAM



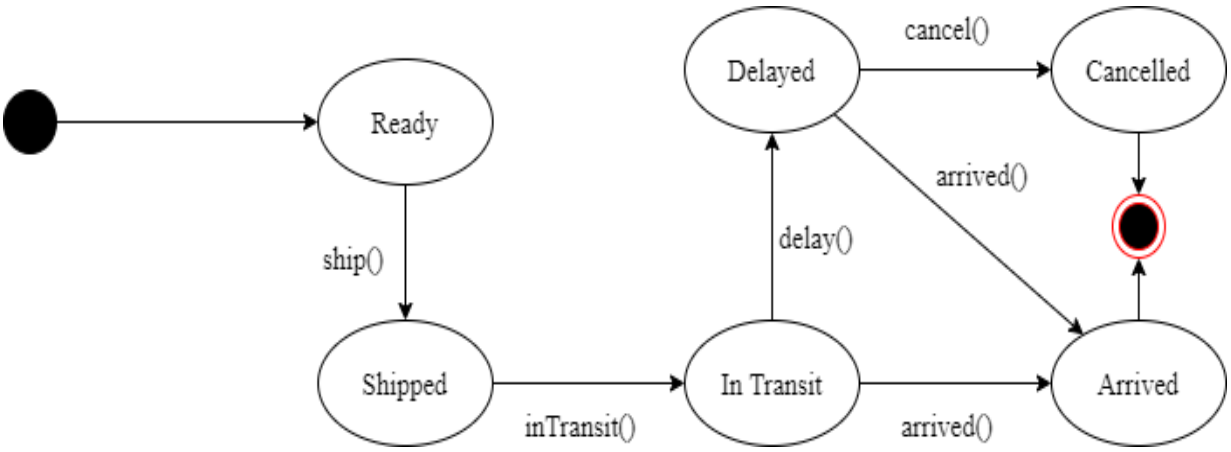
The" Discontinued" state means that the product is no longer available in the market.

Implementation of Product Class state diagram

```
public class Product {

    private static int counter;
    public int id;
    public String name;
    public float price;
    public int count;
    public String description;
    public Category category;
    public Retailer retailer;
    public float averageRating;
    public ArrayList<Review> reviews = new ArrayList<Review>();
    public states state;

    //This is where the product class state diagram is implemented. The following are the states.
    enum states {
        IDLE,
        INSTOCK,
        OUTOFSTOCK,
        DISCONTINUED
    }

    public Product(String name, float price, int count, String description, Category category, Retailer retailer) {
        super();
        Product.counter++;
        this.id = counter;
        this.name = name;
        this.price = price;
        this.count = count;
        this.description = description;
        this.category = category;
```

```
    this.retailer = retailer;
    category.addProduct(this);
    this.averageRating = 0;
    this.state = states.IDLE;
}

//The methods display(), addCount(), subCount, discontinued() are the events of the state diagram that will cause the transitions.

public void display() {
    this.state = states.INSTOCK;
}

public void addCount(int n) {
    count += n;
    state = states.INSTOCK;
}

public void discontinue() {
    state = states.DISCONTINUED;
}

public void subCount(int n) {
    count -= n;
    if(count == 0) {
        state = states.OUTOFSTOCK;
    }
}
```

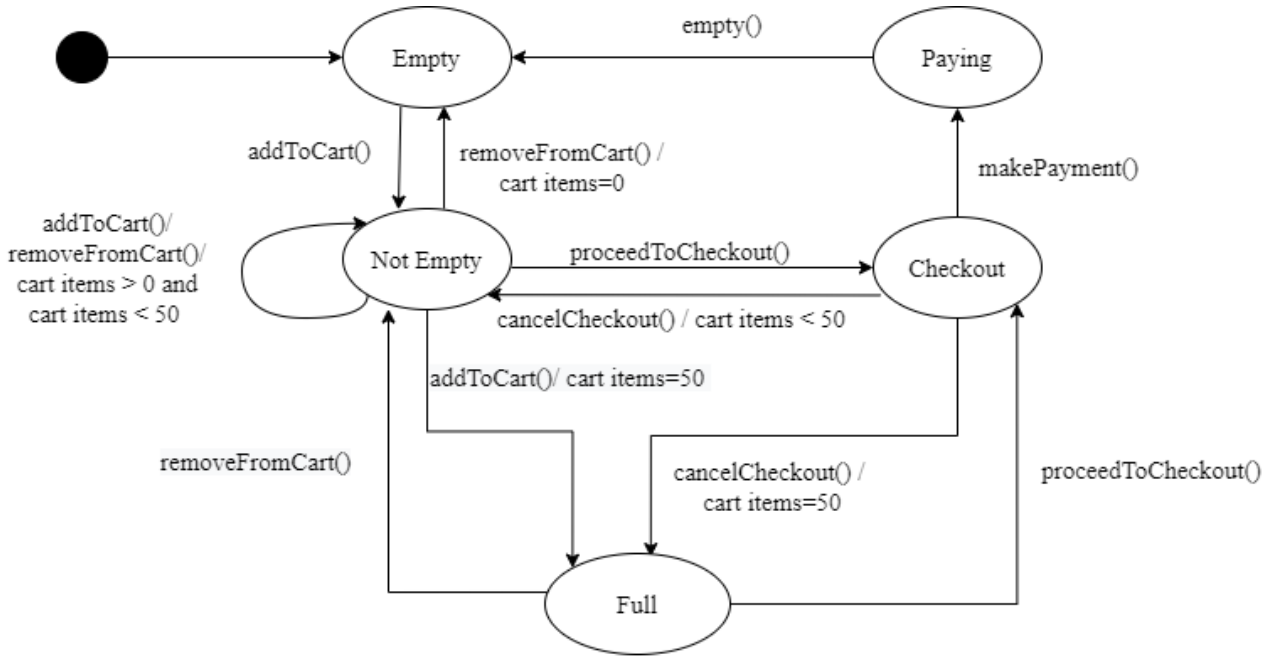
DELIVERY CLASS STATE DIAGRAM



Implementation of Delivery Class State Diagram

```
public class Delivery {

    public int trackingNumber;
    public static int counter;
    public Address address;
    public LocalDateTime dateOfArrival;
    public LocalDateTime expectedDateOfArrival;
    public String liveLocation;
    public int maxDays = 15;
    public static boolean override;
    public states state;

    //This is where the delivery class state diagram is implemented. The following are the states.
    enum states {
        READY,
        SHIPPED,
        INTRANSIT,
        DELAYED,
        CANCELLED,
        ARRIVED
    }

    public Delivery(Address address) {
        super();
        Delivery.counter++;
        this.trackingNumber = counter;
        this.address = address;
        this.expectedDateOfArrival = LocalDateTime.now().plusDays(maxDays);
        this.liveLocation = "NA";
        this.state = states.READY;
    }

//The methods ship(), inTransit(), delay(), cancel(), arrived()  are the events

public void ship() {
    state = states.SHIPPED;
}

public void inTransit() {
    state = states.INTRANSIT;
}

public void delay() {
    state = states.DELAYED;
}

public void cancel() {
    state = states.CANCELLED;
}

public void arrived() {
    state = states.ARRIVED;
    this.dateOfArrival = LocalDateTime.now();
}
```

CART CLASS STATE DIAGRAM



As you can see, the state diagram for the cart class doesn't have a termination because the same cart is used by the customer regardless of what state it's in.

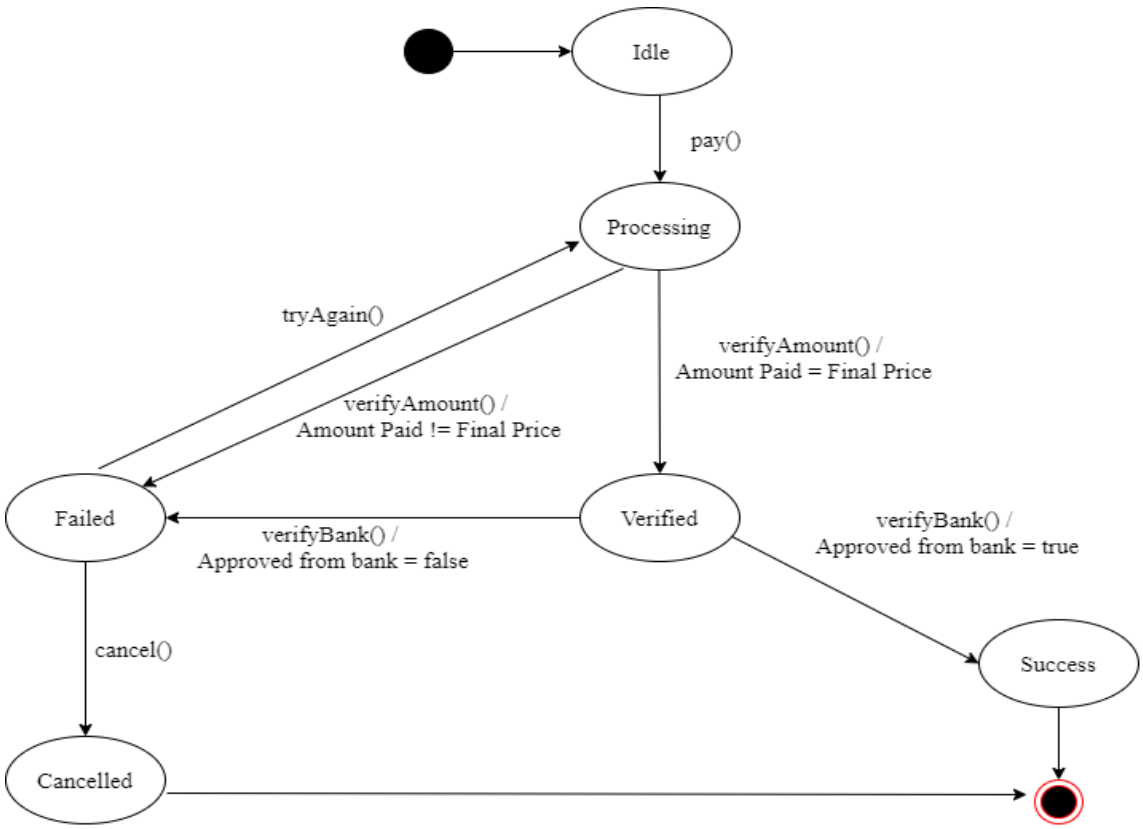
Implementation of Cart Class State Diagram

```
public class Cart {

    public ArrayList<CartItem> cartItems;
    public Customer customer;
    public Payment payment;
    public float totalPrice;
    public static float taxRate = 5;
    public float deliveryCharge;
    public float finalPrice;
    public int totalCount;
    public states state;
    public states prevState;

    //This is where the cart class state diagram is implemented. The following are the states.
    enum states {
        EMPTY,
        NOTEMPTY,
        FULL,
        CHECKOUT,
        PAYING
    }

    public Cart(Customer customer) {
        super();
        this.cartItems = new ArrayList<CartItem>();
        this.totalPrice = 0;
        this.customer = customer;
        this.state = states.EMPTY;
    }
```

```
public void addToCart(Product p) {
    for(int i = 0; i < cartItems.size(); i++) {
        if(cartItems.get(i).p == p) {
            cartItems.get(i).count++;
            totalCount++;
            calculateTotalPrice();
            return;
        }
    }
    cartItems.add(new CartItem(p, this));
    totalCount++;
    calculateTotalPrice(); //update total price
    if(state == states.EMPTY) {
        state = states.NOTEMPTY;
    }

    if(totalCount == 50) {
        state = states.FULL;
    }
}
public void removeFromCart(CartItem c, int count) {
    for(int i = 0; i < cartItems.size(); i++) {
        if(cartItems.get(i) == c) {
            if(cartItems.get(i).count == 1) {
                cartItems.remove(i);
                totalCount--;
            } else {
                cartItems.get(i).count-= count;
                totalCount-=count;
            }
            calculateTotalPrice(); //update total price
            return;
        }
    }

    if(state == states.FULL) {
        state = states.NOTEMPTY;
    }
}
```

```
    public void proceedToCheckout() throws Exception {

//      6. The customer won't be able to proceed to checkout if the cart is empty.
//      Context cart
//      Inv: allInstances() -> forAll(c: Cart | c.state = "Checkout" implies c.CartItem -> isnotEmpty())

        if(cartItems.size() == 0) {
            throw new Exception("Can't proceed to checkout. The cart is empty");
        }
        prevState = state;
        state = states.CHECKOUT;
    }

    public void cancelCheckout() {
        state = prevState;
    }

    public void makePayment(String paymentMethod, String paymentInfo) {

        float finalPrice = totalPrice* (taxRate/100 + 1) + calculateDeliveryCharge();

        payment = new Payment(paymentMethod, paymentInfo, finalPrice, this);

    }

    public void empty() {
        cartItems.clear();
        this.totalPrice = 0;
        this.totalCount = 0;
        this.state = states.EMPTY;
    }
```

PAYMENT CLASS STATE DIAGRAM



Implementation of Payment Class State Diagram

```
public class Payment extends Transaction {

    public Cart cart;
    public float finalPrice;
    public float processedAmount; //amount to be verified
    public states state;
    public static boolean bankApproved;

    //This is where the payment class state diagram is implemented. The following are the states.
    enum states {
        IDLE,
        PROCESSING,
        VERIFIED,
        CANCELLED,
        FAILED,
        SUCCESS
    }

    public Payment(String paymentMethod, String paymentInfo, float finalPrice, Cart cart) {
        super(paymentMethod, paymentInfo);
        state = states.IDLE;
        this.finalPrice = finalPrice;
        this.cart = cart;
    }



    //The methods pay(), verifyAmount(), verifyBank(), tryAgain(), cancel() are the events

    public void pay(float amount) throws Exception{
        state = states.PROCESSING;
        processedAmount = amount;
    }

    public void verifyAmount() {
        if(processedAmount == finalPrice) {
            state = states.VERIFIED;
            amount = processedAmount;
        }
        else {
            state = states.FAILED;
        }

    }

    public void verifyBank() throws Exception{
        if(state != states.VERIFIED) {
            throw new Exception("Payment isn't verified");
        }

        if(bankApproved) {
            state = states.SUCCESS;
        }
        else {
            state = states.FAILED;
        }
    }
```

```
public void tryAgain() throws Exception{
    if(state != states.FAILED) {
        throw new Exception("Can't try again at this time.");
    }

    state = states.PROCESSING;
}

public void cancel() throws Exception{
    if(state != states.FAILED) {
        throw new Exception("Can't cancel payment at this time.");
    }

    state = states.CANCELLED;
}
```

REFERENCES

[1] Class lectures of COEN 6312 by Prof. Abdelwahab Hamou-Lhadj

[2[Sommerville, I. (2016). Software engineering