Group 9: Xander Davis, Cole Phillips, Claudia Pinou
MIST4600: Computer Programming in Business
Dr. Aguar
17 November 2020

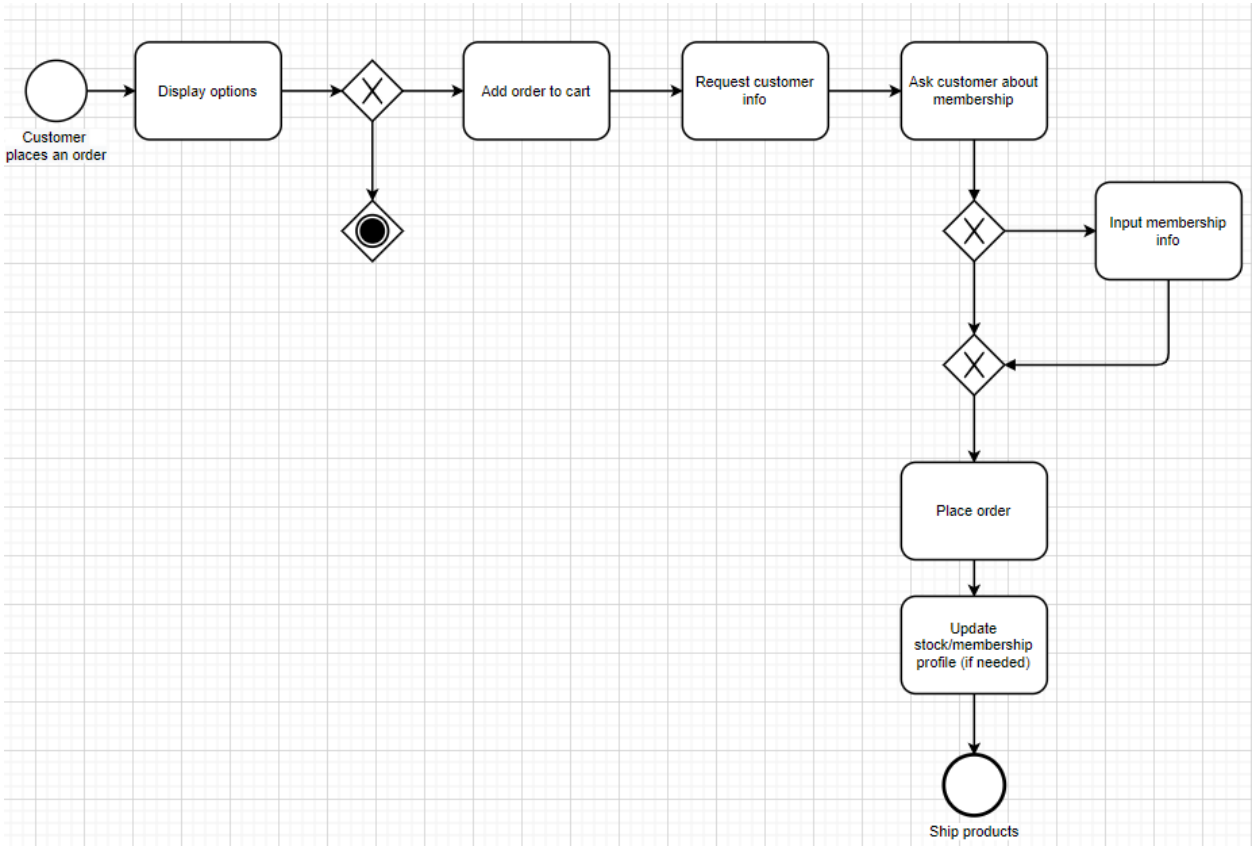
Project Plan:
Lavish Lava Lamps

Business Application:

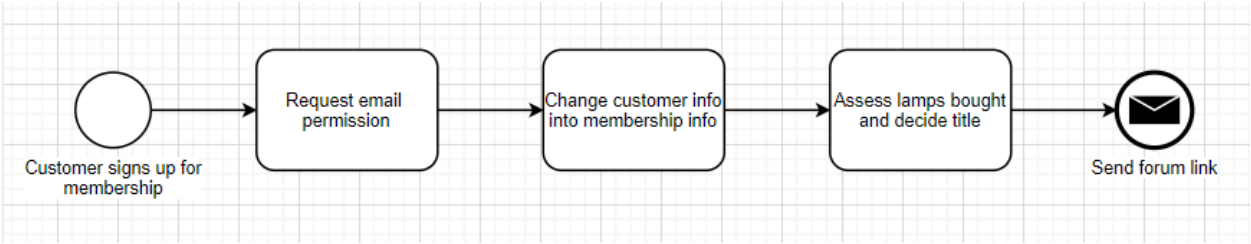
Our business application is an inventory and customer management system for a direct-to-consumer lava lamp store, Lavish Lava Lamps. Our program will help Lavish Lava Lamps view and store information on their lamp inventory, customer orders, customers, and those customers that choose to become members.

Scenarios:

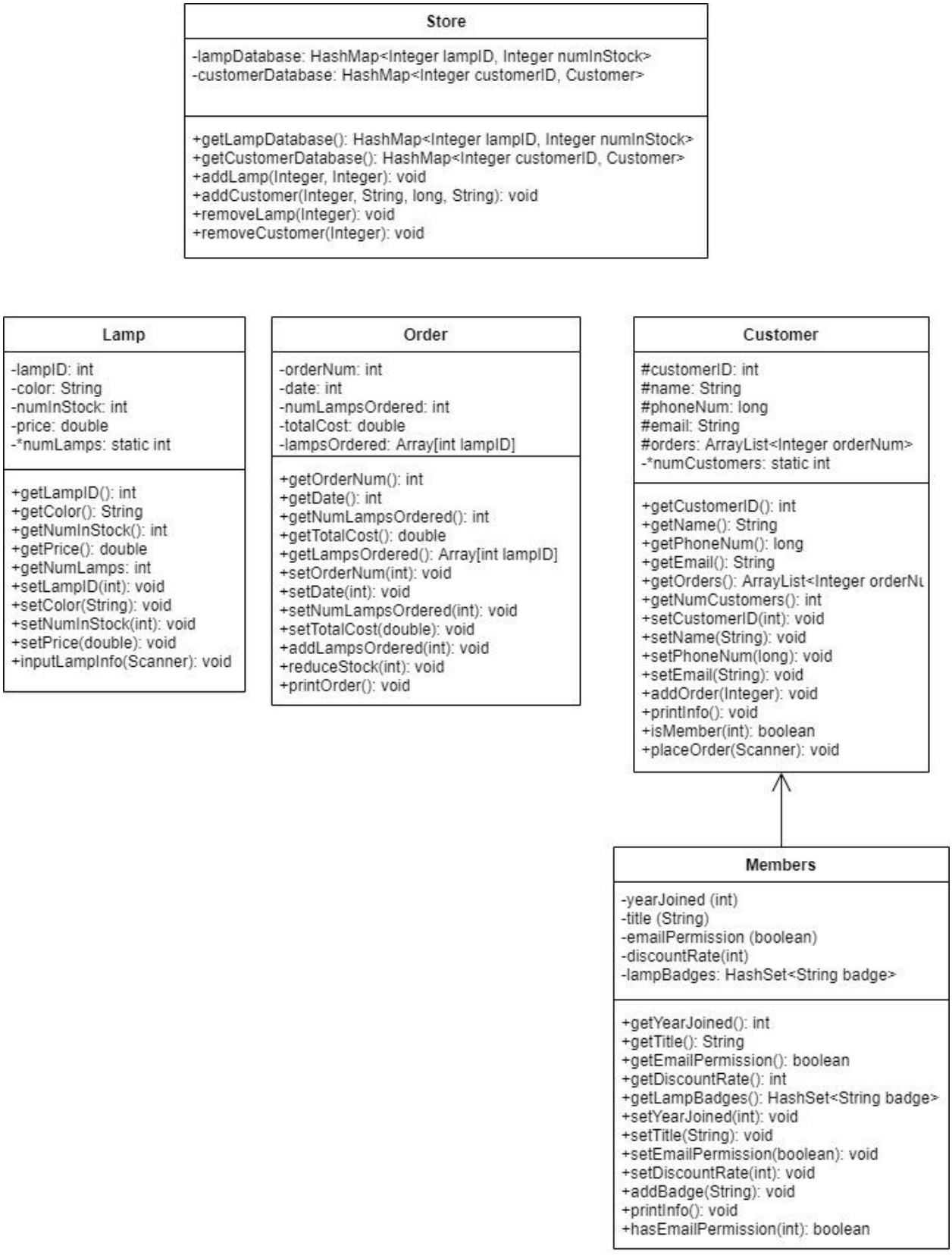
1. One of our scenarios will detail the process of a customer placing an order and Lavish Lava Lamps processing that order.



2. Our other scenario details when a customer signs up for a membership.



UML Diagram (on next page):
Google Drive Link to UML Diagram (with commenting access)
Note: Static attributes should be italicised, but the diagramming software is not friendly to text-editing.


Store
-lampDatabase: HashMap<Integer lampID, Integer numInStock>
-customerDatabase: HashMap<Integer customerID, Customer>
+getLampDatabase(): HashMap<Integer lampID, Integer numInStock>
+getCustomerDatabase(): HashMap<Integer customerID, Customer>
+addLamp(Integer, Integer): void
+addCustomer(Integer, String, long, String): void
+removeLamp(Integer): void
+removeCustomer(Integer): void
Lamp
-lampID: int
-color: String
-numInStock: int
-price: double
-*numLamps: static int
+getLampID(): int
+getColor(): String
+getNumInStock(): int
+getPrice(): double
+getNumLamps: int
+setLampID(int): void
+setColor(String): void
+setNumInStock(int): void
+setPrice(double): void
+inputLampInfo(Scanner): void
Order
-orderNum: int
-date: int
-numLampsOrdered: int
-totalCost: double
-lampsOrdered: Array[int lampID]
+getOrderNum(): int
+getDate(): int
+getNumLampsOrdered(): int
+getTotalCost(): double
+getLampsOrdered(): Array[int lampID]
+setOrderNum(int): void
+setDate(int): void
+setNumLampsOrdered(int): void
+setTotalCost(double): void
+addLampsOrdered(int): void
+reduceStock(int): void
+printOrder(): void
Customer
#customerID: int
#name: String
#phoneNum: long
#email: String
#orders: ArrayList<Integer orderNum>
-*numCustomers: static int
+getCustomerID(): int
+getName(): String
+getPhoneNum(): long
+getEmail(): String
+getOrders(): ArrayList<Integer orderNu
+getNumCustomers(): int
+setCustomerID(int): void
+setName(String): void
+setPhoneNum(long): void
+setEmail(String): void
+addOrder(Integer): void
+printInfo(): void
+isMember(int): boolean
+placeOrder(Scanner): void
Members
-yearJoined (int)
-title (String)
-emailPermission (boolean)
-discountRate(int)
-lampBadges: HashSet<String badge>
+getYearJoined(): int
+getTitle(): String
+getEmailPermission(): boolean
+getDiscountRate(): int
+getLampBadges(): HashSet<String badge>
+setYearJoined(int): void
+setTitle(String): void
+setEmailPermission(boolean): void
+setDiscountRate(int): void
+addBadge(String): void
+printInfo(): void
+hasEmailPermission(int): boolean

Team Contract – MIST 4600 Final Group Project

Group #: 9 **Lecture Time:** 11:10am **Date:** 11/10/20

GOALS: What are our team goals for this project? **What do we want to accomplish? What skills do we want to develop or refine?**
Our team goal is to utilize our skills from this class to create a fun project detailing the inventory of a Lava Lamp store. We want to develop and refine a comfort with collections, object-oriented programming, and inheritance.
EXPECTATIONS: What is our means of communication? How often will we meet? What do we expect of one another in regard to frequency of communication, attendance of meetings, sharing of the workload, the quality of work, etc.?
Our group has a GroupMe for main communication, and we will have meetings on Zoom. We will meet at least twice a week during class time, and we will meet more if we need to. As we are a small group, we will expect attendance, and we will divide the project based on our strengths. We will be sure to review and discuss any parts of the project we are unsure about to maintain a good quality of work.
POLICIES & PROCEDURES: What rules can we agree on to help us meet our goals and expectations?
We will communicate promptly when any issues (with scheduling, workload, etc.) come up, so that our group is on the same page about our project progress. We will try to respect group meeting times and the project plan to stay on course.

CONSEQUENCES: How will we address non-performance in regard to these goals, expectations, policies and procedures?
If any issues come up, we will communicate clearly and honestly and try to develop new group procedures to better fit everyone.

We share these goals and expectations, and agree to these policies, procedures, and consequences.

eSignatures:

Claudia Pinou

Cole Phillips

Xander Davis

Link to UML Diagram:

https://drive.google.com/file/d/1Vsxw2A01sGbFTP6qw4g70BAcqpfQa0l4/view?usp=sharing

Code

Store Class:

```
import java.util.HashMap;


public class Store {


	//Store Attributes

	private HashMap<Integer, Integer> lampDatabase; //HashMap storing lampID and numInStock

	private HashMap<Integer, Customer> customerDatabase; //HashMap storing customerID and Customer objects


	//Default Constructor

	public Store()

	{

		lampDatabase = new HashMap<Integer, Integer>();

		customerDatabase = new HashMap<Integer, Customer>();

	}


	//Accessors and Mutators

	public HashMap<Integer, Integer> getLampDatabase()
```

```
{
        return lampDatabase;
}


public HashMap<Integer, Customer> getCustomerDatabase()
{
        return customerDatabase;
}

public void addLamp(Integer lampID, Integer numInStock)
{
        lampDatabase.put(lampID, numInStock);
}

public void addCustomer(String name, long phoneNum, String email)
{
        Customer c = new Customer(name, phoneNum, email);
        customerDatabase.put(c.getCustomerID(), c);
}

public void removeLamp(Integer lampID)
{
```

```
            lampDatabase.remove(lampID);
        }

        public void removeCustomer(Integer customerID)
        {
            customerDatabase.remove(customerID);
        }

        public void printLamps()
        {
            for(int lamp : lampDatabase.keySet())
            {
                System.out.println(lamp + "-" + lampDatabase.get(lamp));
            }
        }

        public void printCustomers()
        {
            for(int customer : customerDatabase.keySet())
            {
                System.out.println(customer + "-" +
customerDatabase.get(customer).getName());
            }
        }
```

```
}
```

Lamp Class:

```
import java.util.Scanner;

public class Lamp {

	//Lamp Attributes
	private int lampID;			//ex: 000
	private String lampColor;		//	Green
	private int numInStock;		//	12
	private double price;			//	$10
	private static int lampTracker = 001;

	Scanner scnr = new Scanner(System.in);

	//Default Constructor
	public Lamp() {
		lampID = lampTracker;
```

```
        ++lampTracker;
        lampColor = "TBD";
        numInStock = 0;
        price = 0.0;
}

//Constructor with Parameters
public Lamp(String color, int quantity, double price) {
        lampID = lampTracker;
        lampTracker++;
        lampColor = color;
        numInStock = quantity;
        this.price = price;
}

//Lamp Accessors and Mutators
public int getLampID() {
        return lampID;
}

public void setLampID(int ID) {
        this.lampID = ID;
}
```

```
public String getLampColor() {
        return lampColor;
}

public void setLampColor(String color) {
        this.lampColor = color;
}

public int getNumInStock() {
        return numInStock;
}

public void setNumInStock(int quantity) {
        this.numInStock = quantity;
}

public double getPrice() {
        return price;
}

public void setPrice(double price) {
        this.price = price;
```

```
    }

        public int getLampTracker() {
                return lampTracker;
        }

    public void printLampInfo() {
        System.out.println("Lamp ID: "+ lampID);
        System.out.println("Lamp Color: "+ lampColor);
        System.out.println("Lamps in stock: "+ numInStock);
        System.out.println("Lamp price: "+ price);
    }

    public void printStock(Scanner scnr) {
        Lamp blueLamps = new Lamp("Blue", 10, 15.0);
                Lamp redLamps = new Lamp("Red", 15, 12.0);
                Lamp yellowLamps = new Lamp("Yellow", 5, 18.0);
                System.out.println("Here are our lamps: ");
                blueLamps.printLampInfo();
                System.out.println("--------------------------");
                redLamps.printLampInfo();
                System.out.println("--------------------------");
                yellowLamps.printLampInfo();
```

```
                System.out.println("--------------------------");
    }


}
```

Order Class:

```
import java.util.Scanner;


public class Order {


                //Order Attributes
                private int orderNum; //specific order
                private int date;
                private double totalCost;     //total cost of all the lamps in the order
                private int numLamps;                 //total number of lamps ordered --> size
for Array lampsOrdered
                private int[] lampsOrdered; //Array storing lampIDs of the lamps ordered
                private static int orderTracker = 100;
```

```
Scanner scnr = new Scanner(System.in);

//Default Constructor
public Order() {
        orderNum = orderTracker;
        orderTracker++;
        date = 0;
        totalCost = 0.0;
        numLamps = 10;
        lampsOrdered = new int[numLamps];
}

//Constructor with Parameters
public Order(int date,  double totalCost, int numLamps) {
        this.orderNum = orderTracker;
        orderTracker++;
        this.date = date;
        this.totalCost = totalCost;
        this.numLamps = numLamps;
        this.lampsOrdered = new int [numLamps];
}

//Order Accessors and Mutators
```

```
public int getOrderNum(){
        return orderNum;
}

public void setOrderNum (int orderNum) {
        this.orderNum = orderNum;
}

public int getDate() {
        return date;
}

public void setDate(int date) {
        this.date = date;
}


public void setNumLamps(int numLamps) {
        this.numLamps = numLamps;
}

public int getNumLamps() {
        return numLamps;
```

```
}

public void setTotalCost (double cost) {
        this.totalCost = cost;
}

public double getTotalCost() {
        return totalCost;
}

public int getOrderTracker() {
        return orderTracker;
}

public void addLampsOrdered (Scanner scnr) {
        for(int i =0; i < lampsOrdered.length; i++)
        {
                int lampID = scnr.nextInt();
                lampsOrdered[i] = lampID;
        }
}

public void getLampsOrdered()
```

```
        {
            System.out.println("Lamps Ordered: ");
            for(int i = 0; i < lampsOrdered.length; i++)
            {
                System.out.print(i + ", ");
            }
        }

        public void printOrder() {
            System.out.println("Order Number: " + orderNum);
            System.out.println("Date: "+ date);
            System.out.println("Total Cost: $"+ totalCost);
            System.out.println("Number of Lamps: "+ numLamps);
        }

        public void placeOrder(int numLamps, Scanner scnr) {
            Order o = new Order(11302020, 0.0, numLamps);
            double tc = 0.0;
            System.out.println("Which lamps would you like to order? Enter
Lamp ID");
            for(int i = 0; i < numLamps; i++)
            {
                int tempID = scnr.nextInt();
                if(tempID == 2)
```

```
                {
                        tc = tc + 15.0;
                }
                if(tempID == 3)
                {
                        tc = tc + 12.0;
                }
                if(tempID == 4)
                {
                        tc = tc + 18.0;
                }
            }
            System.out.println("Your total cost: $" + tc);
            o.setTotalCost(tc);
            System.out.println("Please re-enter your order (lamp IDs).");
            o.addLampsOrdered(scnr);
            o.printOrder();
        }


}
```

Customer Class:

```
import java.util.ArrayList;

import java.util.Scanner;


public class Customer {


	//Customer Attributes

	protected int customerID;

	protected String name;

	protected long phoneNum;

	protected String email;

	protected ArrayList<Integer> orders; //ArrayList storing the orderNums associated with this Customer, in order bought

	private static int customerTracker = 1000;


	Scanner scnr = new Scanner(System.in);


	//Default Constructor

	public Customer()

	{

		customerID = customerTracker;

		name = "TBD";

		phoneNum = 0000000000;
```

```
        email = "tbd@lamps.com";
        orders = new ArrayList<Integer>();
        customerTracker++;
}

//Constructor with Parameters
public Customer(String name, long phoneNum, String email)
{
        customerID = customerTracker;
        this.name = name;
        this.phoneNum = phoneNum;
        this.email = email;
        orders = new ArrayList<Integer>();
        customerTracker++;
}

//Customer Accessors and Mutators
public int getCustomerID()
{
        return customerID;
}

public String getName()
```

```
{
        return name;
}

public long getPhoneNum()
{
        return phoneNum;
}

public String getEmail()
{
        return email;
}

public ArrayList<Integer> getOrders()
{
        return orders;
}

public static int getNumCustomers()
{
        return customerTracker;
}
```

```
public void setCustomerID(int customerID)
{
	this.customerID = customerID;
}

public void setName(String name)
{
	this.name = name;
}

public void setPhoneNum(long phoneNum)
{
	this.phoneNum = phoneNum;
}

public void setEmail(String email)
{
	this.email = email;
}

public void addOrder(Integer orderNum)
{
```

```
        orders.add(orderNum);
}

public void printInfo()
{
        System.out.println("ID: " + customerID);
        System.out.println("Name: " + name);
        System.out.println("Phone Number: " + phoneNum);
        System.out.println("Email: " + email);
        System.out.print("Past Orders: ");
        for(int i = 0; i < getOrders().size(); i++)
        {
                System.out.print(orders.get(i));
                System.out.print(", ");
        }
}

public void placeOrder(Scanner scnr)
{
        //Prompt for and store customer information
        System.out.println("Please enter your name.");
        this.name = scnr.next();
        System.out.println("Please enter your phone number.");
```

```
this.phoneNum = scnr.nextLong();
System.out.println("Please enter your email.");
this.email = scnr.next();
//Ask to become member and create member object
boolean validAnswer = false;
while(validAnswer == false)
	{
		System.out.println("Do you want to become a member and receive emails with discount? Y/N");
		String memAnswer = scnr.next();
		if(memAnswer.equals("Y"))
		{
			validAnswer = true;
			Member mem = new Member(2020, true);
			mem.createNewMember(scnr);
		}
		else if (memAnswer.equals("N")) {
			validAnswer = true;
			System.out.println("Goodbye!");
			break;
		}
		else
		{
```

```
                                        System.out.println("Invalid response. Please type
either Y or N");
                                        System.out.println("");
                                }
                        }
        }


}
```

Member Class:

```
import java.util.HashSet;
import java.util.Scanner;

public class Member extends Customer {

        //Member Attributes
        private int yearJoined;
        private String title;
        private boolean emailPermission;
        private double discountRate;
```

```
	private HashSet<String> lampBadges; //HashSet storing what "badges" they've earned (based on different lamp colors)


	Scanner scnr = new Scanner(System.in);


	//Default Constructor
	public Member()
	{
		yearJoined = 0000;
		title = "TBD";
		emailPermission = false;
		discountRate = 0.0;
		lampBadges = new HashSet<String>();
	}


	//Constructor with Parameters
	public Member(int yearJoined, boolean emailPermission)
	{
		this.yearJoined = yearJoined;
		this.emailPermission = emailPermission;
		lampBadges = new HashSet<String>();
	}


	//Member Accessors and Mutators
```

```
public int getYearJoined() {
	return yearJoined;
}

public String getTitle() {
	return title;
}

public boolean isEmailPermission() {
	return emailPermission;
}

public double getDiscountRate() {
	return discountRate;
}

public HashSet<String> getLampBadges() {
	return lampBadges;
}

public void setYearJoined(int yearJoined) {
	this.yearJoined = yearJoined;
}
```

```
public void setTitle(String title) {
	this.title = title;
}

public void setEmailPermission(boolean emailPermission) {
	this.emailPermission = emailPermission;
}

public void setDiscountRate(double discountRate) {
	this.discountRate = discountRate;
}

//Add badges and assess to determine title
public void addLampBadge(String badge) {
	for(String existingBadges : lampBadges)
	{
		if(badge.equalsIgnoreCase(existingBadges))
		{
			System.out.println("You already have this badge!");
			break;
		}
		else
```

```
                continue;
        }
                lampBadges.add(badge);
                if(lampBadges.size() == 1)
                {
                        this.title = "Novice";
                        this.discountRate = 0.05;
                }
                if(lampBadges.size() == 2)
                {
                        this.title = "Hobbyist";
                        this.discountRate = 0.10;
                }
                if(lampBadges.size() == 3)
                {
                        this.title = "Aficionado";
                        this.discountRate = 0.20;
                }
}

public Member createNewMember(Scanner scnr) {
        System.out.println("Hello! Would you like to become a member? Y/N");
        Member m = new Member();
```

```
            if(scnr.next().equalsIgnoreCase("Y"))
            {
                System.out.println("Great! Can we use your email for discounts and
Lavish Lamps news? Y/N");
                if(scnr.next().equalsIgnoreCase("Y"))
                {
                    m.setEmailPermission(true);
                }
                else
                {
                    m.setEmailPermission(false);
                }
                System.out.println("Which badges have you earned?");
                System.out.println("Have you bought a blue lamp? Papa Smurf!
(Y/N)");
                if(scnr.next().equalsIgnoreCase("Y"))
                {
                    m.addLampBadge("Papa Smurf!");
                }
                System.out.println("Have you bought a red lamp? A True Bulldawg!
(Y/N)");
                if(scnr.next().equalsIgnoreCase("Y"))
                {
                    m.addLampBadge("A True Bulldawg!");
                }
```

```
                System.out.println("Have you bought a yellow lamp? Sunny
Disposition!");

                if(scnr.next().equalsIgnoreCase("Y"))

                {

                        m.addLampBadge("Sunny Disposition!");

                }

                System.out.println("Thank you! Here is the link to our forum of other
lava lamp lovers: www.LavishLampLovers.com");

                System.out.println("Join the conversation!");

        }

        else

        {

                System.out.println("Okay. Goodbye!");

        }

        return m; //In Main method, have to create a Member object to run
createNewMember method but also another Member object

                                //to store this info and print it out later

}


@Override

public void printInfo()

{

        super.printInfo();

        System.out.println();

        System.out.println("Year Joined: " + yearJoined);
```

```
            System.out.println("Title: " + title);
            if (emailPermission == true)
            {
                System.out.println("Email Permission: Yes");
            }
            else
            {
                System.out.println("Email Permission: No");
            }
            System.out.println("Discount Rate: " + discountRate);
            System.out.print("Lamp Badges Earned: ");
            for(String badges : lampBadges)
            {
                System.out.print(badges + ", ");
            }
        }


}
```

Fake Testing Data

Lamps:

- LampID: 001
  - Color: Blue
  - Num in Stock: 10
  - Price: $15
- LampID: 002
  - Color: Red
  - Num in Stock: 15
  - Price: $12
- LampID: 003
  - Color: Yellow
  - Num in Stock: 5
  - Price: $18
- TotalLampCount = 30

Customers:

- CustomerID: 1000
  - Name: Xander
  - PhoneNum: 123 456 7890
  - Email: XD@LL.com
  - Orders: [100, 102]
- CustomerID: 1001
  - Name: Cole
  - PhoneNum: 098 765 4321
  - Email: CP@LL.com
  - Orders: [101]
- CustomerID: 1002
  - Name: Claudia
  - PhoneNum: 654 123 0987
  - Email: CEP@LL.com
  - Orders: [103]
- CustomerID: 1003
  - Name: Dr. Aguar
  - PhoneNum: 789 432 1506
  - Email: drA@uga.edu
  - Orders: [104]

Member:

Lamp Badges:

Blue: Papa Smurf!

Red: A True Bulldawg!

Yellow: Sunny Disposition!

Titles: Num of badges earned-

Novice: one

Hobbyist: two

Aficionado: three

Discount:

Novice- 5% off

Hobbyist- 10% off

Aficionado- 20% off

- CustomerID: 1000
  - Name: Xander
  - PhoneNum: 123 456 7890
  - Email: XD@LL.com
  - Orders: [100, 101]
  - Year Joined: 2019
  - Title: Aficionado
  - Email Permission: Yes
  - Discount Rate: 20%
  - LampBadges: {“Papa Smurf!”, “A True Bulldawg!”, “Sunny Disposition!”}
- CustomerID: 1003
  - Name: Dr. Aguar
  - PhoneNum: 789 432 1506
  - Email: drA@uga.edu
  - Orders: [102]
  - Year Joined: 2020
  - Title: Novice
  - Email Permission: No

    - Discount Rate: 5%
    - LampBadges: {“A True Bulldawg!”}

Order:

- OrderNum: 100
    - Date: 12312019
    - TotalCost: $45
    - NumLamps: 3
    - LampsOrdered[001, 002, 003]
- OrderNum: 101
    - Date: 01312020
    - TotalCost: $12
    - NumLamps: 1
    - LampsOrdered: [002]
- OrderNum: 102
    - Date: 03142020
    - TotalCost: $15 (there should be a discount here of 20% off)
    - NumLamps: 1
    - LampsOrdered: [001]
- OrderNum: 103
    - Date: 06142020
    - TotalCost: $18
    - NumLamps: 1
    - LampsOrdered: [003]
- OrderNum: 104
    - Date: 08142020
    - TotalCost: $24
    - NumLamps: 2
    - LampsOrdered: [002, 002]