Assignment 2 - Object Oriented Programming

Daniel Hannon (19484286)

November 2020

1 Overview

Within this assignment we have to Define a simple Online Shopping facility as described by the Diagram featured in Lecture 8, among others.
The function of my Code is fairly straight forward. In order to implement the Shopping Cart I used ArrayList to hold the items as it is more efficient than using a standard array for it has a seach speed of O n comparred to O n squared.
For the creation of the various IDs throughout the code I created a private method which utilizes Math.random() as Math.random only returns values between zero and one, I multiply it by an arbitrary number before typecasting the result to long
The transaction ID is accompanied by a timestamp which is stored as a string both of which cannot be modified by the end user whatsoever.
The Order Class is slightly more complex but still very straightforward. One piece of oversight I made was the ability to make it without the existence of an Address, this is to mimic real life transactions.
As a result there is a check before the email is sent to verify the existence of an Address in some capacity.
The Address is a dynamic object as it allows for varying amounts of fields, much like a normal address. I created a method Stringify() to merge the address into a formatted string and return it in such a way that it looks nice to the user, this also reduces the amount of get calls that need to be made, hence reducing overall lines of code.
Payment records the credit card details and the order details and it validates itself in two ways. It verifies the date is a date and in the correct format, if it is not. Uses the unix Epoch as the substitute date, so it will always fail validation. then it verifies the card type by doing some string compares, and finally it verifies that the number is of adequate length by checking if it is between a certain numeric range, as it is 16-digits, it must be stored as a Long as it is out of the range of the Integer type.

2 Code

2.1 TransactionTest

```java
public class TestTransaction  {
        public TestTransaction(){
                System.out.println("Test Transaction created!");
        };
        public static void main(String[] args) {
                TestTransaction one = new TestTransaction();
                System.out.println("Test Transaction 1");
                one.transaction1();
                System.out.println("******************");
                System.out.println("Test Transaction 2");
                one.transaction2();
        }
        public void transaction1() {
                Customer customer = new
                 ↪ Customer("Robert","Paulson","t.durden1998@email.com");
                ShoppingCart cart = new ShoppingCart();
                cart.addItem(new Item("Soap",3.24f));
                cart.addItem(new Item("Spray Paint",5.50f));
                cart.addItem(new Item("Ikea Furniture",19.99f));
                cart.listShopping();
                cart.close();
                Order order = new Order(customer,cart);
                order.processOrder();
                order.addAddress(new Address("34","Memory Lane","Churchill Road",
                 ↪ "F42XYXY","Croydon"));
                Payment payment = new Payment(order,"Visa",4319000000000000L,"12/21");
                payment.validate();
                payment.Confirm();

        }
        public void transaction2() {
                Customer customer = new Customer("Luigi","Mario","Luigi@mariobros.com");
                ShoppingCart cart = new ShoppingCart();
                cart.addItem(new Item("Plumbers hat",23.45f));
                cart.addItem(new Item("Powerup",19.86f));
                cart.addItem(new Item("Movie",19.93f));
                cart.removeItem(new Item("Star",19.93f));
                cart.close();
                Order order = new Order(customer,cart);
                order.processOrder();
                order.addAddress(new Address("Mushroom Kingdom","Street","Mario Luigi","Mario"));
                Payment myPayment = new Payment(order, "Pizza", 4319999999999999L, "11/12");
                myPayment.validate();
                myPayment.Confirm();

        }
}
```

2.2 ShoppingCart

```java
import java.util.ArrayList;
public class ShoppingCart {
        private ArrayList<Item> Cart;
        private float TotalCost;
        private boolean locked;
        private String time;
        private final long cartId;

        /*Initilaisation*/
        public ShoppingCart() {
                this.TotalCost = 0;
                this.locked = false;
                this.Cart = new ArrayList<Item>();
                /*The time in which the shopping cart is created depends on the time it is called,
                the customer should have no ability to modify this whatsoever */
                this.time = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm").format(new java.util.Date());
                this.cartId = makeCartID();
        }

        /*Locking Mechanism*/
        public void close() {
                this.locked = true;
        }

        public void UnlockCart() {
                this.locked = false;
        }

        /*ShoppingCart Interractions*/
        public void addItem(Item newItem) {
```

```java
		if (!this.locked) {
			this.Cart.add(newItem);
			this.TotalCost += newItem.getCost();
			System.out.println("Item: "+newItem.getName()+" Has been added to your cart!");
		} else {
			System.out.println("Sorry the shopping cart is closed");
		}
	}

	public void removeItem(Item removItem) {
		if (!this.locked) {
			this.Cart.remove(removItem);
			this.TotalCost -= removItem.getCost();
			System.out.println("Item: " + removItem.getName() + " Has Been removed!");
		} else {
			System.out.println("Cannot remove item, cart is locked");
		}
	}

	public Item takeItem(int index) {
		if(this.Cart.size() <= index) {
			System.out.println("Item Does not exist");
			return null;
		} else {
			Item temp = this.Cart.get(index);
			this.Cart.remove(index);
			this.TotalCost -= temp.getCost();
			return temp;
		}
	}

	public int getCartSize() {
		return this.Cart.size();
	}

	public void listShopping() {
		System.out.println("|\tItem\t\t|\t\tCost\t|");
		for (int i = 0; i < this.Cart.size();i++) {
			Item temp = this.Cart.get(i);
			System.out.println("|\t"+temp.getName()+"\t|\t"+temp.getCost()+
			"\t|");
		}
		System.out.println("|\tTotal:\t\t|\t\t"+TotalCost+"\t|\n");
	}

	/*Getters*/
	public float getTotalCost() {
		return this.TotalCost;
	}
	public String getTime() {
		return this.time;
	}

	public long getCartId() {
		return this.cartId;
	}

	/*Internal methods*/
```

```
        private long makeCartID() {
                return (long) (Math.random() * 1000000);
        }
}
```

2.3 Order

```
import java.util.*;
public class Order {
        private ArrayList<Item> OrderItems;
        private Customer customer;
        private ShoppingCart shoppingCart;
        private Address billingAddress;
        private Address deliveryAddress;
        private long orderNo;
        private boolean isConfirmed;
        private float totalCost;

        /*Billing/delivery have same address*/
        Order(Customer customer, ShoppingCart Cart) {
                this.OrderItems = new ArrayList<Item>();
                this.customer= customer;
                this.shoppingCart = Cart;
                this.orderNo = shoppingCart.getCartId();
                this.isConfirmed = false;
                this.totalCost = 0;
        }

        Order(Customer customer, ShoppingCart Cart, Address address) {
                this.OrderItems = new ArrayList<Item>();
                this.customer= customer;
                this.shoppingCart = Cart;
                this.deliveryAddress = address;
                this.billingAddress = address;
                this.orderNo = shoppingCart.getCartId();
                this.isConfirmed = false;
                this.totalCost = 0;
        }
        /*Billing/delivery have different addresses*/
        Order(Customer customer, ShoppingCart Cart, Address deliveryAddress, Address
         ↪  billingAddress) {
                this.OrderItems = new ArrayList<Item>();
                this.customer= customer;
                this.shoppingCart = Cart;
                this.deliveryAddress = deliveryAddress;
                this.billingAddress = billingAddress;
                this.orderNo = shoppingCart.getCartId();
                this.isConfirmed = false;
                this.totalCost = 0;
        }

        private String stringifyOrder() {
                if (this.OrderItems.size() == 0) {
                        return "";
                }
                String output = "";
                for(int i = 0;i < OrderItems.size(); i++) {
```

```java
                        output += OrderItems.get(i).toString() + "\n";
                }
                return output;
        }

        public void processOrder() {
                this.totalCost = shoppingCart.getTotalCost();
                int length = shoppingCart.getCartSize();
                for (int i = 0; i < length; i++) {
                        Item temp = shoppingCart.takeItem(0);
                        if (temp == null) {
                                break;
                        }
                        this.OrderItems.add(temp);
                }
        }

        public void confirmOrder() {
                if(this.OrderItems.size() == 0) {
                        processOrder();
                        if(this.OrderItems.size() == 0) {
                                return;
                        }
                }
                this.isConfirmed = true;
        }

        public void sendEmail() {
                Email email = new
                 ↪  Email(customer.getEmailAddress(),customer.getFirstName());
                if(isConfirmed) {
                        if (this.deliveryAddress!= null && this.billingAddress != null) {
                                email.SendSuccess(this.orderNo,this.stringifyOrder(),
                                totalCost,this.deliveryAddress,this.billingAddress);
                        } else {
                                System.out.println("You forgot to add an address!");
                        }
                }
                else {
                        email.SendFailure();
                }
        }

        public long getOrderNumber() {
                return this.orderNo;
        }

        public boolean getStatus() {
                return this.isConfirmed;
        }

        public void addAddress(Address address) {
                this.deliveryAddress = address;
                this.billingAddress = address;
        }

        public void addAddress(Address delivery, Address billing) {
                this.deliveryAddress = delivery;
                this.billingAddress = billing;
        }
```

```
}
```

2.4 Address

```
public class Address {
        private String HouseNumber;
        private String StreetName;
        private String City;
        /*Set to string to allow for British and Irish Postcodes*/
        private String ZipCode;
        private String Country;

        Address(String StreetName, String City, String ZipCode, String Country) {
                this.HouseNumber = "";
                this.StreetName = StreetName;
                this.City = City;
                this.ZipCode = ZipCode;
                this.Country = Country;
        }

        Address(String HouseNumber, String StreetName, String City, String ZipCode, String
        ↪   Country) {
                this.HouseNumber = HouseNumber;
                this.City = City;
                this.StreetName = StreetName;
                this.ZipCode = ZipCode;
                this.Country = Country;
        }

        public String stringify() {
                String output = "";
                if (HouseNumber.length()!= 0) {
                        output+=HouseNumber+"\n";
                }
                output+=StreetName+"\n";
                output+=City+"\n";
                output+=ZipCode+"\n";
                output+=Country+"\n";
                return output;
        }

        public String getHouseNumber() {
                return this.HouseNumber;
        }

        public String getStreetName() {
                return this.StreetName;
        }

        public String getCity() {
                return this.City;
        }

        public String getZipCode() {
                return this.ZipCode;
        }

```

```java
	public String getCountry() {
		return this.Country;
	}

	public void setHouseNumber(String HouseNumber) {
		this.HouseNumber = HouseNumber;
	}

	public void setStreetName(String StreetName) {
		this.StreetName = StreetName;
	}

	public void setCity(String City) {
		this.City = City;
	}

	public void setZipCode(String Zip) {
		this.ZipCode = Zip;
	}

	public void setCountry(String Country) {
		this.Country = Country;
	}

}
```

2.5 Payment

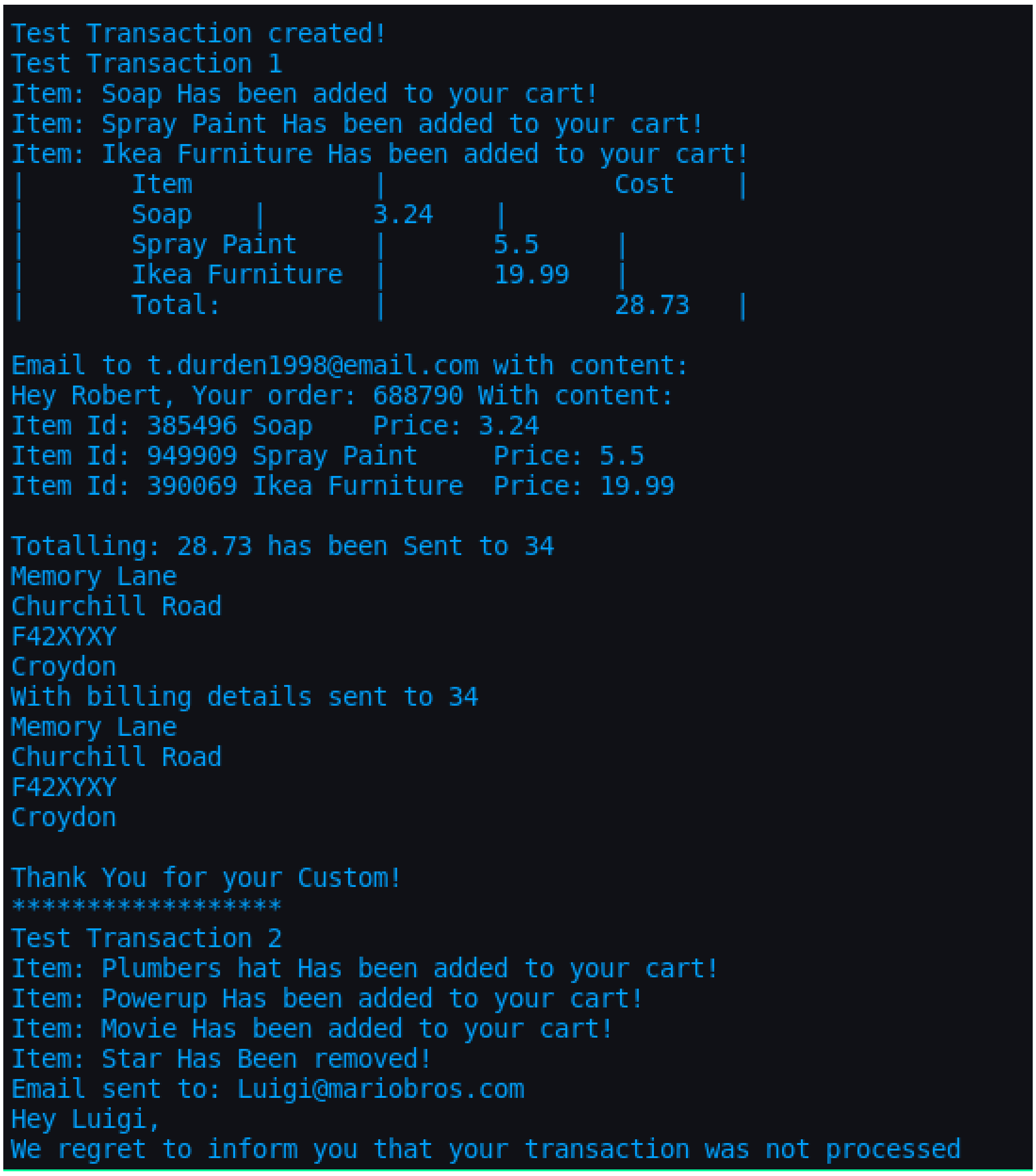
```java
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.text.SimpleDateFormat;

public class Payment {
	private long cardNumber;
	private Date date;
	private String cardType;
	private Boolean isValid;
	private Order order;

	Payment(Order order, String cardType, long cardNumber, String Date) {
		this.order = order;
		this.cardType = cardType;
		this.cardNumber = cardNumber;
		this.isValid = false;
		verifyDate(Date);
	}

	private void verifyDate(String date) {
		try {
			this.date = new SimpleDateFormat("MM/yy").parse(date);
		} catch (Exception e) {
			/*Set to UNIX epoch if the date could not be processed solely
			for validation purposes*/
			this.date = new
			↪ GregorianCalendar(1970,Calendar.JANUARY,1).getTime();
		}
```

```java
        }

        public void validate() {
                if(this.cardType == "Visa" || this.cardType == "MasterCard") {
                        if (cardNumber >= 1000000000000000L && cardNumber <=
                         ↪  9999999999999999L) {
                                Date today = new Date();
                                if(today.compareTo(date) < 0) {
                                        this.isValid = true;
                                        return;
                                }
                        }
                }
                this.isValid = false;
        }

        public void Confirm() {
                if(this.isValid) {
                        order.confirmOrder();
                        order.sendEmail();
                }
                else {
                        /*Just to check if it wasn't validated*/
                        this.validate();
                        if(this.isValid) {
                                order.confirmOrder();
                                order.sendEmail();
                        }
                        else {
                                order.sendEmail();
                        }
                }
        }

        public void setDate(String Date) {
                verifyDate(Date);
        }

        public void setCardNumber(long CardNumber) {
                this.cardNumber = CardNumber;
        }

        public void setCardType(String cardType) {
                this.cardType = cardType;
        }
}
```

2.6 Email

```java
public class Email {
        /*Test class for email as I do not have to implement actual email*/
        private String emailAddress;
        private String fname;
        public Email(String emailAddress, String fname) {
                this.emailAddress = emailAddress;
                this.fname = fname;
        }
        public void SendSuccess(long orderno, String content, float tcost, Address address,
            Address bAddress) {
                try {
                        String temp1 = address.stringify();
                        String temp2 = bAddress.stringify();
                        System.out.println("Email to "+ emailAddress + " with
                            content:\n"+"Hey "+fname+", Your order: "+orderno+" With
                            content:\n" + content + "\nTotalling: "+tcost+" has been Sent
                            to "+ temp1 +"With billing details sent to " + temp2 + "\nThank
                            You for your Custom!");
                } catch (Exception e) {
                        System.out.println("Something went wrong!");
                        try {
                                System.out.println(content);
                                System.out.println(tcost);
                                System.out.println(address.stringify());
                                System.out.println(bAddress.stringify());
                        }catch (Exception a) {
                                System.out.println("Whoops!");
                        }
                }
        }

        public void SendFailure() {
                System.out.println("Email sent to: " + emailAddress+ "\nHey "+fname+",\nWe
                    regret to inform you that your transaction was not processed");
        }
}
```

3 Output


```
Test Transaction created!
Test Transaction 1
Item: Soap Has been added to your cart!
Item: Spray Paint Has been added to your cart!
Item: Ikea Furniture Has been added to your cart!
|       Item            |               Cost    |
|       Soap    |       3.24    |
|       Spray Paint     |       5.5     |
|       Ikea Furniture  |       19.99   |
|       Total:          |               28.73   |

Email to t.durden1998@email.com with content:
Hey Robert, Your order: 688790 With content:
Item Id: 385496 Soap    Price: 3.24
Item Id: 949909 Spray Paint     Price: 5.5
Item Id: 390069 Ikea Furniture  Price: 19.99

Totalling: 28.73 has been Sent to 34
Memory Lane
Churchill Road
F42XYXY
Croydon
With billing details sent to 34
Memory Lane
Churchill Road
F42XYXY
Croydon

Thank You for your Custom!
******************
Test Transaction 2
Item: Plumbers hat Has been added to your cart!
Item: Powerup Has been added to your cart!
Item: Movie Has been added to your cart!
Item: Star Has Been removed!
Email sent to: Luigi@mariobros.com
Hey Luigi,
We regret to inform you that your transaction was not processed
```